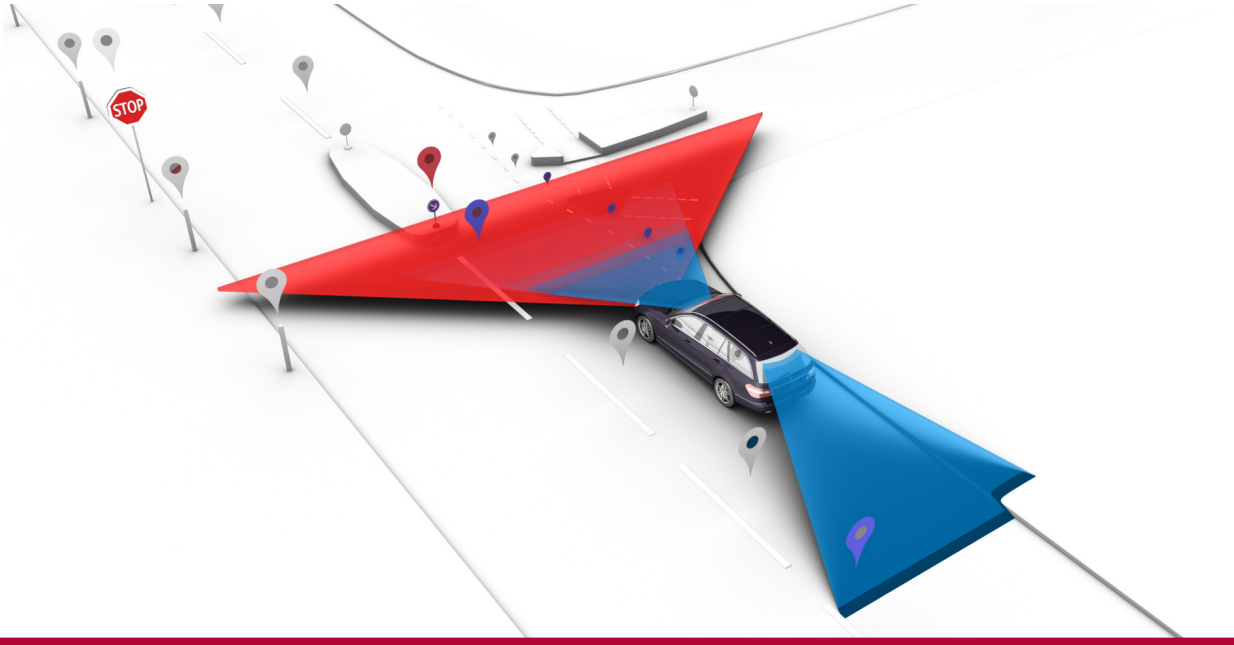
ulm university universität
uulm



Random Finite Set-Based Localization and SLAM for Highly Automated Vehicles

Hendrik Deusch

Band 17

Schriftenreihe des Instituts für Mess-, Regel- und Mikrotechnik

Hendrik Deusch

Random Finite Set-Based Localization and SLAM for Highly Automated Vehicles

Schriftenreihe des
Instituts für Mess-, Regel- und Mikrotechnik
Universität Ulm

Herausgeber:
Prof. Dr.-Ing. Klaus Dietmayer

Band 17

Hendrik Deusch

Random Finite Set-Based Localization and SLAM for Highly Automated Vehicles

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar.

Dissertation, Universität Ulm,
Fakultät für Ingenieurwissenschaften, Informatik und Psychologie, 2015

Impressum

Universität Ulm
Institut für Mess-, Regel- und Mikrotechnik
Prof. Dr.-Ing. Klaus Dietmayer
Albert-Einstein-Allee 41
89081 Ulm
http://www.uni-ulm.de/mrm

Eine Übersicht über alle Bände der Schriftenreihe finden Sie unter
http://www.uni-ulm.de/mrmschriften.

Institut für Mess-, Regel- und Mikrotechnik der Universität Ulm 2016
Print on Demand

ISBN 978-3-941543-22-5 e-ISBN 978-3-941543-23-2

ulm university universität
uulm

Random Finite Set-Based Localization and SLAM for Highly Automated Vehicles

DISSERTATION

zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS
(Dr.-Ing.)

der Fakultät für Ingenieurwissenschaften, Informatik
und Psychologie der Universität Ulm

von

Hendrik Deusch
aus Stuttgart

Gutachter:	Prof. Dr.-Ing. Klaus Dietmayer
	Prof. Dr.-Ing. Christoph Stiller
Amtierende Dekanin:	Prof. Dr. Tina Seufert

Ulm, 18.12.2015

Acknowledgements

This thesis is the product of my work at the Daimler Research Institute for Vehicle Environment Perception at Ulm University (driveU). It would not have been possible to complete this dissertation without the contributions of a many different people. First and foremost, I would like to express my gratitude to my supervisor Prof. Dr.-Ing. Klaus Dietmayer, for his constant support and guidance as well as for providing such an excellent working environment. Besides my advisor, I would like to thank Prof. Dr.-Ing. Christoph Stiller for being the second examiner of this thesis. Further, I am thankful to Dr.-Ing. Martin Fritzsche for supporting my thesis project on behalf of the Daimler AG.

I would also like to thank Dr.-Ing. Michael Buchholz and Jürgen Remmlinger for all the effort they put into the LaTeXtemplate that I used for this thesis – it saved me a lot of time. My sincere thanks also goes to Claudia Fricke for taking care of all administrative things during my time at the Institute of Measurement, Control and Microtechnology as well as the institute's technical staff Oliver Betz, Franz Degenhard, Thomas Löffler and Martin Nieß for the time they took to modify our test vehicle.

Although I am thankful to all my colleagues at the institute, I want to emphasize my special gratitude to two of these colleagues: Stephan Reuter and Jürgen Wiest, with whom I have spent hours and hours over technical and scientific discussions, solving problems (especially with our beloved Cassandra) but also with just having fun together. Working with such inspiring people has been a great honor for me. Further, I would want to thank Felix Kunz, Daniel Meißner, Dominik Nuß, Elias Strigel, Manuel Stübler and Benjamin Wilking for the enriching discussions, for proof-reading this thesis, and for making my time in Ulm much more worthwhile.

I would also like to thank my family, my sister and especially my parents who always supported my academic journey. Finally – and most importantly – I want to thank my wife and best friend Lea for all her love, considerateness and patience during the completion of this thesis (and also for the linguistic improvements).

Abstract

The majority of research efforts in the automotive community is currently focused on alternative fuel vehicles and highly automated driving. The development of highly automated vehicles is not only promoted by Original Equipment Manufacturers (OEMs) and automotive suppliers, but especially by well-known IT companies (e.g., Google and NVIDIA). That alone is an indicator for the potential of this direction of research. Despite these combined efforts, highly automated driving will not be available on arbitrary roads in the foreseeable future. This would require an actual understanding of complex driving situations at a level that is currently not possible to implement on computers. Typically, very detailed maps are used in order to still allow for highly automated driving. These maps provide plenty of information about the stationary vehicular environment that could not be detected with equivalent accuracy relying on sensors only. Furthermore, using detailed maps instead of trying to calculate everything online significantly reduces computational cost. However, the exact pose of the vehicle relative to such a map is a mandatory prerequisite in order to facilitate a sensible usage of this source of information.

This thesis presents a localization method for highly automated vehicles that are equipped with close-to-market sensors. This localization method is based on the well known Monte Carlo Localization (MCL) algorithm, i.e., in this case the feature-based estimation of the vehicle's pose using a particle filter. When using MCL, two aspects are of major importance for reaching good performance: The features and the function that is used to determine the particle weights. Features that are especially suitable for the localization of a highly automated vehicle are described within this thesis. These features are provided by different sensors in order to reach a certain degree of redundancy which is definitely needed for safety relevant systems such as highly automated vehicles. The function that is used for determining the particle weights is inspired by Random Finite Set (RFS) Theory which has also been the cornerstone of recent progress in multi-object tracking. Using RFSs to represent the set of measurements and the set of mapped landmarks allows for a mathematically sound method to describe the similarity between those sets, thereby also considering measurement statistics (detection probability, false alarm rate).

The proposed localization approach has been utilized successfully during public demonstrations of the highly automated vehicle of the Institute of Measurement, Control and Microtechnology at Ulm University, Germany.

The second part of this thesis presents a novel approach to Simultaneous Localization

And Mapping (SLAM). SLAM algorithms have the potential to facilitate generating and updating landmark maps with any vehicle that is capable of detecting the respective landmarks. Currently, only vehicles that are equipped with expensive reference systems can be used for this purpose. The proposed SLAM approach is based on RFS theory as well: Each particle of a Rao-Blackwellized particle filter uses an Labeled Multi-Bernoulli filter to estimate the map of landmarks (RB-LMB-SLAM). It is demonstrated by means of simulation that – in cases of high clutter rates – the RB-LMB-SLAM approach outperforms the RB-PHD-SLAM which is well known for its performance in this kind of scenario.

Kurzfassung

Ein sehr großer Teil der aktuellen Automobilentwicklung konzentriert sich auf zwei Bereiche: Die Abkehr von fossilen Brennstoffen und das hochautomatisierte Fahren. Die Entwicklung hin zu hochautomatisierten Fahrzeugen wird dabei nicht nur von den großen OEMs und Zulieferern der Automobilbranche intensiv vorangetrieben, sondern insbesondere auch von namhaften IT-Unternehmen wie Google und NVIDIA. Allein daran schon lässt sich das Potenzial dieses Forschungsbereichs ermessen. Hochautomatisierte Fahrzeuge werden jedoch auf absehbare Zeit noch nicht in der Lage sein beliebige Strecken zu fahren. Dies würde ein tatsächliches Verständnis für komplexe Fahrsituationen voraussetzen, wie es bisher noch nicht von Algorithmen abgebildet werden kann. Um dennoch hochautomatisiertes Fahren zu ermöglichen, werden typischerweise sehr detaillierte Karten verwendet. Diese Karten stellen sehr viele Informationen über das stationäre Fahrzeugumfeld zur Verfügung, welche zum Teil von Sensoren nicht zuverlässig genug erfasst werden können. Zudem kann durch Karteninformationen ein Großteil der ansonsten nötigen Berechnungen eingespart werden. Damit eine Karte jedoch sinnvoll genutzt werden kann, ist es notwendig die exakte Pose des Fahrzeugs relativ zu dieser zu kennen.

Diese Arbeit präsentiert eine Methode zur Lokalisierung für hochautomatisierte Fahrzeuge, die mit seriennaher Sensor-Technik ausgestattet sind. Dabei wird auf das bekannte Monte-Carlo-Lokalisierungsverfahren – in diesem Fall also die merkmalsbasierte Schätzung der Fahrzeugpose mit Hilfe eines Partikelfilters – zurückgegriffen. Für den Erfolg des Verfahrens sind im wesentlichen zwei Faktoren entscheidend: Die verwendeten Merkmale bzw. Landmarken sowie die Funktion zur Bestimmung der Partikelgewichte. In dieser Arbeit werden Landmarken beschrieben, die besonders zur Lokalisierung von hochautomatisierten Fahrzeugen geeignet sind. Dabei wird auch die notwendige Redundanz beachtet, indem Landmarken, die von unterschiedliche Sensoren detektiert werden können, genutzt werden. Die entwickelte Funktion zur Berechnung der Partikelgewichte beruht auf der Random-Finite-Set-Theorie, welche auch die Grundlage für große Fortschritte im Multi-Objekt-Tracking der vergangenen Jahre ist. Durch eine entsprechende Modellierung kann die Ähnlichkeit der Menge der gemessenen Merkmale und der Menge der in der Karte verzeichneten Landmarken theoretisch fundiert bewertet werden, wobei Falschalarme und Fehldetektionen implizit berücksichtigt werden.

Das im Rahmen dieser Arbeit entwickelte Verfahren wurde erfolgreich für die Lokalisierung des hochautomatisierten Fahrzeugs des Instituts für Mess-, Regel- und

Mikrotechnik der Universität Ulm eingesetzt – auch bei öffentlichen Präsentationen. Der zweite Teil dieser Arbeit befasst sich mit einer neuartigen Methode zur simultanen Lokalisierung und Kartenerstellung – engl. Simultaneous Localization And Mapping (SLAM). SLAM-Verfahren könnten das Erstellen und Aktualisieren von Karten mit jedem Fahrzeug ermöglichen, das in der Lage ist die entsprechenden Landmarken zu detektieren. Bislang wird dazu teure Referenzmesstechnik benötigt, um die wahre Fahrzeugpose zu bestimmen. Das hier präsentierte SLAM-Verfahren beruht ebenfalls auf der Random-Finite-Set-Theorie. Konkret wird für jedes Partikel eines Rao-Blackwellized Partikelfilter ein Labeled Multi-Bernoulli Filter verwendet (RB-LMB-SLAM), um die Positionen der Landmarken zu schätzen. Anhand simulierter Daten wird gezeigt, dass der RB-LMB-SLAM Ansatz mit hohen Falschalarmraten besser zurecht kommt als das RB-PHD-SLAM-Filter, welches bislang als das beste Verfahren bei hohen Falschalarmraten gilt.

Contents

Chapter 1

Introduction

Highly automated driving has been a dream ever since the car has been available to a broad public. Although the first ideas were pure science fiction at that time, the advantages of having automated vehicles were already clear (cf. Figure 1.1): more safety, less traffic jams, relaxed traveling and saving time. The ideas towards autonomous driving presented in the fifties and sixties focused primarily on making roads smarter, instead of the vehicles. It was not before the mid to late eighties that the first research projects, lead by the pioneering work of Ernst Dickmanns, actually produced cars that reached a certain degree of automation [DMC90]. Building up on these advances, the pan-European Eureka PROMETHEUS Project (PROgraMme for a European Traffic of Highest Efficiency and Unprecedented Safety) from 1987 to 1995 lead to an autonomous drive from Munich, Germany, to Odense in Denmark in 1995. In the first decade of the 21st century, especially the Defense Advanced Research Projects Agency (DARPA) pushed the development of highly automated vehicles forward by organizing its Grand Challenges [GLM+12; TMD+06]. In recent years autonomous driving has become one of the hottest research topics. Several research teams have publicly demonstrated their highly automated vehicles, e.g., by a drive through downtown Parma [BBD+13] or by an autonomous drive in Germany on the Bertha Benz memorial route by Mercedes-Benz [ZBS+14].
At least in the foreseeable future, highly automated vehicles will not be able to actually understand arbitrary traffic scenarios as humans do by intuition. Hence, these systems need a reliable source of information that supports them in understanding the scene. Typically, detailed maps are used for this purpose. A map contains information about the possible locations of relevant traffic participants and in which direction these may be moving. Further, a road map limits the space in which the vehicle may operate. This significantly narrows down the space of possible solutions that has to be checked by the trajectory planner. Furthermore, also other modules benefit highly from map data: For example, the decision making progress may use the information about the right-of-way at an intersection of a map, instead of interpreting traffic signs and trying to figure out the layout of the intersection from sensor data. Thus,

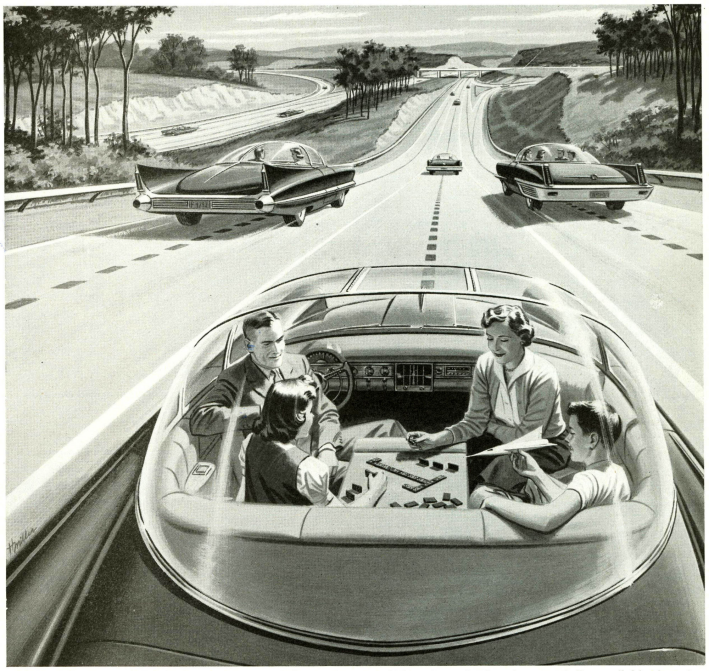


Figure 1.1: "ELECTRICITY MAY BE THE DRIVER. One day your car may speed along an electric super-highway, its speed and steering automatically controlled by electronic devices embedded in the road. Highways will be made safe – by electricity! No traffic jam.. no collisions... no driver fatigue." – Driverless Car of the Future advertisement for "America's Electric Light and Power Companies", 1950s, Image courtesy of The Advertising Archives.

vast amounts of computational power can be saved and the complexity of many modules can be reduced extremely, by simply providing as much information as possible with a map. However, a map is relatively useless, as long as no knowledge about the pose of the vehicle relative to that map is available. Localization aims at finding that exact pose. Hence, it facilitates a significant reduction in the complexity of driving autonomously by making maps actually usable. Thus, it is one prerequisite for highly automated vehicles. Although Global Navigation Satellite System (GNSS) solutions, such as Global Positioning System (GPS), are already used in every modern car, they do not yield the accuracy that is required for highly automated driving. Further, GNSS is also not always available (e.g., in tunnels or in adverse satellite constellations).

One major contribution of this thesis is a feature-based Monte Carlo Localization (MCL) algorithm that uses multiple sensors to estimate the vehicle's pose robustly. The function that was developed to evaluate the likelihood of the pose hypotheses originates from Random Finite Set (RFS) theory and is hence mathematically motivated. This localization algorithm yields an accuracy that facilitates highly automated driving on rural roads and in inner city scenarios. Further, the design of the features being used as landmarks allows for an entirely unsupervised mapping process without the need for post-processing: The vehicle just has to be driven along the route to be mapped while using a reference system for localization and the landmarks are detected and inserted into the database online.
Based on such a localization approach, the next step towards even more autonomy for highly automated vehicles could be taken by utilizing Simultaneous Localization And Mapping (SLAM). Enabling vehicles to create their own maps with SLAM could open up new prospects: While driving the vehicle along a certain route, the vehicle could create a map of the environment and improve this map through multiple journeys on that route by using a SLAM algorithm. At some point, the vehicle would reach a certain confidence in the map and then it could be able to drive autonomously on that route.
Thus, the second major contribution of this thesis is a novel approach to SLAM. This approach has been inspired by recent advances in multi-object tracking using RFSs. The Rao-Blackwellized Labeled Multi-Bernoulli Simultaneous Localization And Mapping (RB-LMB-SLAM) filter utilizes multiple instances of the recently developed Labeled Multi-Bernoulli (LMB) filter [Reu14]. In contrast to most conventional SLAM algorithms, it facilitates an integrated handling of measurement statistics (i.e., false alarm rates and detection probabilities). The only other popular algorithm with this qualities which is also based on RFSs theory is the Rao-Blackwellized Probability Hypothesis Density Simultaneous Localization And Mapping (RB-PHD-SLAM) algorithm. However, the exact update of the LMB filter is superior to that of the Probability Hypothesis Density (PHD) filter. Further, in RB-LMB-SLAM, no additional approximations are required to determine the weights for the Rao-Blackwellized particle filter. It is shown, that the RB-LMB-SLAM approach performs very promising in the presence of high clutter levels – even better than the RB-PHD-SLAM.

This thesis is organized as follows: The next chapter outlines the theoretical foundations that are required for the understanding of the algorithms developed in this work (i.e., for the Chapters 4 and 5). It comprises a description of the particle filter algorithm as well as a short introduction to Finite Set Statistics (FISST) and to state estimation using RFSs. Thereafter, in Chapter 3, the technical preliminaries for this work are described: the highly automated vehicle, its sensors and the software framework running on the vehicle's computers are presented. Further, the required coordinate systems are defined. The chapter concludes with an overview of the database system and its interface which has been developed to facilitate an easy

handling of all relevant geospatial data required by all algorithms running on the highly automated test vehicle. Based on these theoretical and technical foundations, Chapter 4 introduces the feature-based localization algorithm that was designed to facilitate autonomous driving with the test vehicle. The features that are used for localization, the mapping procedure and the localization algorithm itself are described in detail. A possible next step towards full autonomy is outlined in Chapter 5: A novel approach to SLAM using LMB filters is described. This could render the predefined map required for the previously presented localization algorithm expendable, since it could enable the vehicle to create its own map while localizing itself. Finally, a conclusion of this thesis is given.

Chapter 2

Theoretical Foundations

This chapter introduces the theoretical preliminaries that form the basis of the algorithms presented within this thesis. First of all, the notation used throughout this thesis is presented. Then, the particle filter, as a general concept for state estimation, is outlined. Finally, a short introduction to Finite Set Statistics (FISST) is given, laying the ground for the Rao-Blackwellized Random Finite Set based Simultaneous Localization And Mapping (RB-RFS-SLAM) algorithms introduced in Chapter 5.

2.1 Notation and Abbreviations

Throughout this work, state vectors are denoted by bold face lower-case characters and state spaces are denoted using blackboard bold letters (e.g., $\boldsymbol{x}$ is a state vector of the state space $\mathbb{X}$). Random Finite Sets are denoted by capital letters (see Section 2.3.1): A RFS $\mathrm{X} = \{\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(n)}\}$ may represent multiple objects (i.e., $n \geq 0$), each with its respective state $\boldsymbol{x}^{(i)}$. Further, labeled RFSs are denoted by bold face capital letters, e.g., $\mathbf{X}$.
In the following, the letter $\pi(\cdot)$ is used to discern multi-object probability densities of random finite sets from other probability densities which are typically denoted as $p(\cdot)$. Further, labeled multi-object probability densities are printed in bold letters, e.g., $\boldsymbol{\pi}(\cdot)$. For some explanations, the index k is used to describe the time step at which the entity at hand is referred to, e.g. Z_k denotes the RFS describing the measurements of time step k. Nevertheless, whenever possible the time step index is omitted for the sake of readability. Moreover, to distinguish predicted entities from other quantities, the index $+$ is used, for example $p_+(\cdot)$ describes a predicted probability density.
Further, a set in general, but not in particular an RFS, is printed in calligraphic font, e.g., $\mathcal{Z}$ may describe a set of measurements $\boldsymbol{z}^{(j)}$ originating from the measurement

space $\mathbb{Z}$, which again is represented by a blackboard bold letter, as all spaces are. The finite subsets of the space $\mathbb{Z}$ are denoted by $\mathcal{F}(\mathbb{Z})$, while the finite subsets of $\mathbb{Z}$ containing exactly n elements are denoted by $\mathcal{F}_n(\mathbb{Z})$.
Moreover, as proposed in [VV13], the multi-object exponential notation h^{X} is used in order to express that the real-valued function $h(\boldsymbol{x})$ is evaluated for all state vectors $\boldsymbol{x}$ contained in the RFS X, i.e.,

$$h^{\mathrm{X}} \triangleq \prod_{\boldsymbol{x}\in\mathrm{X}} h(\boldsymbol{x}), \tag{2.1}$$

with $h^{\mathrm{X}} = 1$ if $\mathrm{X} = \emptyset$, by definition. Further, the well-known Kronecker delta function is generalized to be applicable for arbitrary arguments, such as e.g., sets:

$$\delta_{\mathrm{Y}}(\mathrm{X}) \triangleq \begin{cases} 1, & \text{if } \mathrm{X} = \mathrm{Y} \\ 0, & \text{otherwise.} \end{cases} \tag{2.2}$$

Additionally, in order to evaluate whether or not a set X is a subset of Y, the inclusion function is used within this thesis:

$$1_{\mathrm{Y}}(\mathrm{X}) \triangleq \begin{cases} 1, & \text{if } \mathrm{X} \subseteq \mathrm{Y} \\ 0, & \text{otherwise.} \end{cases} \tag{2.3}$$

Finally, the inner product of the continuous functions $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ is abbreviated by:

$$\langle f, g \rangle \triangleq \int f(\boldsymbol{x}) g(\boldsymbol{x}) d\boldsymbol{x}. \tag{2.4}$$

2.2 Particle Filtering

The particle filter, or Sequential Monte Carlo (SMC) method, is a non-parametric recursive Bayes filter. The first true implementation of this filtering approach was presented by Gordon et al. in 1993 [GSS93]. In the last decade it has continuously gained attention which is not only due to the elegance of the algorithm – predominantly, it is due to the fact that the constant increase in computing power made the use of this algorithm for sensible applications feasible. The great benefit of this filter is that, in contrast to other filters (e.g., the Kalman filter [Kal60]), it does not make any assumptions about the distribution in the state-space and does not rely on any local linearization technique [TBF05]. Instead of representing the filter posterior by some defined distribution (e.g., a Gaussian distribution), ν_P random samples which may originate from an unknown arbitrary distribution are used as an assumption free posterior representation. These samples are called particles and

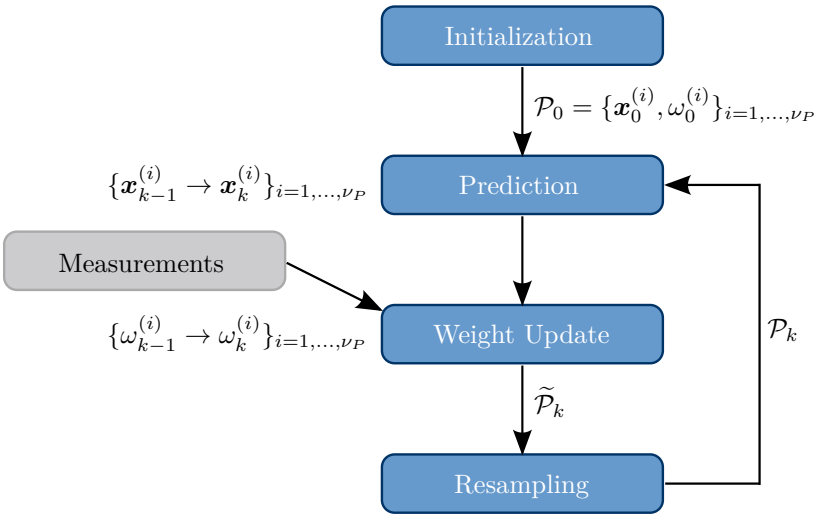


Figure 2.1: The particle filter algorithm.

each of them is a hypothesis about what the true system state may be. Ideally, the denser a sub-region of the state space is populated by particles, the more likely the true system state is found in this sub-region [TBF05].
At time step k, each of the $i = 1, ..., \nu_P$ particles represents a possible system state $\boldsymbol{x}_k^{(i)}$ along with its associated importance factor, which is also referred to as the particle's weight $\omega_k^{(i)}$. The entirety of these particles forms the particle set $\mathcal{P}_k = \{\boldsymbol{x}_k^{(i)}, \omega_k^{(i)}\}_{i=1:\nu_P}$. Obviously, the more particles are used, the better the approximation of the true system state distribution will get. The general idea is to assess those particles based on the measurements that are made. The better the state represented by a particle explains the measurements, the higher the weight associated with this sample becomes. Finally, in order to avoid degeneration of the filter, i.e., all but one particle having a negligible weight, a mechanism called resampling is applied. During resampling particles are drawn (possibly multiple times) with a probability corresponding to their weight. Those drawn particles then form the set of particles for the next time step. Hence, resampling in the particle filter represents a kind of "survival of the fittest" technique.
The particle filter algorithm is depicted in Figure 2.1 and one possible implementation is described by the pseudo-code in Algorithm 1 (cf. [AMGC02; TBF05]). Furthermore, the individual steps are presented in more detail in the following paragraphs.

Algorithm 1 Particle filter algorithm

procedure PARTICLEFILTER($\mathcal{Z}_{1:K_{max}}$, $\boldsymbol{u}_{1:K_{max}}$, ν_P)
  $\mathcal{P}_0 \leftarrow \{\boldsymbol{x}_0^{(i)}, \omega_0^{(i)}\}_{i=1:\nu_P} \sim p_0(\cdot)$
  $k \leftarrow 1$
  while $k < K_{max}$ **do**
    $\mathcal{P}_k \leftarrow$FILTERINGSTEP($\mathcal{Z}_k$, $\boldsymbol{u}_k$, $\mathcal{P}_{k-1}$)
    $k \leftarrow k + 1$
  end while
  return $\mathcal{P}_{1:K_{max}}$
end procedure
procedure FILTERINGSTEP($\mathcal{Z}_k$, $\boldsymbol{u}_k$, $\mathcal{P}_{k-1}$)
  for $i = 1 : \nu_P$ **do**
    $\boldsymbol{x}_k^{(i)} \sim f_+(\boldsymbol{x}_k^{(i)} | \boldsymbol{x}_{k-1}^{(i)}, \boldsymbol{u}_k)$
    $\omega_k^{(i)} \leftarrow \omega_{k-1}^{(i)} \cdot g(\mathcal{Z}_k | \boldsymbol{x}_k^{(i)})$
  end for
  $\omega_{Sum} \leftarrow \sum_{i=1}^{\nu_P} \omega_k^{(i)}$
  for $i = 1 : \nu_P$ **do**
    $\omega_k^{(i)} \leftarrow \omega_k^{(i)} / \omega_{Sum}$
  end for
  $\tilde{\nu}_P^{Eff} \leftarrow 1 / \left(\sum_{i=1}^{\nu_P} (\omega_k^{(i)})^2 \right)$
  if $\tilde{\nu}_P^{Eff} < \nu_P^{min}$ **then**
    $\mathcal{P}_k \leftarrow$RESAMPLE($\mathcal{P}_k$)
  end if
  return $\mathcal{P}_k$
end procedure
procedure RESAMPLE($\mathcal{P}_k$)
  $\widehat{\mathcal{P}}_k \leftarrow \emptyset$
  $r \leftarrow$RANDOM$(0, 1/\nu_P)$
  $c \leftarrow \omega_k^{(1)}$
  $j \leftarrow 1$
  for $i = 1 : \nu_P$ **do**
    $u = r + (i-1) \cdot \nu_P^{-1}$
    while $u > c$ **do**
      $j \leftarrow j + 1$
      $c \leftarrow c + \omega_k^{(j)}$
    end while
    $\widehat{\mathcal{P}}_k \leftarrow \widehat{\mathcal{P}}_k \cup \{\boldsymbol{x}_k^{(j)}, 1/\nu_P\}$
  end for
  return $\widehat{\mathcal{P}}_k$
end procedure

Initialization During initialization, the particle set $\mathcal{P}_0$ is constructed by drawing ν_P samples from the prior distribution p_0 (line 2 in Algorithm 1). The initial weight $\omega_0^{(i)}$ of the particles is usually set to $1/\nu_P$. For many applications an initial guess about the true system is available, so that it is reasonable to draw the samples from a distribution that corresponds to this initial knowledge. To this end e.g., a (multivariate) Gaussian is often utilized. If no a-priori information about the true system state is available, the samples are commonly drawn from a uniform distribution.
Although, the necessary number of particles ν_P heavily depends on the estimation task, it is typically relatively large, e.g., in the order of 1000.

Prediction The prediction stage generates the new hypotheses about the system state by incorporating the previous particles' states and the optional control input $\boldsymbol{u}_k$ (line 12 in Algorithm 1). That is, the new states are drawn from the transition density

$$\boldsymbol{x}_k^{(i)} \sim f_+(\boldsymbol{x}_k^{(i)}|\boldsymbol{x}_{k-1}^{(i)}, \boldsymbol{u}_k). \tag{2.5}$$

In many use cases, this corresponds to the prediction of the particles' states based on a motion model, thereby respecting the given uncertainties by adding system noise.

Weight Update This step uses the measurements to evaluate the particle states, by calculating their importance factors. Hence, it incorporates the information gain that is made through the measurements into the particle set $\mathcal{P}_k$. A typical method to update the particle weight is to combine the previous weight with the likelihood of the new measurement given the particle's state:

$$\omega_k^{(i)} = \omega_{k-1}^{(i)} \cdot g(\mathcal{Z}_k|\boldsymbol{x}_k^{(i)}). \tag{2.6}$$

The choice of a sensible likelihood function $g(\mathcal{Z}_k|\boldsymbol{x}_k^{(i)})$ is of crucial importance to the filtering process. Given this measurement update, the weighted particles then again represent an approximation of the Bayes filter posterior.

Resampling The resampling of the particles is an important step in order to overcome the degeneracy phenomenon: That is, after a few filter iterations, only very few (or even only one) particles would posses a significant weight, since the variance in the importance weights can only increase over time (see [DGA00] for details). Hence, a lot of computational power would be wasted for the calculations required to predict and update particles with negligible weights, if measures to avoid

the degeneracy were omitted. Nevertheless, it is not expedient to resample after each and every update, since this would lead to an immediate loss of diversity. In order to determine if resampling should be executed, the effective number of particles ν_P^{Eff} is typically used (cf. line 19 of Algorithm 1). The actual effective number of particles, as described in [KLW94], cannot be obtained but may very well be approximated through this rather simple equation:

$$\tilde{\nu}_P^{Eff} = \frac{1}{\sum_{i=1}^{\nu_P} (\omega_k^{(i)})^2}. \tag{2.7}$$

As long as this number is above a certain fixed threshold (e.g., 10% of ν_P), it is not necessary to resample.
The resampling procedure described in Algorithm 1 is one possible implementation of the 'low-variance sampling' [TBF05] – other approaches to resampling can be found in [DC05] for example. After this algorithm is executed on the particle set, the distribution of the particles changes: On the one hand, particles with relatively low weights which were contained in the set before the resampling, are no longer existent. On the other hand particles with relatively high weights are represented by multiple copies in the new particle set. Thus, the resampling forces the distribution of the particles back to the posterior (cf. [TBF05]).

2.3 Finite Set Statistics

This section briefly outlines the fundamentals of Finite Set Statistics (FISST) that are required for the methods presented in Chapter 5. FISST enables the application of random finite set theory to the multi-object tracking problem. Within this context, especially the comprehensive work of Mahler [Mah07] is notable. This section first introduces the general idea of RFSs and then presents the types of RFSs that are relevant for this work. Thereafter, the concept of estimating a multi-object state within this framework is described.

2.3.1 General Introduction to Random Finite Sets

In general, an RFS is a set with random cardinality of finite size containing values – or objects – that are also random variables. That is, to be more specific, for a random number $n_M \geq 0$, the RFS of a map of landmarks

$$\mathrm{M} = \{\boldsymbol{m}^{(1)}, ..., \boldsymbol{m}^{(n_M)}\} \tag{2.8}$$

comprises n_M unordered landmarks, each of them exhibiting a random state $\boldsymbol{m}^{(i)}$. Hence, this map representation directly encapsulates the uncertainty in the number of landmarks being present in the map. Further, an RFS can also be used to represent the measurements:

$$\mathrm{Z} = \{\boldsymbol{z}^{(1)}, ..., \boldsymbol{z}^{(m)}\}. \tag{2.9}$$

Employing an RFS as well to describe the measurements makes perfect sense: This way, it is possible to directly consider missed detections and false alarms, which will be very useful for the algorithms presented in the subsequent chapters of this thesis. The multi-object probability density of the RFS X is given by

$$\pi(\mathrm{X}) = \begin{cases} \pi(\emptyset) & \text{if } \mathrm{X} = \emptyset \\ \pi(\{\boldsymbol{x}^{(1)}\}) & \text{if } \mathrm{X} = \{\boldsymbol{x}^{(1)}\} \\ \pi(\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\}) & \text{if } \mathrm{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\} \\ \vdots & \vdots \end{cases} \tag{2.10}$$

This Probability Density Function (PDF) reflects the uncertainty in the number of objects as well as the uncertainty in the individual states of those objects. Integrating over such a PDF requires the calculation of a set integral, cf. [Mah07]:

$$\int \pi(\mathrm{X})\delta\mathrm{X} = \sum_{i=0}^{\infty} \frac{1}{i!} \int_{\mathbb{X}^i} \pi(\{\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(i)}\}) d\boldsymbol{x}^{(1)} \cdots d\boldsymbol{x}^{(i)}, \tag{2.11}$$

with $\pi(\{\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(i)}\}) = 0$ if $|\{\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(i)}\}| \neq i$. This calculation is in general computationally intractable.

Poisson Random Finite Set

Poisson RFSs are used to model false alarm processes throughout this work. Therefore they are described here shortly. The Poisson distribution is a discrete probability distribution. If the number of objects follows a Poisson distribution with an expected number of objects λ, the probability of obtaining exactly n objects (i.e., the cardinality distribution) is:

$$\rho(n) = e^{-\lambda} \cdot \frac{\lambda^n}{n!}. \tag{2.12}$$

A multi-object Poisson RFS is an RFS, where the cardinality distribution is a Poisson distribution and the objects are assumed to be independent and identically distributed. Therefore, such a multi-object Poisson RFS is given by:

$$\pi(\{\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(n)}\}) = e^{-\lambda} \cdot \lambda^n \cdot p(\boldsymbol{x}^{(1)}) \cdots p(\boldsymbol{x}^{(n)}), \tag{2.13}$$

with $p(\boldsymbol{x}^{(i)})$ being the probability density function that represents the spatial distribution of object i.

Multi-Bernoulli Random Finite Set

The Bernoulli RFS facilitates the direct representation of the uncertainty regarding the existence of an object. That is, the Bernoulli RFS X contains exactly one element with the probability r and is empty with the probability $1 - r$. Hence, r is called the existence probability. The spatial distribution of this object is again given by $p(\boldsymbol{x})$. The probability density of the Bernoulli RFS X is:

$$\pi(\mathrm{X}) = \begin{cases} 1 - r, & \text{if } \mathrm{X} = \emptyset \\ r \cdot p(\boldsymbol{x}), & \text{if } \mathrm{X} = \{\boldsymbol{x}\}. \end{cases} \tag{2.14}$$

Based on this Bernoulli RFS, which is able to represent a singleton, the extension to the Multi-Bernoulli RFS, representing multiple objects, is straightforward. Under the assumption that the objects are independent from each other, the Multi-Bernoulli RFS is given by the union of the n_O single Bernoulli RFSs $\mathrm{X}^{(i)}$: $\mathrm{X} = \bigcup_{i=1}^{n_O} \mathrm{X}^{(i)}$. Since one Bernoulli RFS is completely described by its parameters r and $p(\boldsymbol{x})$, the Multi-Bernoulli RFS is defined by the respective parameter set $\{r^{(i)}, p^{(i)}\}_{i=1}^{n_O}$. Determining the probability density of such a multi-Bernoulli RFS involves summing over all possible permutations (with $n \leq n_O$) of the n_O state vectors $\boldsymbol{x}^{(i)}$ (cf. [Mah07]):

$$\pi(\{\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(n)}\}) = \prod_{j=1}^{n_O} \left(1 - r^{(j)}\right) \times \sum_{1 \leq i_1 \neq ... \neq i_n \leq n_O} \prod_{j=1}^{n} \frac{r^{(i_j)} p^{(i_j)}(\boldsymbol{x}^{(j)})}{1 - r^{(i_j)}}. \tag{2.15}$$

The first product accounts for the non-existence of all objects. The sum then basically runs over all permutations of the objects. For each of these realizations, the spatial distributions of the respective objects along with their existence probabilities are considered in the nominator. Further, the denominator cancels out the non-existence of these existing objects that has been assumed by first product.
Such multi-object probability densities of multi-Bernoulli RFSs are abbreviated by their parameters in the following: $\pi = \{r^{(i)}, p^{(i)}\}_{i=1}^{n_O}$.

Labeled Multi-Bernoulli Random Finite Set

If the component indices i of the multi-Bernoulli RFS $\pi = \{r^{(i)}, p^{(i)}\}_{i=1}^{n_O}$ are interpreted as object labels $\ell \in \mathbb{L}$ the corresponding Labeled Multi-Bernoulli (LMB) RFS is obtained. This LMB RFS is a finite set on the space $\mathbb{X} \times \mathbb{L}$ (cf. [VV13]) and can

be described by:

$$\boldsymbol{\pi}(\mathbf{X}) = \{r^{(\ell)}, p^{(\ell)}\}_{\ell \in \mathbb{L}}. \tag{2.16}$$

The respective multi-object probability of such an LMB RFS is determined by:

$$\boldsymbol{\pi}\left(\{(\boldsymbol{x}^{(1)}, \ell_1), ..., (\boldsymbol{x}^{(n)}, \ell_n)\}\right) = \delta_n(|\{\ell_1, ..., \ell_n\}|) \prod_{i \in \mathbb{L}} \left(1 - r^{(i)}\right) \prod_{\ell=1}^{n} \frac{1_{\mathbb{L}}(\ell) r^{(\ell)} p^{(\ell)}(\boldsymbol{x})}{1 - r^{(\ell)}} \tag{2.17}$$

using the inclusion function $1_{\mathbb{L}}(\ell)$ (cf. equation (2.3)). The Kronecker delta function is used to ensure the validity of the labeled RFS: In general, all components of a labeled RFS, and therefore also those of an LMB RFS, are required to have a distinct label, since the idea behind labeling the components is that it facilitates an unambiguous identification of the tracks. If this is not the case, the labeled RFSs have to be marked as invalid. This can also be achieved by using the distinct label indicator [VV13]:

$$\Delta(\mathbf{X}) = \delta_{|\mathbf{X}|}(|\mathscr{L}(\mathbf{X})|), \tag{2.18}$$

with $\mathscr{L}(\mathbf{X})$ being the function that extracts the set of track labels from the multi-object state $\mathbf{X}$. Using the distinct label indicator, the LMB RFS can be represented more compactly by (cf. [Reu14; VV13]):

$$\boldsymbol{\pi}(\mathbf{X}) = \Delta(\mathbf{X}) w(\mathscr{L}(\mathbf{X})) p^{\mathbf{X}}, \tag{2.19}$$

with the weight

$$w(\mathcal{L}) = \prod_{i \in \mathbb{L}} (1 - r^{(i)}) \prod_{\ell \in \mathcal{L}} \frac{1_{\mathbb{L}}(\ell) r^{(\ell)}}{1 - r^{(\ell)}}, \tag{2.20}$$

and by using the multi-object exponential notation (cf. equation (2.1))

$$p^{\mathrm{X}} \triangleq \prod_{(\boldsymbol{x}, \ell) \in \mathbf{X}} p^{(\ell)}(\boldsymbol{x}). \tag{2.21}$$

The LMB RFS is the most important type of RFS within the context of this thesis, as it is extensively used in the RB-LMB-SLAM filter.

Generalized Labeled Multi-Bernoulli Random Finite Set

The Generalized Labeled Multi-Bernoulli (GLMB) RFS is, as its name implies, a generalization of the LMB RFS [VV13]. In the GLMB RFS the tracks are no longer assumed to be independent as it is the case with the LMB RFS. While in the LMB RFS, the weights of all realizations $\mathbf{X}$ depend on the tracks' existence probabilities $r^{(\ell)}$ (cf. equation 2.20), the GLMB RFS enables the usage of arbitrary weights

and cardinality distributions. For a discrete index set $\mathbb{C}$ which allows for multiple realizations of a set of track labels, the GLMB is given by:

$$\boldsymbol{\pi}(\mathbf{X}) = \Delta(\mathbf{X}) \sum_{c \in \mathbb{C}} w^{(c)}(\mathscr{L}(\mathbf{X})) \left[p^{(c)}\right]^{\mathbf{X}}, \tag{2.22}$$

with the weights being normalized and the spatial distributions being valid PDFs. In contrast to the LMB RFS, which can only represent a single hypothesis (or component) c, the GLMB RFS facilitates propagating multiple hypothesis $c \in \mathbb{C}$. Therefore it can consistently account for the data association uncertainty during the Bayes multi-object filter update (cf. [Reu14] and Section 2.3.2).

δ-Generalized Labeled Multi-Bernoulli Random Finite Set

The Delta-Generalized Labeled Multi-Bernoulli (δ-GLMB) RFS is a special case of the GLMB RFS. Given an arbitrary discrete space Ξ and the collection of all finite subsets of the space $\mathbb{L}$, $\mathcal{F}(\mathbb{L})$, the following substitutions are made within the aforementioned GLMB RFS (cf. [Reu14; VV13]):

$$\mathbb{C} = \mathcal{F}(\mathbb{L}) \times \Xi, \tag{2.23}$$

$$w^{(c)}(\mathcal{L}) = w^{(\mathcal{I},\xi)} \delta_{\mathcal{I}}(\mathcal{L}), \tag{2.24}$$

$$p^{(c)} = p^{(\mathcal{I},\xi)}, \tag{2.25}$$

where $\mathcal{I}$ denotes a set of track labels and each ξ is a realization of Ξ. Using these parameters the distribution given by equation (2.22) becomes:

$$\boldsymbol{\pi}(\mathbf{X}) = \Delta(\mathbf{X}) \sum_{(\mathcal{I},\xi) \in \mathbb{C}} w^{(\mathcal{I},\xi)} \delta_{\mathcal{I}}(\mathscr{L}(\mathbf{X})) \left[p^{(\mathcal{I},\xi)}\right]^{\mathbf{X}}. \tag{2.26}$$

Although this equation is still equivalent to (2.22) in the general case, the δ-GLMB RFS representation has the advantage that the computational complexity can be reduced significantly: If Ξ is chosen to represent the history of the associations between track labels and measurements, then the number of spatial distributions is reduced to $|\Xi|$ (in comparison to $|\mathcal{F}(\mathbb{L}) \times \Xi|$) as each association history ξ already encapsulates the track labels [Reu14; VV13].
The δ-GLMB RFS is essential for the LMB filter that is used in Chapter 5, because it requires the δ-GLMB representation of the multi-object state during the update step.

2.3.2 State Estimation with Random Finite Sets

The basis for estimating a multi-object state using RFS is the multi-object Bayes filter, which is described below. Since it is computationally intractable, approximations are usually made to obtain filtering algorithms that only require limited computing power. One such algorithm is introduced below – the PHD filter. Another RFS filtering algorithm which is also essential for this thesis, is the LMB filter. It is described in detail in Section 5.4.

Bayesian Multi-Object Filtering

The Bayes filter is a recursive approach to estimate the state of a system, or more precisely a PDF representing that state, based on a process model and observations of the system. In order to apply Bayesian filtering, the system is assumed to be Markovian, i.e., the probability of the system state at time k only depends on the system state at time step $k-1$ and no earlier states. Recursive Bayesian filtering methods have been widely used for single object tracking algorithms. However, the extension to the multi-object case, which is facilitated by the use of RFS, has only become a major field of research within the last decade.
An RFS naturally models the uncertainty in the number of objects as well as the uncertainty in the states of the individual objects. As with Baysian single object filtering, the predicted density of such an RFS is obtained by the usage of the Chapman-Kolmogorov equation [Mah07]:

$$\pi_+(\mathrm{X}_+) = \int f_+(\mathrm{X}_+|\mathrm{X})\pi(\mathrm{X})\delta\mathrm{X} \tag{2.27}$$

Although, at first sight this equation looks very similar to the well-known equation for single object filtering, the usage of RFSs has severe effects: First of all, the integral in equation 2.27 is a set integral (cf. 2.11) which has great impact on computability, since it not only sums over all possible object states but also over all possible numbers of objects [Mah04]. Further, $f_+(\mathrm{X}_+|\mathrm{X})$ is a multi-object Markov transition density which is more complex than a single-object transition density, since it not only regards the change in the objects' states (e.g., due to the objects' motions) but also has to consider birth and death of objects.
The multi-object transition density that considers all possible object associations θ between the n' tracks at time step $k-1$ and the n tracks at time step k is [Mah07]:

$$f_+(\mathrm{X}_+|\mathrm{X}) = \pi_B(\mathrm{X}_+)\pi_+(\emptyset|\mathrm{X}) \sum_{\theta} \prod_{i:\theta(i)>0} \frac{p_S(\boldsymbol{x}^{(i)}) \cdot f_+(\boldsymbol{x}_+^{(\theta(i))}|\boldsymbol{x}^{(i)})}{(1-p_S(\boldsymbol{x}^{(i)})) \cdot \lambda_B p_B(\boldsymbol{x}_+^{(\theta(i))})}, \tag{2.28}$$

where p_S is the state dependent survival probability and λ_B is the expected number of new objects which are spatially distributed according to $p_B(\boldsymbol{x}_+^{(\theta(i))})$. Furthermore, $f_+(\boldsymbol{x}_+^{(\theta(i))}|\boldsymbol{x}^{(i)})$ denotes the single-object Markov transition density. Aside from that, it is assumed that the objects' motions are independent and that object appearance follows the Poisson distributed birth density π_B (which is statistically independent from existing objects). The respective probabilities that all objects are new born and that none of the objects survive (i.e., $\theta(i) = 0\ \forall i$) are obtained through:

$$\pi_B(\mathrm{X}_+) = e^{-\lambda_B} \prod_{i=1}^{n} \lambda_B p_B(\boldsymbol{x}_+^{(i)}), \tag{2.29}$$

$$\pi_+(\emptyset|\mathrm{X}) = \prod_{i=1}^{n'} (1 - p_S(\boldsymbol{x}^{(i)})). \tag{2.30}$$

Then, the multi-object posterior density using the Bayes filter update is given by:

$$\pi(\mathrm{X}|\mathrm{Z}) = \frac{g(\mathrm{Z}|\mathrm{X}_+)\pi_+(\mathrm{X}_+)}{\int g(\mathrm{Z}|\mathrm{X}_+)\pi_+(\mathrm{X}_+)\delta\mathrm{X}}, \tag{2.31}$$

where the multi-object likelihood $g(\mathrm{Z}|\mathrm{X}_+)$ has to consider the sensor's Field Of View (FOV), the state dependent detection probability p_D and false alarms. Hence, using the single-object spatial likelihood function $g(\boldsymbol{z}_{\theta(i)}|\boldsymbol{x}_+^{(i)})$, the appropriate standard multi-object likelihood function according to [Mah07] is:

$$g(\mathrm{Z}|\mathrm{X}_+) = \pi_C(\mathrm{Z})\pi(\emptyset|\mathrm{X}_+) \sum_{\theta} \prod_{i:\theta(i)>0} \frac{p_D(\boldsymbol{x}_+^{(i)}) \cdot g(\boldsymbol{z}_{\theta(i)}|\boldsymbol{x}_+^{(i)})}{(1 - p_D(\boldsymbol{x}_+^{(i)})) \cdot \lambda_c c(\boldsymbol{z}_{\theta(i)})}, \tag{2.32}$$

where the false alarm process follows a Poisson distribution with an expected number of clutter measurements λ_c that are spatially distributed according to $c(\boldsymbol{z})$. In many cases, the false alarms are assumed to be distributed uniformly in the FOV. The probability that all measurements originate from clutter is

$$\pi_C(\mathrm{Z}) = e^{-\lambda_c} \prod_{\boldsymbol{z}\in\mathrm{Z}} \lambda_c c(\boldsymbol{z}). \tag{2.33}$$

Further, the probability that no object is detected is given by

$$\pi(\emptyset|\mathrm{X}_+) = \prod_{i=1}^{n} (1 - p_D(\boldsymbol{x}_+^{(i)})). \tag{2.34}$$

The multi-object likelihood (2.32) function averages over all association hypotheses $\theta : \{1, ..., n\} \to \{0, 1, ..., n_Z\}$ between the n objects and the n_Z measurements. For

each hypothesis, measurements have to be assigned uniquely to the tracks (i.e., $\theta(i) = \theta(j) > 0 \implies i = j$) and missed detections are taken into account by the association $\theta(i) = 0$.

The Probability Hypothesis Density Filter

As the multi-object Bayes filter is computationally challenging in general, it is usually necessary to resort to principled approximative solutions instead. In 2003, Mahler introduced the Probability Hypothesis Density (PHD) filter [Mah03], which still is one of the most popular approximations of the multi-object Bayes filter.
The filter propagates the first-order statistical moment of the multi-object posterior $\pi(\mathrm{X})$, its PHD. Hence, it somehow represents the multi-object analog to the constant-gain Kalman filter [MVAV11b]. The value of the PHD $v(\boldsymbol{x})$ at any point $\boldsymbol{x}$ gives the number of expected features at that exact point. Thus, the total number of objects in any given region can be calculated by the integral of the PHD over that region S:

$$\widehat{n} = \int_S v(\boldsymbol{x}) d\boldsymbol{x}. \tag{2.35}$$

Since the PHD filter approximates the multi-object Bayes filter, it also follows the predictor-corrector cycle. Given the single-object Markov transition density $f_+(\boldsymbol{x}_+|\boldsymbol{x})$, the predicted PHD is determined by (cf. [Mah03; Reu14]):

$$v_+(\boldsymbol{x}) = \int p_S(\boldsymbol{\xi}) f_+(\boldsymbol{x}|\boldsymbol{\xi}) v(\boldsymbol{\xi}) d\boldsymbol{\xi} + b(\boldsymbol{x}), \tag{2.36}$$

where the birth of new objects is described by $b(\boldsymbol{x})$, the PHD of the birth RFS (again, the integral of $b(\boldsymbol{x})$ over a region S would yield the expected number of new objects that emerging from within that region). Further, the state-dependent survival probability is given by $p_S(\cdot)$.
Under the assumption that the predicted PHD approximately follows a multi-object Poisson distribution [Mah03], the corrector equation using the measurement likelihood function $g(\boldsymbol{z}|\boldsymbol{x})$, is given by:

$$v(\boldsymbol{x}) = (1 - p_D(\boldsymbol{x}))v_+(\boldsymbol{x}) + \sum_{\boldsymbol{z} \in \mathcal{Z}} \frac{p_D(\boldsymbol{x}) g(\boldsymbol{z}|\boldsymbol{x}) v_+(\boldsymbol{x})}{\lambda_c c(\boldsymbol{z}) + \int p_D(\boldsymbol{\xi}) g(\boldsymbol{z}|\boldsymbol{\xi}) v_+(\boldsymbol{\xi}) d\boldsymbol{\xi}}, \tag{2.37}$$

where the probability of detection is $p_D(\boldsymbol{x})$. Further, the false alarm process follows a Poisson distribution, thereby creating an average of λ_c false alarms which are spatially distributed according to $c(\boldsymbol{z})$. As it can be seen from the equations (2.36) and (2.37), this approximation, in contrast to the full multi-object Bayes filter, does

not require any calculations of set integrals and therefore is much more applicable due to the reduced computational complexity. This advantage comes at a price: The PHD filter approximation results in an unstable estimate of the number of objects [EWB05]. Further, the set representation is lost due to this approximation.
In order to actually implement a PHD filter, either sequential Monte Carlo methods [Sid03] or the Gaussian mixture version, proposed by Vo and Ma in 2006 [VM06], can be employed. The latter is used in all PHD filters utilized in this work. The Gaussian Mixture (GM) implementation approximates the PHD by:

$$v(\boldsymbol{x}) = \sum_{i=1}^{J} w^{(i)} \mathcal{N}\left(\boldsymbol{x}; \hat{\boldsymbol{x}}^{(i)}, \underline{\mathrm{P}}^{(i)}\right), \tag{2.38}$$

where the Gaussian components are described by their mean $\hat{\boldsymbol{x}}^{(i)}$ and their respective covariance $\underline{\mathrm{P}}^{(i)}$. Using this GM implementation, the PHD is a powerful tool for many state estimation tasks.

Chapter 3

Technical Preliminaries

This chapter presents all relevant technical aspects that play an important role for the localization algorithm presented in Chapter 4. These aspects range from a description of the used hardware, i.e., the actual vehicle along with its sensors, over the definition of the coordinate systems to software technical issues.

3.1 Test Vehicle and Sensory Setup

This section describes the overall hardware setup. It comprises the vehicle itself, the relevant sensory equipment installed by the manufacturer, the computing hardware, as well as the sensors that have been installed additionally. Further, the synchronization concept for the sensors and the software framework are presented.

3.1.1 Vehicle

The vehicle that was used for the evaluation of the localization algorithm presented in this thesis is a V8-powered Mercedes Benz E 500 station wagon (S 212) built in 2009, see Figure 3.1. Although the vehicle has been modified to meet the requirements of a wide range of research projects at the Institute of Measurement, Control and Microtechnology at Ulm University, it is entirely road legal. Therefore, it has been possible to conduct all the relevant experiments on German public roads under realistic traffic conditions.
The vehicle's default on-board sensory equipment comprises a wide range of different sensors: five radars, two cameras and many more smaller sensors of various kinds. The data of some of these sensors is used within the algorithms presented in this work. These sensors and their respective data, which is provided via Controller Area

Figure 3.1: The highly automated vehicle of the Institute of Measurement, Control and Microtechnology at Ulm University [Ebe12].

Network (CAN) bus, is described in Section 3.1.2. Nevertheless, several additional sensors have been installed to enhance the vehicle's sensing capabilities.
Three types of sensors that are of great importance for the work presented in this thesis have been added to the vehicle: First of all, three laser scanners were integrated into the front bumper as described in Section 3.1.3. Further, three additional cameras were installed, see Section 3.1.4. And finally, three rearward facing radars (3.1.5) were added.
Another important source of measurement data is the Real Time Kinematic (System) (RTK) which has an integrated module that facilitates using Differential Global Positioning System (DGPS) (see 3.1.6). This system was used to generate ground truth localization data.
All the sensor data has to be processed and evaluated. Therefore, three computing units are installed in the vehicle's trunk. The specifications of these computers are given in Section 3.1.7. Further, handling the triggering and synchronization of the sensors is essential. Details on these mechanisms are presented in Section 3.1.8.

3.1.2 Data Provided via CAN Bus

In its original state, the vehicle is already equipped with a wide range of sensors. Most of these sensors provide their data on the vehicle's CAN bus. The computers in the trunk of the vehicle, which were additionally installed, are connected to the vehicle's CAN bus and can hence access the data available on the bus. Within the context of this thesis, especially the data measured by the wheel speed sensors is of great interest. The wheel speed sensors integrated into the Mercedes-Benz E-Class measure the rotations per second of each of the four wheels with a rate of 50 Hz. As these wheel speed sensors provide the crucial input data for safety relevant assistance

systems, e.g., the Anti-lock Blocking System (ABS) and the Electronic Stability Control (ESC), the data of these sensors has to be of high precision. Thus, these sensors can be used very well to estimate the vehicle speed with sufficient accuracy for the purposes of this thesis. A typical, modern active wheel speed sensor is either based on the Anisotropic Magneto Resistive (AMR) effect or on the Hall effect to measure the rotational speed of the wheel [WHW12].
The second important sensor within the context of this work is the vehicle's yaw rate sensor. It measures the angular velocity around the vehicle's vertical axis and provides this data with a rate of 50 Hz on the CAN bus. Again, this sensor provides crucial input data for the safety relevant ESC system, which is why the quality of this data is ensured to be relatively high. Typically, a Micro-Electro-Mechanical System (MEMS) is used for such a gyroscope. If the vehicle rotates around its vertical axis, a Coriolis acceleration acts on an oscillating mass contained in this system. The resulting Coriolis force is measured by capacitive sense fingers that are placed along the mass housing. The yaw rate can then be determined based on these measurements.

3.1.3 Laser Scanners

A LIght Detection And Ranging (LIDAR) sensor, or laser range finder, is a device which is used to measure distances. These sensors typically emit light beams in the infrared spectrum which are reflected by the environment. These reflections are then again detected by the sensor. LIDARs measure the distance to objects using the Time of Flight (ToF) principle: Given the speed of light c and the time span t between sending and receiving the laser pulse, the distance to the measured object is given by $d = \frac{1}{2} \cdot c \cdot t$. Theoretically, it would be possible to also measure the velocity of

Figure 3.2: An Ibeo Lux laser scanner [Lof14]

objects by using the Doppler effect as it is done in radars. However, in reality this is not done as it would require even more precise, and hence more expensive, technology in order to measure the Doppler frequency with sufficient accuracy [WHW12]. Thus, the velocity of objects is determined by simple differentiation using two or more successive measurements.
Fixed beam LIDAR sensors as e.g., the three channel Continental SRL 1, have already been on the mass market for a few years now and are typically used in low cost emergency brake assistance systems. Actively scanning LIDAR sensors have been gaining attention of the OEMs recently due to their wide range of possible applications. However, they come at a significantly higher price. These laser scanners use a rotating mirror to deflect the laser beam, thereby scanning a 2D slice of the environment. Multilayer 2D laser scanners extend that principle by using multiple beams to scan multiple of these 2D planes (typically 4 or 8). Further, 3D laser scanners, as e.g., the Velodyne HDL-64E, have already been successfully used in highly automated research test vehicles [PPO+12; Urm+08]. These sensors produce an entire 3D point cloud of the surroundings. However, they are even more expensive and much harder to seamlessly integrate into a normal car.
The test vehicle is equipped with three 4-layer IBEO Lux laser scanners. These three LIDARs were integrated into the front bumper, to provide a total FOV of approximately 210 deg, see Figure 3.1. Each of the three scanners has a horizontal FOV of 85 deg, which is covered by all four layers. Further this FOV is extended to 110 deg, where the outer angular ranges are only covered by two layers. The maximum detection range is 200 m and a distance independent measurement accuracy of 0.1 m is reached. The distance resolution is 0.04 m, while the horizontal angular resolution is 0.125 deg. The sensors can be operated with an update rate of 50.0, 25.0 or 12.5 Hz. In the test vehicle they are always running at 12.5 Hz. The laser scanners communicate with the vehicle's computers using Ethernet. Further, the laser scanners are time triggered and synchronized.

3.1.4 Cameras

In the Advanced Driver Assistance System (ADAS) context, both – monochrome cameras and color cameras – are very popular sensors. They are already in use for systems such as lane-keeping assist, forward collision warning or traffic sign assist. There are many reasons for their popularity: First of all, they provide huge volumes of data at high frame rates and thereby consume relatively little energy. Thus, they are suitable for a wide range of applications. Further, images can be handled more intuitively for humans than the data of other sensors, which – depending on the problem at hand – may make it easier to develop algorithms using this kind of data. One major drawback comes with the high data rate – this amount of data has to be processed and hence requires much more computing power than other sensors

Figure 3.3: A Baumer TXG14f camera (image based on [Lof14]; identifier was pixelized)

do. Also, (mono) cameras do not provide any depth information and are strongly influenced by the lighting conditions. However, since cameras are already present in modern day vehicles, it makes sense to further broaden their area of application.
The test vehicle is equipped with three cameras, all of them are monochrome Charge-Coupled Device (CCD) cameras [WHW12] produced by Baumer: There is one forward facing Baumer TXG14f, which has a resolution of 1392×1040 px and a horizontal FOV of about 56 deg. It is mounted behind the windshield, next to the rear-view mirror. This camera is the only one used within the context of this work – specifically it is used for localization (see Chapter 4). Although the two rearward facing cameras (another Baumer TXG14f and a Baumer TXG06) which are mounted on the inside of the rear window are not used for this purpose, they could nevertheless be integrated easily if it was necessary. All cameras are synchronized and time triggered capturing exactly 15 frames per second.

3.1.5 Radars

With the introduction of Autonomous Cruise Control (ACC) the radar has become one of the standard sensors for ADAS. In modern cars they are, among others, also used for pedestrian collision warning systems and blind spot assists. The radar has three major advantages over the other sensors for environment perception: First of all, it is not sensitive towards adverse environmental conditions, i.e., it is indifferent with respect to lighting and only heavy rainfall or snow impair the sensor performance. Secondly, it is able to measure the distance and radial velocity of objects, by using the Doppler effect. Finally, radars offer a larger detection range than other sensors (more than 200 m).
As implied by its name, which originally is an acronym “RAdio Detection And

Ranging", radar sensors send out electro-magnetic waves in the radio spectrum and detect the respective echoes. For some types of radars, the distance towards objects can be determined by using the ToF principle. For others, as the Frequency-Modulated Continuous-Wave (FMCW) radars [WHW12], the distance is determined by measuring the difference in the frequency of the emitted and the received signal. The test vehicle is equipped with multiple radars. Its standard equipment comprises a Continental ARS 310 that is used for ACC and several smaller radars for emergency braking and blind spot assist. Further, three additional rearward facing radars were added (one Bosch LRR 3 and two Bosch MRR rear), to allow for tracking traffic participants behind the vehicle. The two Bosch MRR rear are also used for localization, as described in Chapter 4. They are mounted in the two corners of the rear bumper, rotated ± 45 deg with respect to the longitudinal axis. They offer a wide FOV of 150 deg for a range of up to 80 m.
All the radars are of the FMCW type and operate in the $76 - 77$ GHz frequency band which is the standard for automotive radar applications in almost all countries worldwide. They provide their data with a frequency of 15 Hz only via CAN bus, except for the ARS 310, which is also connected via Universal Serial Bus (USB) to the computers.

3.1.6 Reference System – ADMA

In order to facilitate a sensible analysis of the localization algorithm presented in this thesis, a reference system is inevitable. The test vehicle is equipped with such a system: The GeneSys Automotive Dynamic Motion Analyzer (ADMA). The ADMA is an Inertial Measurement Unit (IMU) system that measures and tracks all dynamic movements of the vehicle (e.g., accelerations, velocities, position, angular rates, angles) with high precision.
Under ideal conditions, the system is able to provide the global position of the vehicle with an accuracy of 0.02 m. The global position of the vehicle is determined by an integrated GNSS solution, which uses satellites of two major global systems: the Russian GLObalnaja NAwigazionnaja Sputnikowaja Sistema (GLONASS) and the American GPS. Using the satellites of both systems, it is much more likely that data from enough satellites is received. To increase not only the availability of the system but also its precision, the receiver is much more complex than a standard GPS receiver. It does not only evaluate the messages received from the satellites, but it also evaluates the carrier waves. Furthermore, RTK-DGPS is utilized: That is, a network of reference stations, which are operated by the "Satellitenpositionierungsdienst der deutschen Landesvermessung" (SAPOS), is used to significantly alleviate the influence of atmospheric effects. Whenever possible, one of the reference stations transmits correction data to the vehicle using the mobile communications network. This service is obviously not for free and hence it makes sense to use it in a reference

Figure 3.4: A GeneSys ADMA-G with a Novatel GPS receiver [Lof14]

system, but not in an actual localization algorithm. In order to measure rotational motion, the ADMA possesses three fiber optic gyroscopes. Such a fiber optic gyroscope is much more precise than the vehicle's standard gyroscope. Further, three servo accelerometers are used to determine accelerations. With these precise sensors, it is possible to accurately calculate the vehicle's global pose by using dead reckoning. Thus, it is possible to bridge short DGPS drop outs without losing too much accuracy. If, however, a precise global position is available, it can be used by the system to compensate offsets of the other sensors. The ADMA is connected to the computers via CAN bus and operates at 50 Hz.

3.1.7 Computers

The test vehicle is equipped with three additional computers. Two of these are standard computers running the Automotive Data and Time-Triggered Framework (ADTF) application on an Ubuntu 12.04 64-bit operating system. The first one, which disposes of an Intel Core i7-970 3.2 GHz CPU and 12 GB of RAM, handles the incoming sensor data. It processes the raw sensor inputs and compresses this raw information to object representations of all kinds: It runs – beneath others – the vehicle detection algorithm [GLW+13], the feature extraction algorithms for the localization (see Section 4.2.1) and the clustering algorithm for the laser scanner reflections [MMD09]. These object representations are transferred to the second computer via Ethernet, which then executes the high-level tasks based on this preprocessed data, e.g, object tracking [RVVD14], localization (cf. Chapter 4), environmental modeling [NSD14] or trajectory planning. This computer obviously

has to process many tasks in parallel, hence it disposes of two Intel Xenon E5-2640 2.5 GHz Hexa-Core-CPUs and 32 GB RAM.
The third computer is an actual real-time system – a dSPACE MicroAutoBox. It is used for all critical or safety relevant tasks such as controlling the actuators and handling system faults. Further, the timing pulses to trigger the sensors are generated by this MicroAutoBox.

3.1.8 Time Triggering and Synchronization

Timing is a very crucial subject when it comes to sensor fusion. In a dynamic environment, there is no way of fusing measurements if no information about the time at which these measurements were made is available. Furthermore, with fast moving vehicles the timing has to be very precise. This becomes obvious by a simple example: If the vehicle drives on a rural road with a typical speed of 70 km/h and a stationary object in front of the vehicle is measured, a timing error of as little 10 ms leads to a longitudinal position error of 0.194 m (which is in the region of the required localization accuracy).
In order to facilitate sensible communication between the computers, their clocks have to be synchronized. The two high-level computers run a program ("chrony") which maintains the accuracy of the system clocks using Network Time Protocol (NTP) servers. Their time is then communicated to the MicroAutoBox. The MicroAutoBox additionally uses the time of the GPS Pulse Per Second (PPS) signal to then generate an exact time reference and trigger signals for the sensors. The laser scanners and the camera generate new measurements whenever they receive their respective trigger signal. This trigger signal is also sent to the two computers. This is necessary since the measurements are received at the computers' Ethernet ports with a significant delay, which is not constant. Thus, the computers have to associate the measurements with their respective trigger signals. That way, it is ensured that the exact time stamp of each measurement is available to the subsequent processing chain.
Unfortunately, this method can only be applied for triggerable sensors. However, not all of the vehicle's sensors are triggerable (e.g., the radars are not). These sensors provide their measurements via CAN bus or USB with a more or less constant frequency. The only timing information that is available, is the time at which they are received by the computer. The latency between the reception of the data and the actual time at which a measurement was made, had to be evaluated in experiments for each of the radars. This latency then has to be subtracted from the time at which radar measurements are received on the computer. Such an approach is obviously significantly less precise than triggering sensors at exact points in time.

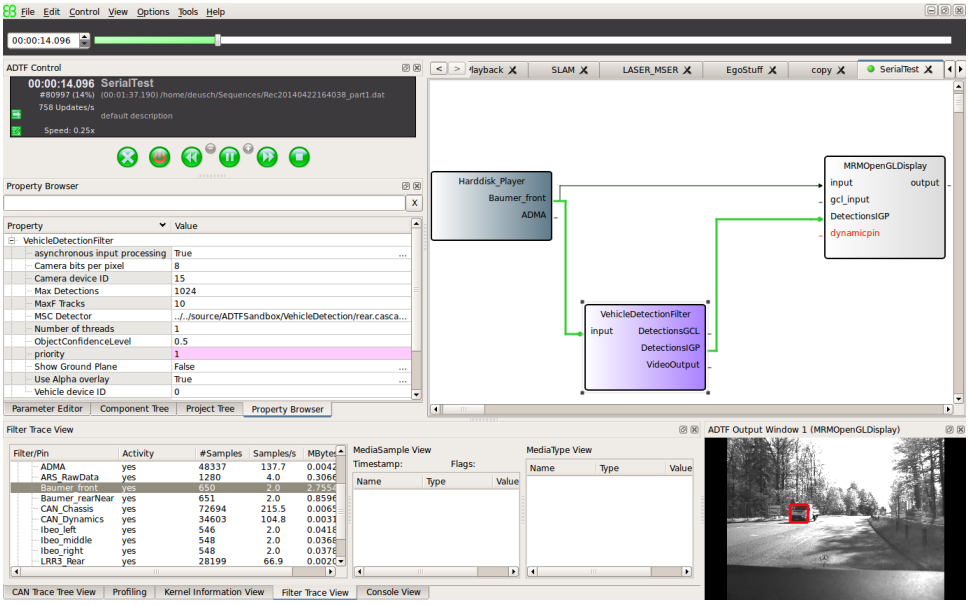


Figure 3.5: Screenshot of the ADTF software framework. The filter graph (top right window) shows an example of a vehicle detection process: The raw image data is fed from a recording to the "Vehicle Detection Filter", whose output is then drawn using an OpenGL Display (bottom right corner).

3.1.9 Software Framework

EB Assist ADTF is a software framework that has been designed to facilitate the development of ADAS. This framework provides many useful tools to simplify the process of creating complex driver assistance systems. Within the framework, a filter graph structure (see Figure 3.5) is used to design and thereby visualize the information flow. Any filter of this graph can be configured by setting the respective parameters in the GUI. The filters' processing methods are executed either whenever new data arrives at an input port, or whenever a certain trigger condition is fulfilled (e.g., after a defined time interval or whenever all input pins have received new data). Each filter may contain arbitrarily complex algorithms which are implemented in C++ or Python. However, for the design process of an entire ADAS function, only the inputs and outputs of the required components are relevant. Thus, for developing the software architecture of an highly automated vehicle, it is possible to reuse multiple filters that implement one functionality each (e.g., a vehicle detector or a tracking algorithm) and connect them all together in one filter graph. This flexibility

allows for a relatively fast development of complex functions, given the required sub-components already exist. Further, ADTF also enables the implementation of services. These services can be accessed by any filter without the need of making a connection in the filter graph. This feature is valuable for tasks that are constantly running in the background and may be related to multiple functions. One example for this concept is a database service, which continuously provides the currently required map data to the environmental model filter but also to the trajectory planning filter and several other components.
Another key feature of ADTF is that it provides a "Harddisk Recorder", which is used to record the data streams that are connected to its input ports. This way, it is possible to record the raw data as provided by the sensors. Then that data can be replayed any time later using the "Harddisk Player" within an arbitrary filter graph. Hence, one recording can be used for testing multiple functionalities. This feature has been used extensively during the development and for the evaluation of the localization algorithm presented in this thesis.
Nevertheless, the real challenge is to process the entire functionality required for an highly automated vehicle online. ADTF also facilitates this. Ideally this can be achieved by simply replacing the "Harddisk Player" of a filter graph by the respective filters that capture the raw sensor data. However, managing such a complex functionality in reality requires a little more effort, due to e.g., timing issues that only arise when real sensors are used.

3.2 Coordinate Systems

Whenever multiple sensors are used, the measurements have to be transformed into a common coordinate frame in order to allow for a sensible fusion. Further, when localizing the vehicle, the relations between the vehicle's coordinate systems and objects defined in a global coordinate system have to be considered as well. The required coordinate systems and transformations are described in the following sections.

3.2.1 Vehicle Coordinate Systems

The definition of the vehicle's main coordinate frame ζ^{veh} is depicted in Figure 3.6. This Cartesian coordinate system's origin is pinned to the ground below the center of the vehicle's rear axle. The x-axis points towards the front of the vehicle, i.e., in the main direction of motion. The positive y-axis is oriented towards the left side of the vehicle and the z-axis points upwards.

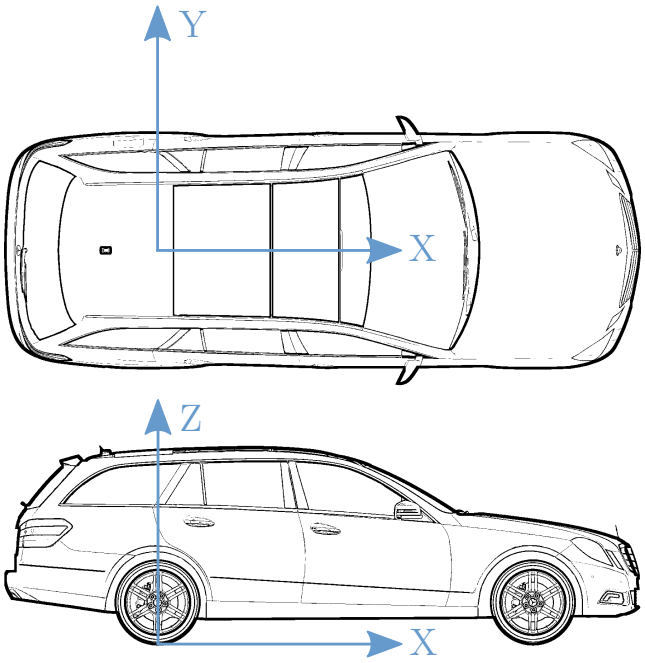


Figure 3.6: Definition of the vehicle coordinate system (depiction based on [Wal13]; axes were modified)

The measurements of all laser scanners and radars are transformed from their respective coordinate systems ζ^{ls_i}, ζ^{r_j} to the vehicle coordinate system ζ^{veh}, before they are being processed any further. These transformations are executed using homogeneous coordinates [BSH+03]. To facilitate the transformation, the exact knowledge of the sensors 3D-mounting poses with respect to the vehicle coordinate system is required. This pose has been ascertained by measuring the position of multiple special objects with the respective sensors. The exact position of these objects with respect to the vehicle coordinate frame had been determined beforehand by using a Leica 3D Disto, i.e., a device specifically designed to precisely measure the position of points in three dimensions [Lei14]. Thus, it was possible to determine the necessary translations and rotations using an adequate optimization algorithm. Since the measurement data produced by cameras is very different from the range- and bearing-measurements of the other two sensors, it is also handled differently. The image that is created by a camera does not convey absolute depth information, hence it is not possible to directly assign a 3D position to any pixel of the image. Thus, it makes sense to first detect objects of interest for which also another assumption about their true 3D position can be safely made. Such an assumption could for example be that they can only be found on the ground plane. With such a valid

assumption, the objects may be transformed from the 2D image coordinate system to the 3D camera coordinate system. This transformation has to respect the intrinsic parameters of the camera (focal length, image sensor format, and principal point) as well as the distortion of the lens. The latter parameter describes a nonlinear property and hence the transformation cannot be described directly by a linear transformation matrix. Thus, typically the image is first undistorted and then the linear camera transformation is used, as e.g., described in [HS97]. Finally, the transformation from camera coordinates ζ^{cam} to the vehicle coordinate system ζ^{veh} is the same as with the other sensors.

3.2.2 Global Coordinate Systems

In the context of localization, global coordinate systems are of central importance. Absolute localization aims at finding the pose of the vehicle within a global coordinate system rather than only determining its pose relative to some arbitrary starting point. This is especially necessary for road vehicles. Thus, a common standard to describe the vehicle's pose on the terrestrial globe is required. The World Geodetic System 1984 (WGS84) standard defines this coordinate system and is hence used as the reference system of GPS. WGS84 describes the earth's shape by a spheroid with an equatorial radius of 6 378 137.0000 m and a radius of 6 356 752.3142 m at the poles. The origin of the spheroid coincides with the center of the earth. Commonly, global positions described in this system are given in geographic coordinates, i.e.,

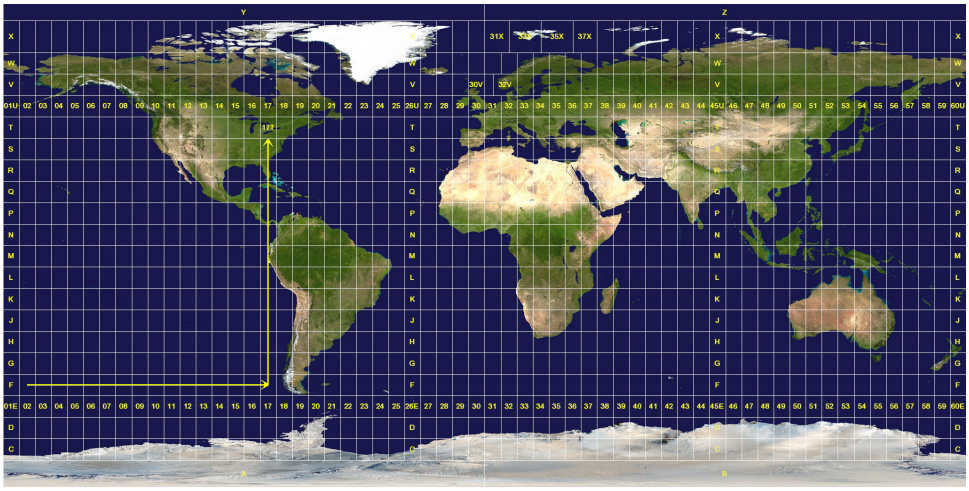


Figure 3.7: Global UTM grid [Kry07]

longitude, latitude and elevation above the reference spheroid. However, in order to facilitate simple calculations when relating globally defined objects (e.g, landmarks for localization) and measurements relative to the vehicle, a Cartesian coordinate system is much more adequate to use. The metric-based Universal Transverse Mercator (UTM)-system is therefore a good choice for this kind of task. This system divides the Earth into longitude bands of six-degree width (60 in total) and utilizes a secant transverse Mercartor projection in each zone (a special kind of cylinder projection). These zones are enumerated from 1 to 60. Further, these zones are separated in north-south-direction in a similar manner: Each zone is divided into 20 eight-degree high lateral bands, each labeled by a letter. Figure 3.7 shows the resulting grid of UTM zones. In order to describe a position on the Earth using the UTM system, a zone number and letter and the easting and northing coordinate pair in that zone are necessary. More details on global coordinate systems and the respective transformations can be found in [Sny87] for example.

3.3 Data Storage Concept

The landmarks that are used for the feature-based localization approach presented in Chapter 4 have to be stored in a sensible manner. That is, the data storage concept has to fulfill several requirements: First of all, the storing system has to provide the requested landmarks to the localization algorithm as fast as possible, in order to facilitate a high update rate of the pose estimate. Thus, it makes sense to use a system that facilitates geospatial queries as this reduces the complexity to find the requested landmarks. A typical query could be: "Fetch all landmarks that are found along the next 500 m of the route the vehicle is currently driving on."
Secondly, the data storage system has to be flexible so that it can handle different types of landmarks found on entirely different routes. Further, it should be relatively easy to update the landmarks in order to be able to quickly react to changes in the environment or route layout. Another requirement is that the system is not limited to storing landmarks, instead it has to be usable by other modules as well, e.g., by the trajectory planning algorithm, which needs a globally defined route to plan the trajectory of the vehicle. Object relational database management systems with geospatial extensions fulfill all these requirements and in addition, they usually provide a wide range of tools that simplify handling of the data.
This section presents the database management system that was selected to handle the landmarks for the localization algorithm proposed in this thesis. Further, the database schema specifically designed for managing these landmarks along with their respective uncertainties is outlined. Finally, also an ADTF service that was developed within the context of this thesis is described. It allows for a seamless integration of the database within the software framework.

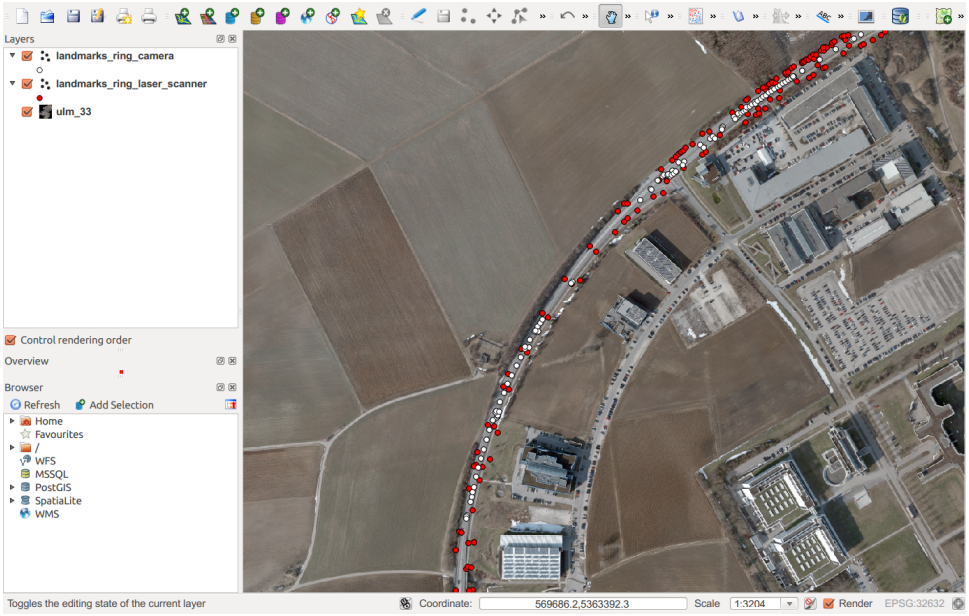


Figure 3.8: Screenshot of QGIS showing a section of the Berliner Ring along with the landmarks on that part of the route (white dots depict landmarks from images and red dots depict landmarks extracted from laser scans)[1]

3.3.1 Geospatial Data in PostgreSQL

The free PostgreSQL Object Relational DataBase Management Systems (ORDBMS) is a very versatile database management system that implements almost all features of the Structured Query Language (SQL) standard. Its origins date back to the early 1980s. It offers standard database functionalities such as storing large amounts of data and retrieving the data which the users demand by queries. Further, it is relative easy to extend since it facilitates the creation of new data types. Therefore, relatively many extensions to PostgreSQL do exist. One very important extension is PostGIS [HO11] which adds support for geospatial data to PostgreSQL. It implements the standard "Simple Features for SQL" defined by the Open Geospatial Consortium (OGC). However, PostGIS implements not only the new data types for the "geometry objects" as defined by the standard (e.g., POINT or POLYGON), but also provides numerous functions to interact with these geometries. These functions include

[1] Depiction based on: "Orthophoto der Stadt Ulm aus Befliegung März 2010" © Stadt Ulm, Abteilung Vermessung [Ulm15]

beneath others: intersecting geometries, determining the area of objects or calculating the distance between geometries. The latter is particularly useful to the localization algorithm: It can be used to query for all the landmarks that are within the FOV of the vehicle's sensors. Further, the execution times for geospatial queries are very low thanks to the possibility to define a geospatial index [HO11]. Furthermore, updating data, i.e., in this case landmarks, is a standard database functionality of PostgreSQL and hence can be done easily. Thus, the combination of PostgreSQL and PostGIS provides everything that is required for a flexible handling of the landmarks for localization. Furthermore, several tools do exist that make the handling of geospatial data with PostGIS easier. One such tool is QGIS which can be used to edit and visualize the geospatial data stored in the database (see Figure 3.8).

3.3.2 Database Schema

Several functionalities of the highly automated vehicle use the database as an important source of information. Hence, this database does not only contain the landmarks for the localization algorithm described in Chapter 4, but also global definitions of lanes and other significant information. This paragraph gives a short overview of the most important aspects of the database schema which was designed to meet the requirements of all the functionalities of the highly automated vehicle interacting with the database.

The relevant tables of the database and their connections are depicted in Figure 3.9. All existing relations between the three major tables (landmarks, lanes, lane markings) are described by a "many-to-many relationship" which requires additional tables to represent the relation. That is, for example a lane may be connected to multiple landmarks (i.e., those landmarks can be seen when driving on that lane), but also a landmark may be connected to multiple lanes (i.e., this landmark can be detected when driving on any of the respective lanes). The same holds for the relations between lanes and lane markings. Further, lanes are interconnected with each other: For example, the vehicle has to change from one lane to another while making a turn at an intersection. This information about the links between lanes, which is modeled by the respective relations table, is very important to the trajectory

ID	geometry	uncertainty	type	data	tags	p_{ex}
0	POINT(..)	POINTZ(..)	IMG_MSER	NULL	{}	0.94
1	POINT(..)	POINTZ(..)	TRAFFICSIGN	NULL	{vmax,70}	0.89
...	...	...	...	...	...	...

Table 3.1: An exemplary landmarks table

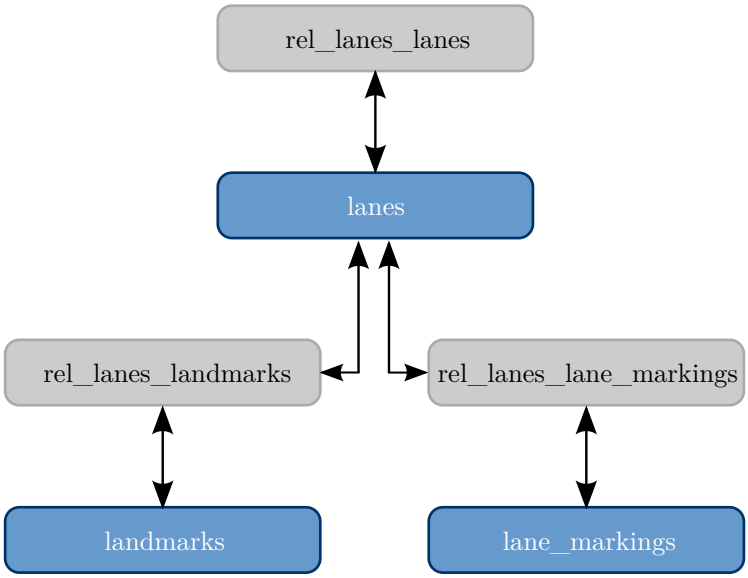


Figure 3.9: The tables of the map database. The three central tables are depicted in blue.

planner of the highly automated vehicle. However, within the context of this work, the storage of landmarks is of more central importance. The layout of the landmarks table is described in Table 3.1. As it is common for data stored in a database, each landmark has a unique identifier, its ID. Further, the global position of any landmark in the UTM coordinate frame is stored in the geometry column. The data is stored as PostGIS geometry type. In most cases it is a POINT but it may also be of any other geometry type, e.g., a POLYGON. A spatial index (cf. [HO11]) is created for this column of the database in order to speed up geospatial queries. The uncertainty about the position of a landmark is described by a 3-tuple which comprises the entries of the uncertainty matrix. This column is also of a geometry data type which simplifies the handling of the landmark's uncertainty during the processing in the ADTF filter. Further, the actual type of each landmark is explicitly stored in another column. The reason for this is that the table contains – besides the landmarks used for localization – several important infrastructural elements whose exact type is of great importance (e.g., traffic lights or roundabouts). The data column and the tags column can be used to provide additional information about a landmark. While the first one basically serves as storage for arbitrary binary data (i.e., complex serialized C++ objects), the latter contains multiple strings that describe the landmarks in more detail. Finally, the p_{ex} column stores the probability of existence of each landmark which could be used to consider its importance.

3.3.3 Integration in the Software Framework

An ADTF service is used to communicate with the PostgreSQL database. This custom-built service was particularly developed to match the needs of the map-based functions of the highly automated vehicle and especially those of the localization algorithm presented in this thesis. The service can be invoked by any filter that requires access to the database. Multiple filters of the ADTF filter graph could possibly request data with a high frequency. Further, some filters may actually require the exact same data. That is why the service implements a spatial caching mechanism which reduces the number of queries to be executed by the database. Using this caching mechanism, all filters may ask the service for new data each time they are triggered (e.g., with a frequency of 50 Hz) while the database is only queried when new data is actually required. This significantly decreases the workload for the database server.
Since all services for ADTF are implemented in C++, interaction with the PostGIS database is facilitated by using the libpqxx C++ library [Ver15]. This library provides a clean interface for executing database queries and thus also for retrieving data from it. Further, the data retrieved from the database can be directly mapped to C++ objects by using the Boost Geometry library [GLL+15]. The reason for this is that both – PostGIS and Boost Geometry – implement the same OGC geometry standard. This greatly simplifies the handling of the geospatial data and facilitates fast processing.
Using the PostgreSQL database in combination with the ADTF service concept, the spatial data can be processed on the vehicle's computers in real-time. Further, feature maps can be created or updated in two ways: either by directly updating the database while driving or by making a recording with ADTF and updating the database later by replaying the recording (or only parts of it).
Finally, a central database server can be used to store the map. Then, all users can work on the same map database, which facilitates parallel development of additional functionalities. This database is always synchronized with the vehicle's database, such that new functions can be tested on the vehicle using the same data as on the workstations.

Chapter 4

Localization for Highly Automated Vehicles

Localization describes the process of determining the position or pose of a vehicle: In most current vehicles, a GPS device is installed to solve this problem. Although a GPS device is able to provide the vehicle's location with sufficient accuracy for todays requirements, this will not be the case for many future driver assistance systems – especially not for highly automated driving. At least within the foreseeable future, highly automated driving will only be possible using precise maps of the environment. Thus, it is not only necessary to reliably determine the vehicle's position on such a map with an accuracy within the decimeter range [LMT07], but to also have precise knowledge of its orientation (accuracy below 1 deg [Sch13]).
This chapter presents an approach to the localization problem, tailored to the requirements of highly automated driving. Albeit other researchers already have published MCL methods for road vehicles, the work presented in the following features novel aspects that may enable the use of highly automated vehicles in more realistic scenarios. First of all, the weighting strategy utilized in the update step is generic and can be used with data from various, entirely different sensors. This is crucial since a highly automated vehicle should not rely solely on one sensor. It has to be able to still function, although with reduced performance, even if one sensor fails – at least to allow for a safe stopping of the vehicle. Further, data of sensors of different suppliers can be fused without having to know too much about the internal processing – basically, even if the sensors only provide object positions they can be integrated. Based on this generic approach, a localization performance is achieved, that truly allows for highly automated driving, using sensors that can be integrated into a road vehicle inconspicuously. Although, this approach would function with various kinds of input data, this work also presents features that have proven to be very helpful to localize a highly automated vehicle on rural roads and in urban scenarios.
This chapter is organized as follows: First, a review of recently published localization

methods is given, before the landmarks used for localizing the vehicle in its "natural environment" (i.e., on roads) are defined. Then, a method to generate adequately accurate maps is described. Thereafter, the localization algorithm is presented. Finally, the developed algorithm is evaluated in different scenarios and a conclusion is drawn.

4.1 State of the Art

Since localization is not only a cutting-edge research topic within the automotive community, but has also been investigated within the robotics community for a significantly longer time, a large amount of publications does exist. In the following, only a small excerpt of these publications is summarized without making a claim to completeness, i.e., only publications being very recent and/or relevant to this work are described.
Most definitely, the development of the so called MCL method by Dellaert et al. [DFBT99] is the fundamental prerequisite for the localization scheme for highly automated vehicles presented in this work. The authors were the first to use a particle filter to tackle the self-localization problem. To be more specific, they described a localization algorithm for an indoor navigating robot equipped with laser range finders, that was able (amongst others) to precisely localize the robot within Smithsonian's National Museum of American History. Their seminal paper lay the ground for many publications on the topic of localization for robots or road vehicles to come.
One of these publications within the context of automotive localization is the work of Miller et al. from 2008 [MC08]. The authors propose a MCL algorithm that fuses GPS data with detections of lane markings and a map of these markings. The publication of Mattern et al. [MSW10] goes into a similar direction. The authors use MCL to find a car's position based on GPS, mapped data and coherency images with a strong focus on the image processing. Other recent approaches also utilize camera data and MCL to determine a vehicle position or pose: For example Schindler [Sch13] focuses on the representation of the map contents (e.g., lane markings) as arc splines to allow for fast computation and reduced storage space. Nevertheless, the actual localization is conducted using the MCL method.
In [KND12] localization is carried out differently. A map match between a digital map of the road course and data extracted from three different types of grid maps is calculated. The optimization problem of finding the best match is solved using one out of those two approaches: either a Levenberg-Marquardt [MNT04] algorithm is executed or a particle filter is employed which uses the error function of the map match to calculate the particle weights. The three grid maps used for the match are defined as follows: a so called feature grid map, where each grid cell represents

the likelihood of the corresponding area containing lane markings, a video grid map created by using inverse perspective mapping and an occupancy grid map based on the range bearing measurements from a laser scanner.
A computationally more challenging approach was introduced by Badino et al. in 2011. In [BHK11] the authors propose a visual topometric localization method using a discrete Bayes filter and a great deal of image features (one gigabyte of Speeded Up Robust Features (SURF) for an 8.8 km long route). The image features are stored in a database and are then matched in every time step to the features extracted from the current camera frame. The discrete Bayes filter uses the probability of the feature matches in the update process and thus gives a smoother and more reliable estimation of the vehicle position than individual frame matches would.
The work of Lategahn et al. goes into the same direction [Lat13; LSZS13]. Here, also huge amounts of features are extracted from a rearward facing camera which together with an IMU is used to estimate the vehicle's pose in six degrees of freedom. However, they use an optimization algorithm to find a so called "single shot" estimate. Then, during a second step called "pose adjustment", the past few single shot matches are fused with the measurements from an IMU to create the final pose estimate. Thus, a rather special method of temporal filtering is used.
Another approach, presented by Pink et al. [PMB09; PS10], also relies heavily on image processing. Instead of creating a map from images recorded by a camera mounted in a road vehicle, the authors (automatically) construct their map of lane markings from aerial images. They solely rely on an on-board camera to localize the vehicle on this map. In contrast to standard ego motion estimation procedures, which typically use measurements from wheel rate sensors or an IMU, their system uses visual odometry for this part of the localization. The vehicle pose in this purely visual navigation system is estimated by a Kalman filter structure that combines the information from the map match and the visual odometry.
Schreiber et al. also propose to execute localization based on lane markings [SKF13]. In contrast to Pink, they use a vehicle equipped with a Velodyne laser scanner, stereo cameras and a very precise GNSS to create maps that comprise all types of lane markings as well as curbs. In their online localization system, they use a stereo camera to detect these features, thereby practically extending the use cases of this localization system to roads without lane markings.
An approach that uses the Velodyne laser scanner mounted on the top of the vehicle not only for mapping but also for localization is described in [LT10]. The authors use a grid map containing infrared reflectivity information of the environment and then estimate the vehicle's 2D position within that grid map with a histogram filter. The histogram filter avoids some issues that might arise when using a particle filter (e.g., it cannot happen that the correct position is not found just because that particular area of the state space is not perfectly covered with particles), however this comes at a cost – in this case it means that only the position is estimated but not the orientation. Nevertheless, this approach leads to an accurate (i.e., close to the resolution of the grid map) position estimation.

An earlier approaches using less salient laser scanners in the context of localization for road vehicles, that is – at least regarding the sensory equipment – closer to the approach presented here, is the one presented by Weiss in [Wei11; WKD05]. The author uses a forward facing IBEO ALASCA laser scanner mounted to the vehicle's front bumper to generate an occupancy grid map. Then features extracted from this map are used to find the correct vehicle pose with the help of a special triangle association scheme.
Finally, the survey published by Skog and Händel [SH09] gives an in-depth insight into localization methods up to the year 2009, with a focus on the state-of-the-art methods at that time. That is, GNSS and ego motion estimation based on on-board sensors are described in detail. Nevertheless, they also describe map matching and information fusion methods.

4.2 Landmarks in the Vehicular Environment

The environment in which a normal road vehicle navigates is usually well structured. This is especially true when thinking about highly automated driving: The scenarios in which most drivers would be willing to hand over control to the vehicle are either scenarios which are boring due to their repetitive nature (e.g., the daily way to work and back home), or are scenarios that are exhausting (of capital importance: traffic jams). These scenarios, at least in Germany, typically arise on well structured roads: on the autobahn, inner city roads or major rural roads. In general these roads are paved and hence well separable from the surroundings. Further, most – but not all – of these roads have painted lane markings. Premised on the assumption that the world around the vehicle is flat, lane markings are well detectable and measurable using a (low cost) mono camera, which is already present in many modern vehicles. Hence, lane markings provide valuable information for finding the vehicles position. Also, in contrast to (arbitrary) image features, lane markings convey a meaning, i.e., they describe the area which can be occupied by the vehicle and therefore have to be heeded by any (highly automated) vehicle. Nevertheless, they usually cannot be used as sole source of information, since lane markings are in general not uniquely identifiable but instead their shape is well defined and they occur with a defined frequency. One way to compensate for that ambiguity would be to integrate further camera detectable and uniquely identifiable features in the localization algorithm such as traffic lights or traffic signs. This approach would require more complex image processing techniques and also – at least – a reliable estimation of the height of these objects, or even a stereo camera. Besides, it would not extend the use cases for such a system but rather it would further limit the situations in which the localization system would work as expected.
Complementing the camera-based localization with data from other sensors is a

better way to improve the reliability. One obvious choice here would be to utilize a radar, since many modern cars have such a sensor because of the ACC system. The radar is able to detect arbitrary objects, that reflect radio waves at least to a certain degree (e.g., light posts, traffic signs,...). The relative velocity of these objects can be directly measured as well by using the Doppler effect. Thus, it is relatively easy to find out which objects are stationary and which ones are not. This is definitely beneficial since a localization algorithm relies on the objects in the map being stationary. The major disadvantage of such an approach is the limited aperture angle and resolution of such radar sensors. These radars are built to make the vehicle follow another car in front of it and hence finding landmarks along the road has not been a focus when these systems were developed. This might change in the near future when radars will also be used for tasks that require a wider FOV as for instance pedestrian detection and cross traffic assists.
Another sensor that currently gains attention in the automotive industry is the laser scanner – not the big types mounted to the top of the vehicle as, e.g., built by Velodyne – but the smaller 4- to 8-layer laser scanners, that can be integrated in the vehicle design in a less conspicuous way. Although these sensors are still quite expensive, some OEMs already plan on integrating them in their upcoming vehicle generation. Laser scanners can be used for a wide range of applications due to their high resolution and precise measurements. In general they can be used to detect arbitrary objects as long as these objects do not absorb the laser beams.

4.2.1 Maximally Stable Extremal Regions

The Maximally Stable Extremal Regions algorithm was introduced by Matas et al. in 2002 [MCUP02]. Originally the intended use of this algorithm was to establish correspondences in image pairs for wide-baseline stereo matching. Good correspondences in image pairs share some qualities with good landmarks for localization: First of all, they should be reliably detectable – no matter what the illumination conditions are like. Secondly, they should be detectable from different viewpoints, at least to a certain degree. Further, it would also be desirable if they were independent of the hardware, i.e., of the camera being used. Maximally Stable Extremal Regions (MSER) posses properties that facilitate these qualities [MCUP02]: They are closed under continuous transformations of image coordinates and they are closed under monotonic transformation of image intensities. The MSER algorithm performs well in detecting affine regions (according to [MTS+05] superior to all other evaluated algorithms). Hence, it is also likely to be a good candidate to find robust landmarks for localization. Further, it has already proven to be useful for detecting lane markings [SWE11] and as an intermediate step in landmark extraction [RIW13].
In a publication especially about the localization accuracy of affine region detectors [LY12] (i.e., not localization as in the context of this work, but localization in terms

of how precisely the position of the centroid of a region is found under varying conditions), the authors also claim that the MSER algorithm performs best.
On the basis of these studies and several experiments the MSER algorithm was selected as landmark extraction algorithm for the purpose of localizing a highly automated vehicle. In the following, the algorithm itself is described. Thereafter, it is demonstrated how it can be applied to laser scanner grid maps and gray-scale images.

MSER Definition and Algorithm

The definitions of the MSER and the related entities below follow closely the definitions as described in the original publication by Matas et al. [MCUP02].

- An image $\mathscr{I}$ describes a mapping $\mathscr{I} : \mathcal{D} \subset \mathbb{N}_0^2 \rightarrow \mathcal{S}$ with $\mathcal{S} \in \{0, 1, ..., 255\}$, in case of a gray-scale image. Extremal regions on that image are well defined if an adjacency relation $A \subset \mathcal{D} \times \mathcal{D}$ is defined and if $\mathcal{S}$ is totally ordered (which obviously is the case for the gray-scale image). The adjacency relation may be chosen to describe a 4-connected neighborhood: $p, q \in \mathcal{D}$ are adjacent (pAq) if and only if $\sum_{i=1}^{2} \mid p_i - q_i \mid \leq 1$.

- A region $\mathcal{R}$ is a contiguous subset of $\mathcal{D}$. This implies $\forall p, q \in \mathcal{R}$ there exists a set of elements $a_i \in \mathcal{R}$ connecting p and q: $pAa_1, ..., a_iAa_{i+1}, ..., a_nAq$.

- The (outer) boundary $\partial\mathcal{R}$ of a region $\mathcal{R}$ is given by the set of elements not belonging to the region, but being adjacent to at least one element of $\mathcal{R}$: $\partial\mathcal{R} := \{q \in \mathcal{D} \setminus \mathcal{R} : \exists p \in \mathcal{R} : qAp\}$.

- A region $\mathcal{R}$ is called an extremal region if $\forall p \in \mathcal{R}$ and $\forall q \in \partial\mathcal{R} : \mathscr{I}(p) > \mathscr{I}(q)$ (maximum intensity region) or equivalently $\mathscr{I}(p) < \mathscr{I}(q)$ (minimum intensity region).

- A sequence $\mathcal{R}_0, ..., \mathcal{R}_n$ is called a sequence of nested extremal regions if $\mathcal{R}_i$ is an extremal region $\forall i = 0, ..., n$ and if $\forall i = 0, .., n-1$ $\mathcal{R}_i \subset \mathcal{R}_{i+1}$.

Given these definitions, a maximally stable extremal region $\mathcal{R}_{i^*}$ is found if and only if for a sequence of nested extremal regions $\mathcal{R}_{i-\Delta}, ..., \mathcal{R}_{i+\Delta}$,

$$r(i) = \frac{\mid \mathcal{R}_{i+\Delta} \setminus \mathcal{R}_{i-\Delta} \mid}{\mid \mathcal{R}_i \mid} \tag{4.1}$$

has a local minimum at i^*. Here, $|\cdot|$ describes the cardinality of the set and $\Delta \in \mathcal{S}$ is the so called stability parameter.
A more intuitively accessible explanation, as similarly also provided in [MCUP02], goes as follows: If one created binary images $\mathscr{I}_t$ for all thresholds $t \in \mathcal{S}$ of the gray-scale image $\mathscr{I}$. Then $\mathscr{I}_0$ would be completely white, but with increasing t black spots (i.e., local intensity minima) would appear. These spots would form regions and these regions would grow further with increasing t and also new black spots would appear in white areas until the entire image would be black. The set of all connected components of all these images is the set of maximal regions of the original image (the set of minimal regions could be visualized the same way, but by inverting the image beforehand). Of particular interest are those extremal regions that are stable over many binarization thresholds (i.e., those regions that deform the least) – they are the MSER.
The original algorithm to determine these MSER possesses quasi-linear complexity with respect to the number of image pixels and therefore scales relatively well to higher resolution images. As described in [MCUP02], the process is as follows: First of all the pixels in the image are sorted by intensity using bucket sort [CSRL01] which is especially suitable due to the fact that the maximum range of intensities is known exactly a priori and is relatively small (i.e., 256 values). Then, according to this ordering, they are placed in the image. While placing the pixels in the image, a list of connected components along with their respective area is kept using a "union find algorithm" [Sed83]. Hereby, merging two components basically means terminating the smaller component and appending its members to the larger one. This process generates a data structure which contains the area of each connected component as a function of the intensity. In the final step those intensity levels being local minima of the rate of change of the area function are selected as thresholds producing maximally stable extremal regions. These MSER are then represented by the position of the local intensity minimum along with the threshold.
In 2008, Nister et al. published an improved version of the MSER algorithm which reduces the computation complexity of MSERs to true worst-case linear complexity [NS08]. Further, the MSERs were also extended to color images by Forssén in 2007 [For07]. This extension was not applied in the algorithm used in this thesis because of the grayscale camera that was utilized. Nevertheless, it is worth mentioning since it establishes a possible extension of the proposed localization algorithm to be investigated in the future.

MSER Features and Laser Scanner Grid Maps

This section describes the extraction of MSER features from laser scanner data. First, the concept of generating occupancy grid maps using laser scanners is described, then the application of the MSER algorithm on these grid maps is elucidated.

Figure 4.1: Generation of the input for the MSER feature detector: The detections of the laser scanner (left) are used to update the occupancy grid map (middle). Finally, only grid cells that are likely to be occupied are considered for the generation of a grid map image (right) (cf. depiction in [WDN+14]).

Occupancy Grid Maps with Laser Scanners Originally presented by Elfes in 1989 [Elf89], occupancy grid maps have become a powerful and important tool in environment modeling as well as for robot localization since then [TBF05]. The basic idea is to represent the surroundings of a robot (or any vehicle equipped with adequate sensors) as a – usually two-dimensional – grid, where each cell γ_i of this grid commonly holds at least one value $p(\gamma_i) \in [0, 1]$ which describes the probability of the area of the grid cell being occupied by some sort of object. An occupancy grid map can either be defined globally, where the center of each grid cell represents one specific UTM coordinate, or it can also be defined locally, e.g., relative to the starting pose of the vehicle. Furthermore, as demonstrated in [LT10], other information, such as for example infrared reflectivity, can be represented in a more general grid map in a sensible manner. Further, a more advanced grid map using Dempster Shafer evidence theory is presented in [NWW+13]. In general, generating occupancy grid maps with a sensor that measures range and bearing, as laser scanners do, is not very complex. Basically, the path of each laser beam is followed and the probability of occupancy for all grid cells through which this beam passes is reduced. This is done until either the sensing range is reached or until the beam actually hits an object: The probability of occupancy of the corresponding cell is then increased. Obviously, measurement uncertainties have to be considered. In more detail, an algorithm for generating an occupancy grid map works as follows:

1. Initialization: The grid map is initialized with a fixed number of grid cells n_G such that the vehicle is positioned in a defined grid cell γ_{origin}(e.g., in the center of the central grid cell). The value $p(\gamma_i) = 0.5$ is assigned to all grid cells $\gamma_i, i = 1, .., n_G$ to express, that no knowledge of the real occupancy state of any of the cells is available.

2. Pose Estimation: In order to integrate new measurements into the grid map, the pose of the vehicle has to be estimated (provided the vehicle is moving). Depending on the type of grid map, i.e., whether it is a global or local map, different input data is required to estimate the pose. In the case of a global grid map, obviously a global pose estimate is required, which can be obtained from, e.g., a RTK system with DGPS such as the ADMA. In contrast, to generate a local grid map, only the relative motion is necessary. In order to obtain the relative motion between two sensor measurements, dead reckoning (cf. Section 4.4.2) is applied using data from standard on-board sensors (i.e., the data from the four wheel rate sensors and a yaw rate sensor are fed into an Extended Kalman Filter (EKF) which estimates the speed and turn rate using a constant velocity and constant turn rate model). Then, the new pose estimate is used to modify the grid layout. This means that new grid cells are created and other grid cells are deleted on the borders of the grid so that the relative position of γ_{origin} within the grid stays constant (i.e., the number of grid cells left, right, behind and in front of γ_{origin} stays the same as during initialization) and the vehicle is always located in γ_{origin}.

3. Occupancy Calculation: Each detection of the laser scanner (i.e., a range and bearing measurement) at time step k is transformed into the grid map coordinate system. The measurements of the laser scanner as well as the pose estimation have a spatial uncertainty, thus the transformation of the measurements into the grid has to respect these combined uncertainties [WD08]. This is done by determining the occupancy probabilities of all cells within a certain Mahalanobis Distance (MHD) of the transformed laser point's spatial uncertainty. The absolute value of the occupancy probability of a cell also depends on its MHD. These novel occupancy probabilities $\tilde{p}_k(\gamma_i)$ are later used to update the affected cells of the grid map.

4. Free Space Detection: Not only information about where objects do exist is valuable, but at least as valuable is information about where no objects exist (especially for other functions required for a highly automated vehicle such as trajectory planning). Therefore, the calculation of the free space is also crucial to the occupancy grid map. If a laser beam is not reflected, it is very likely that the path along this beam is free space. Furthermore, if a laser beam is reflected at some point, it is also very likely that the path of the laser beam up unto that point is also free space. A very simple approach would be to set the occupancy probability of all affected cells to $p_{free} = 0$. Here, a more advanced function that also considers the actual range and bearing of the measurement is used to determine the occupancy probability of those grid cells. More details about this function can be found in [KSD10]. Again, these newly determined occupancy probabilities $\tilde{p}_k(\gamma_i)$ are used to update the occupancy probabilities of the affected grid cells stored in the map.

5. Updating the Grid Map: The occupancy values determined through obstacle detections and free space calculations are used to update the grid map using the binary Bayes filter approach: As described in [TBF05] each cell is updated independently assuming a static environment. The new occupancy values in time step k for all updated grid cells of the grid map are given by [WSD07]:

$$p_k(\gamma_i) = \frac{S}{1+S} \quad \text{with}$$
$$S = \frac{\tilde{p}_k(\gamma_i)}{1-\tilde{p}_k(\gamma_i)} \cdot \frac{p_{k-1}(\gamma_i)}{1-p_{k-1}(\gamma_i)}.$$

This updated grid map then represents the occupancy probability of the area surrounding the vehicle and can be used for feature extraction.

MSER Extraction from Occupancy Grid Maps In general, the occupancy grid map can be interpreted as a grayscale image just by interpreting the occupancy probabilities as gray levels, i.e., for an 8 bit grayscale image, the probabilities $p \in [0, 1]$ have to be scaled to correspond to the integers $s \in [0, 255]$ (cf. Figure 4.1). Although relevant features in the grid map are only found in occupied areas of the grid map, the free space calculation is still necessary for two reasons: First, it helps in suppressing false positive measurements. Secondly, it helps to estimate the contours of the occupied areas. However, before the MSER detection algorithm is executed on a grid map, all cells having an occupancy probability $p(\gamma_i) < 0.5$ (i.e., those that are arithmetically more likely to be unoccupied than to be occupied) are set to $p(\gamma_i) = 0.0$ to avoid detecting unreliable features. Furthermore, the image is inverted so that maximum intensity regions correspond to occupied areas. An example of the MSER feature detection algorithm on a grid map image is given in Figure 4.2: As can be seen from this example, detections are only present on isolated, relatively small objects. The reasons for this are for one, the design of the algorithm – it is designed to find closed stable regions – and secondly the optional feature of the algorithm to limit the size of the extracted regions. This proves to be very useful here, since it allows for finding only objects that are usually stationary and can be repeatedly detected such as traffic signs or trunks of trees (the detections in the example are all on tree trunks and poles). Objects that are not as reliable and hence should not be detected, as e.g., parked cars (the two thick L-shaped objects below region R_4) or bushes (the dots left of R_3 and the larger object right of R_8 and R_9), are ignored. Thus, this example demonstrates how the MSER detector is able to find objects in grid maps that can be used for localization.

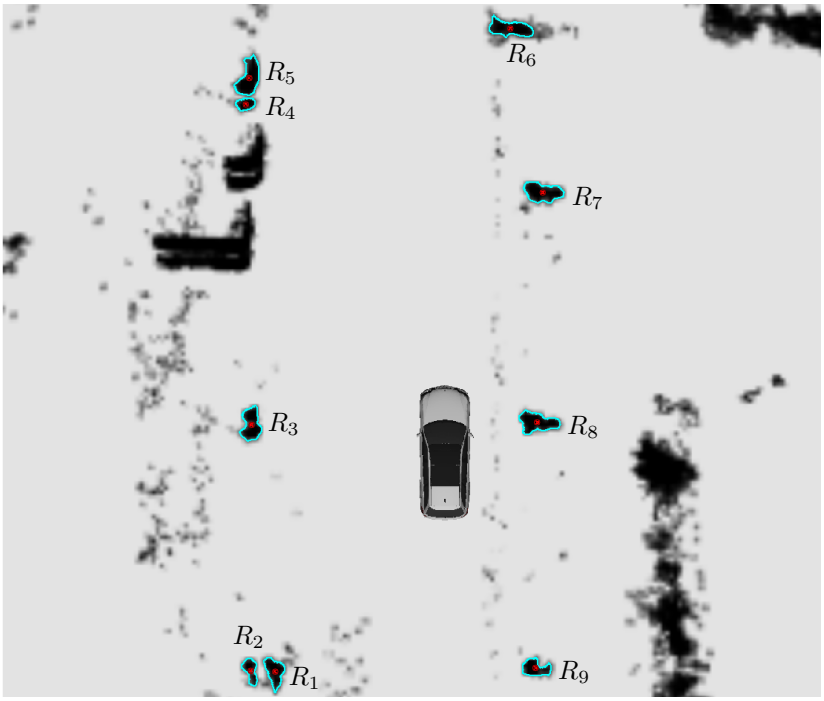


Figure 4.2: An example of MSER detections ($R_1 - R_9$) on a grid map image. The detected MSER' contours are depicted in turquoise and their respective centroids are colored red (depiction based on [DWR+14]; added region labels).

MSER Features from Camera Images

The MSER algorithm can be used on camera images in a much more direct manner than on the laser scanner data, since the algorithm is intended to be used with that kind of data. Nevertheless, one preprocessing step is required before it actually can be used to find sensible landmarks for the purpose of localization. Maximum intensity regions in the vehicular environment are usually found on lane markings of any kind, since they are significantly brighter than the tarmac surrounding them. In order to only detect these features, a Region of Interest (ROI) has to be defined, such that the algorithm is only executed on that part of the image that actually represents the area in which the road is usually found. This ROI is defined assuming a flat world. Only features that are entirely contained in that ROI are used for localization. In order to limit the error that is made due to the flat world assumption, the range in which features are sought has to be restricted. This has a similar effect

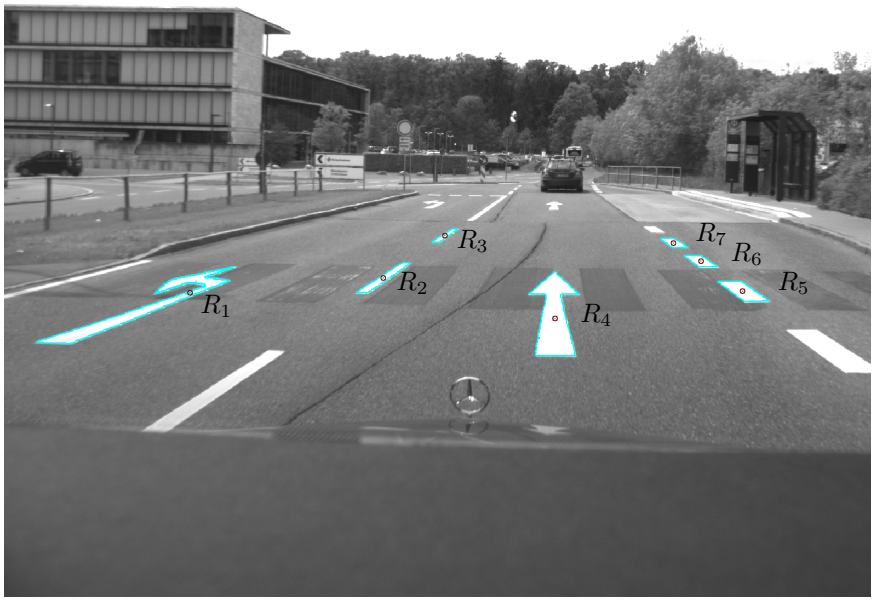


Figure 4.3: An example of MSER detections ($R_1 - R_7$) on a grayscale camera image. The contours of the detected MSER are depicted in turquoise color and their respective centroids are colored red. The lane marking below region R_5 is not detected because it intersects the ROI and only regions that are fully contained in the ROI are extracted.

on the maximum operating range as the resolution of the camera – both limit the maximum distance for reliably detecting features (e.g., the limit for the camera described in Section 3.1.4 is approximately 40 m along the vehicle x-axis). Figure 4.3 shows an example of the MSER detections on an inner city road. It can be seen that the contours of all the different types of lane markings are found perfectly. Thus, features that are very useful for the localization of a vehicle are very well detectable using this approach. Furthermore, even if lane markings are not as new and ideally shaped as those in the example, the algorithm is able to detect them. The reason for that is that it does not make any assumptions about the shape of the landmarks.

Figure 4.4: The top view on the left shows an example of the tracked radar detections. The tracks of the left radar are depicted in red and the tracks of the right radar are colored orange. As can be seen by comparison with the camera image taken from the same scene, tracks only exist on well reflecting stationary objects, for example the bus stop or the traffic lights.[1]

4.2.2 Radar Targets

The features for localization using radar data are based on the raw radar detections provided by the two rearward facing radar sensors of the vehicle (see Section 3.1.5). In contrast to the two MSER-based features presented previously, the features provided by the utilized radars do not require any special feature extraction algorithm. While the radar provides a velocity estimation and therefore a direct method to only select stationary objects, the detections themselves do not provide too much information about the actual objects (at least in comparison with a camera). The most valuable information provided by the detection is the radar cross section which is basically a measure for the reflectivity of the objects. Thus, all detections that originate from stationary objects possessing a reflectivity within a certain range are used as landmark candidates. Typical examples for those kind of objects within the vehicular environment are: traffic lights, lamp posts, traffic signs and other metallic objects.
This restriction obviously limits the radar-based localization to environments that contain many of those objects, which basically implies that it can only be used on urban roads. Nevertheless, this restriction on the reflectivity is necessary in order to solely interpret objects as landmarks that are very likely to be reliably

[1] Depiction based on: "Orthophoto der Stadt Ulm aus Befliegung März 2010" © Stadt Ulm, Abteilung Vermessung [Ulm15]

detectable. However, even these relatively reliable detections cannot be used directly, since the position estimates provided by the raw detections of the objects are not very stable. Hence, a tracking algorithm is used to stabilize and improve these position estimates. A sophisticated filtering algorithm has to be used in order to cope with the false positive rate of the raw radar measurements, the relatively high measurement uncertainty and the non-trivial association between landmark candidates and measurements. Therefore, a PHD filter, as presented in Section 2.3.2 and originally proposed in [Mah03], is used for this multi-object tracking task. As would seem natural, a constant position model is used as state model for the landmark positions to be estimated. These tracked object detections form a valuable input to the localization algorithm.

4.3 Mapping with Known Vehicle Pose

Although this chapter focuses on the localization of a highly automated vehicle, generating a map prior to localizing is inevitable – as long as approaches such as SLAM do not yield the necessary accuracy (cf. Chapter 5). Generating a map that contains the types of landmarks presented before is a relatively easy process. The whole map can be generated by simply driving the route to be mapped once while localizing the vehicle with a precise DGPS. The vehicle that has been used to test and to evaluate the proposed localization algorithm possesses such a system, the ADMA (cf. Section 3.1.6). Thus the ADMA was used to localize the vehicle during map generation.
The exact mapping procedure is depicted in Figure 4.5. The processing of the camera data works as follows: The images are fed to the MSER algorithm as described in the previous section. Then, the detections of each time step k are transformed into the global UTM coordinate frame by first using the transformation from image coordinates to the vehicle coordinate frame (cf. Section 3.2.1) and then transforming these positions to the UTM coordinates using an interpolated pose estimate of the vehicle. The interpolation of the pose estimates is necessary because the camera and the ADMA run on different frequencies. Hence, simply using the pose measurement with the timestamp closest to the image's timestamp would induce a significantly greater error than interpolating between the pose measurements that are made before and after the exact point in time at which the image was shut (which is always possible due to the significantly lower latency of the ADMA). After the MSER detections have been transformed to the global coordinate frame, they are fed into a filter. This simplistic filter basically averages the positions of the landmark candidates over time using a nearest neighbor association between landmark candidates and measurements. As soon as a landmark candidate leaves the camera's FOV it is checked if the landmark candidate has at least been tracked for a number of frames

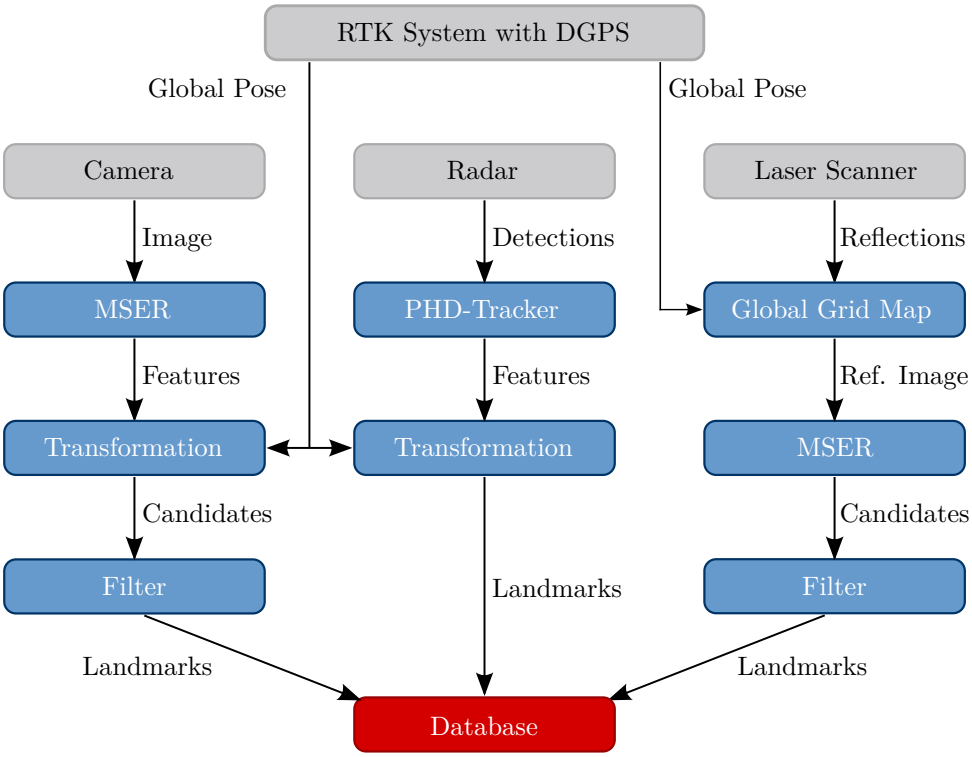


Figure 4.5: The process chain for generating landmark maps

$n_t(v_{veh})$ that depends on the speed of the vehicle. If this is the case, the landmark candidate is inserted into the database. Otherwise the image MSER is dropped.
In comparison to this, the radar measurements are processed quiet differently. In each measurement cycle, all measurements that could possibly originate from a useful landmark, i.e., they posses a negligible velocity and a reasonable reflectivity, are fed into a PHD filter. Then, the output of this filter is analyzed and landmark candidates that have left the radar's FOV and have again been tracked successfully for a certain amount of time are transformed to the UTM coordinate frame. This is done by using the pose measurement from the ADMA. Then these transformed landmarks inserted into the map database.
Finally, the laser scanner data is handled as follows: The measurements are integrated into a globally defined grid map. In order to generate that global grid map, the DGPS measurements are used. Then, the MSER algorithm is executed on an image of the grid map as described in Section 4.2.1. The regions that are found in that process are already globally defined since they are extracted from the globally referenced image of the grid map. Therefore, they can be inserted directly into the database

after a simple filtering step which only ensures that the landmarks candidates have been detected often enough.
The measurement data of the three types of sensors is handled independently and the processing can be executed in parallel very efficiently. Further, the map generation is an entirely unsupervised online process. That means, no interaction with any kind of supervisor is necessary as long as the vehicle's pose is continuously measured precisely by the ADMA. The algorithm inserts the detected landmarks into the map database immediately after they have left the FOV of the vehicle's sensors as long as the ADMA confirms that the vehicle's pose is measured with sufficient accuracy. If, however, the pose cannot be measured with accurately enough, no new landmarks are inserted into the database. In this case, either the respective sections of the route have to be driven once more, or the complete measurement data has to be recorded and a post-processing algorithm, as e.g., presented in [DNK+13], has to be used. Such a post processing algorithm can improve the pose estimations on route sections in which the signal quality of the ADMA is degraded. This is achieved by taking into consideration the measurements of the vehicle's velocity, the measurements of its yaw rate, the last precisely known poses before such a section and the first few precisely known poses after such a section. Based on this data, improved estimates can be calculated for the respective poses. Using these improved pose estimates, the global positions of the detected landmarks can be determined and then they are again ready to be inserted into the database.
Overall, the presented mapping procedure is an uncomplicated approach which nevertheless generates a map that can be relied on during localization (cf. Section 4.5.3).

4.4 Localization Using Particle Filters

The basic concept of particle filtering has already been described in Section 2.2. In this section, the theoretic filtering principle is applied to the problem of localizing a highly automated road vehicle.
While the paragraphs below describe the concrete implementation of the generic particle filtering steps, the overall processing chain from raw sensor input to pose estimate is depicted in Figure 4.6 and elucidated in the following.
First of all, a reliable ego motion estimation is needed to enable good localization performance. Since the raw measurements of wheel rates and yaw rate as provided by the vehicle's standard sensors via CAN bus tend to be noisy, an EKF is used to filter this data [BP99]. Using a Constant Turn Rate and Velocity (CTRV) model, the EKF is able to provide a relatively smooth estimate of the vehicle's velocity and its yaw rate. This ego motion estimate is needed for multiple subsequent calculations: First and foremost it is the control input for the prediction step of the particle filter

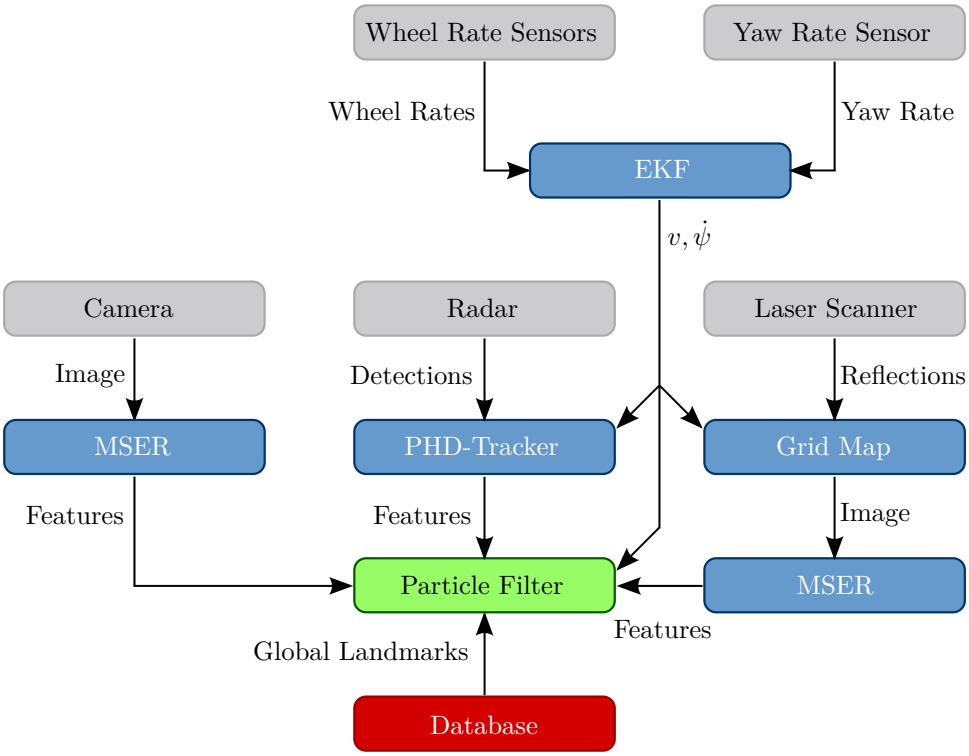


Figure 4.6: The process chain for estimating the vehicle pose

(cf. Section 4.4.2). Additionally, it is also a prerequisite for the processing of two of the feature detectors: The stationary radar targets are tracked by a PHD filter (cf. Section 4.2.2) which obviously needs an ego motion estimate in order to compensate for the vehicle movement. Secondly, the laser scanner data is accumulated in an occupancy grid map (cf. Section 4.2.1) which likewise has to compensate for the ego motion. This grid map is an input to the MSER detector which then provides the detected features as an input to the particle filter's weight update process. Besides these features and the detections of the radar, also the MSER features that are obtained directly from the camera images, are fed into the update process (cf. Section 4.2.1).

Obviously, the filtering algorithm not only needs the current feature detections but also requires a map of landmarks in order to asses the hypotheses about the vehicle's pose represented by the particles. This map is provided by the database component (cf. Section 3.3). Given all these inputs, the particle filter is able to produce a high quality pose estimate of the vehicle by using the functions described in the following.

4.4.1 Initialization

During initialization an initial pose estimate is assigned to each of the ν_P particles. Each pose estimate comprises an UTM easting coordinate, an UTM northing coordinate and an orientation towards the east axis of the UTM cell, i.e., $\boldsymbol{x}_0^{(i)} = [\mathrm{e}_0^{(i)}, \mathrm{n}_0^{(i)}, \psi_0^{(i)}]$ for $i = 1, ..., \nu_P$. The devices that provide these poses are disregarded in Figure 4.6 for reasons of clarity. The actual initial positions $\{\mathrm{e}^{(i)}, \mathrm{n}^{(i)}\}_{i=1,...,\nu_P}$ are drawn from the assumed normal distribution which is given by the position provided by the GPS receiver along with its uncertainty. More precisely, they are drawn from this distribution which however results in latitude-longitude-coordinates of the WGS84 coordinate system. Thus, all these samples have to be transformed to the UTM coordinate frame subsequently. The initial orientations $\{\psi^{(i)}\}_{i=1,...,\nu_P}$ are also drawn from a distribution which is described by a measurement from a compass along with its uncertainty. If no compass is available, the initial orientations may also be obtained by determining the global direction of movement of the vehicle from two or more GPS position measurements.
No matter how the initial poses are obtained, some effort should be put into a valid initialization of the particles, since a bad initialization can have severe effects upon the localization performance, i.e., the particle filter does not converge to the correct pose in reasonable time (see Section 4.5). Thus, if the uncertainty of the GPS is very high, it makes sense to initialize the particle filter with considerably more than ν_P particles and then reduce the number of particles over time.

4.4.2 Prediction

The prediction process determines the state vector of a particle at time step k, $\boldsymbol{x}_k^{(i)} = [\mathrm{e}_k^{(i)}, \mathrm{n}_k^{(i)}, \psi_k^{(i)}]$, given its previous state vector $\boldsymbol{x}_{k-1}^{(i)}$, and the control input $\boldsymbol{u}_k$. Thus, it basically incorporates the current vehicle motion into the last pose hypothesis to form an up-to-date hypothesis about the pose of the vehicle. Probabilistically speaking, the prediction corresponds to sampling from the state transition density $\boldsymbol{x}_k^{(i)} \sim f_+(\boldsymbol{x}_k^{(i)} \mid \boldsymbol{x}_{k-1}^{(i)}, \boldsymbol{u}_k)$. In the case of a car, this state transition is commonly described using a single-track vehicle model or dead reckoning. Although, using dead reckoning is clearly a drastic simplification, it still results in a sufficiently precise approximation of the vehicle's motion in most normal driving situations and it furthermore simplifies calculations significantly.
The actual calculation of the prediction goes as follows: The vehicle's inertial and odometry sensors provide measurements of the four wheel rates as well as a measurement of the yaw rate. An extended Kalman filter then produces a smooth estimate of the vehicle speed v_k and the yaw rate $\dot{\psi}_k$ based on this data. These values describe the motion of the vehicle, assuming constant velocity and turn rate between

time step $k-1$ and time step k. Further, in order to account for the uncertainty of this control input, noise is added to it – for each particle individually – resulting in $\widetilde{v}_k$ and $\widetilde{\dot{\psi}}_k$. The noise is obtained by drawing samples from normal distributions with zero mean and standard deviations that correspond to the expected uncertainties of the respective sensors. According to the dead reckoning formulation, the states are predicted for a time span Δt between the two consecutive time steps $k-1$ and k by:

$$\Delta\psi_k = \widetilde{\dot{\psi}}_k \cdot \Delta t, \tag{4.2}$$

$$\mathrm{e}_k = \mathrm{e}_{k-1} + \frac{\widetilde{v}_k}{\widetilde{\dot{\psi}}_k} \cdot [\sin(\psi_{k-1} + \Delta\psi_k) - \sin(\psi_{k-1})], \tag{4.3}$$

$$\mathrm{n}_k = \mathrm{n}_{k-1} + \frac{\widetilde{v}_k}{\widetilde{\dot{\psi}}_k} \cdot [\cos(\psi_{k-1}) - \cos(\psi_{k-1} + \Delta\psi_k)], \tag{4.4}$$

$$\psi_k = \psi_{k-1} + \Delta\psi_k. \tag{4.5}$$

To avoid the singularity, the model is approximated as follows for yaw rates close to zero:

$$\mathrm{e}_k = \mathrm{e}_{k-1} + \widetilde{v}_k \cdot \Delta t \cdot cos(\psi_{k-1}), \tag{4.6}$$

$$\mathrm{n}_k = \mathrm{n}_{k-1} + \widetilde{v}_k \cdot \Delta t \cdot sin(\psi_{k-1}). \tag{4.7}$$

Given this prediction, noise is added to the predicted states in order to also compensate for the approximations of the simplified model to finally obtain the particle states $\boldsymbol{x}_k^{(i)}$. Again, the noise values are drawn from unbiased normal distributions.[2]

4.4.3 Weight Update

The weight update basically assesses the particle states based on the obtained measurements. Thus, the main objective is to find a sensible implementation of the measurement likelihood function: $g(\mathcal{Z}_k \mid \boldsymbol{x}_k^{(i)})$, i.e., the likelihood of obtaining the set of measurements $\mathcal{Z}_k$ given the state $\boldsymbol{x}_k^{(i)}$.
If one takes a closer look at the concrete case of MCL that means: The goal is to find the likelihood of having measured a set of landmarks given the hypothesis of the vehicle pose that is represented by the particle state. This pose directly translates to a set of landmarks of the a-priori constructed map $\mathcal{M}$ (cf. Section 4.3)

[2] Empirical analysis has shown that assuming a standard deviation – per second – of 0.3 m for easting and northing coordinates, 2 $\frac{\mathrm{m}}{\mathrm{s}}$ for velocity, 1 deg for the orientation angle and respectively 2.3 $\frac{\mathrm{deg}}{\mathrm{s}}$ for the yaw rate is necessary in order to obtain reasonable pose estimates during all evaluated driving maneuvers.

which contains all landmarks that were in the FOV of the vehicle's sensors, if the vehicle's pose actually matched the pose described by the particle's state. This set of landmarks is then transformed from the global UTM coordinate frame $\zeta^{utm}$to the local coordinate frame of the vehicle ζ^{veh}(cf. Section 3.2.2):

$$\mathcal{M} \cap FOV(\boldsymbol{x}_k^{(i)}) \xrightarrow{\zeta^{veh}} \bar{\mathcal{M}}^{FoV}(\boldsymbol{x}_k^{(i)}) \tag{4.8}$$

Given this set of transformed landmarks within the FOV of the respective particle, assessing the particle's weight essentially boils down to finding a multi-object likelihood:

$$\omega^{(i)} \sim g(\mathcal{Z}_k \mid \bar{\mathcal{M}}^{FoV}(\boldsymbol{x}_k^{(i)})) \tag{4.9}$$

When using this multi-object likelihood description, the similarities to the approaches discussed in Section 2.3 become obvious. Basically, the predicted multi-object state X_+ of equation (2.32) is replaced by an expected map state given the vehicle's pose. Since the set of transformed landmarks $\bar{\mathcal{M}}^{FoV}(\boldsymbol{x}_k^{(i)})$ and the set of measurements $\mathcal{Z}_k$ again may very well be explicitly interpreted as RFSs (expressed by using the symbols $\bar{\mathrm{M}}^{FoV}(\boldsymbol{x}_k^{(i)})$ and Z), it makes total sense to use the same approach to find this likelihood:

$$\begin{aligned} &g(\mathrm{Z}_k \mid \bar{\mathrm{M}}^{FoV}(\boldsymbol{x}_k^{(i)})) = \\ &\quad \pi_C(\mathrm{Z}_k)\pi(\emptyset \mid \bar{\mathrm{M}}^{FoV}(\boldsymbol{x}_k^{(i)})) \sum_{\theta} \prod_{j:\theta(j)>0} \frac{p_D(\bar{\boldsymbol{m}}^{(j)}) \cdot g(\boldsymbol{z}_{\theta(j)} \mid \bar{\boldsymbol{m}}^{(j)})}{(1 - p_D(\bar{\boldsymbol{m}}^{(j)})) \cdot \lambda_c c(\boldsymbol{z}_{\theta(j)})}, \end{aligned} \tag{4.10}$$

where the state of the transformed landmark $\bar{\boldsymbol{m}}^{(j)}(\boldsymbol{x}_k^{(i)})$ is abbreviated by $\bar{\boldsymbol{m}}^{(j)}$.
As already described in Section 2.3.2, the probability that all measurements originate from clutter is $\pi_C(\mathrm{Z}_k)$ and the probability that no object is detected is $\pi(\emptyset \mid \bar{\mathrm{M}}^{FoV}(\boldsymbol{x}_k^{(i)}))$. Further, the probability of detecting the transformed landmark $\bar{\boldsymbol{m}}^{(j)}$ is given by $p_D(\bar{\boldsymbol{m}}^{(j)})$ and the expected number of clutter measurements, which are spatially distributed according to $c(\boldsymbol{z}_{\theta(j)})$, is λ_c. Finally, θ describes the association hypotheses.
Since the likelihood (4.10) has to be evaluated for all particles several times in one second (depending on the measurement rate of the sensors e.g., 15 times per second for image features) some approximations have to be made to still allow for real-time performance. First of all, the probability that all measurements are false positives, $\pi_C(\mathrm{Z}_k)$, is dropped since it is identical for all particles and thus is only a scaling factor. Further, the detection probability p_D is assumed to be state independent and the distribution of the clutter measurements is assumed to be uniform within the FOV (denoted as $\mathcal{U}(\mathbb{Z})$). These assumptions make parts of equation (4.10) a constant expression:

$$c_0 = \frac{p_D}{(1 - p_D) \cdot \lambda_c \mathcal{U}(\mathbb{Z})}. \tag{4.11}$$

Finally, the number of association hypotheses θ that are evaluated is limited to the n_θ most likely association hypotheses Θ^*. The actual value of n_θ may even be set to as little as 1, depending on the number of particles and the available computational power.
All associations have to be unique, since each and every landmark generates at most one measurement and all measurements may originate from at most one of the landmarks. Further, more than one landmark may be assigned to the measurement "0", which represents a missed detection. The Munkres algorithm [Mun57] is used to find those n_θ most optimal assignments. In order to find these assignments, the likelihoods for all possible measurement to landmark associations have to be calculated and also the missed detections have to be considered. Since the Munkres algorithm works on costs rather than likelihoods, the costs corresponding to the likelihoods have to be determined. The cost β for assigning landmark (l) to the missed detection (0) is:

$$\beta(\boldsymbol{z}^{(0)} \mid \bar{\boldsymbol{m}}^{(l)}) = -\log(1) = 0. \tag{4.12}$$

The cost for assigning landmark (l) to the measurement (j) is given by (using equation (4.11)):

$$\beta(\boldsymbol{z}^{(j)} \mid \bar{\boldsymbol{m}}^{(l)}) = -\log(c_0 \cdot g(\boldsymbol{z}^{(j)} \mid \bar{\boldsymbol{m}}^{(l)})), \tag{4.13}$$

with $g(\boldsymbol{z}^{(j)} \mid \bar{\boldsymbol{m}}^{(l)})$ describing the likelihood of an assignment between the landmark $\bar{\boldsymbol{m}}^{(l)}$ and the measurement $\boldsymbol{z}^{(j)}$. This likelihood can – at least for all types of landmarks presented above – safely be chosen to only reflect the spatial distance between the landmark and the measurement:

$$g(\boldsymbol{z}^{(j)} \mid \bar{\boldsymbol{m}}^{(l)}) = e^{-0.5 \cdot \left[\left(\frac{d_{lat}^{(j,l)}}{\sigma_{lat}} \right)^2 + \left(\frac{d_{lon}^{(j,l)}}{\sigma_{lon}} \right)^2 \right]}, \tag{4.14}$$

where the lateral and longitudinal distance between landmark and measurement is denoted as d_{lat} and d_{lon} respectively. The standard deviations that are expected when measuring the given kind of landmark are σ_{lat} and σ_{lon}. These standard deviations are obviously different for the lateral and the longitudinal direction, especially when a camera is used.
Although, it was not necessary for the localization tasks presented in the evaluation section, the likelihood may very well be expanded to not only evaluate the spatial likelihood, but to also consider other quantities such as e.g., the sizes of the MSER areas. This can be achieved by simply adding the squared difference between the measured value and the respective value of the landmark divided by the assumed variance to the exponent of equation 4.14, as long as a normal distribution of the values can be assumed.

Given the described assumptions, the total approximated multi-object likelihood is:

$$\tilde{g}(\mathrm{Z}_k \mid \bar{\mathrm{M}}^{FoV}(\boldsymbol{x}_k^{(i)})) = (1-p_D)^{|\bar{\mathrm{M}}^{FoV}(\boldsymbol{x}_k^{(i)})|} \sum_{\theta \in \Theta^*} \prod_{j:\theta(j)>0} c_0 \cdot g(\boldsymbol{z}_{\theta(j)} \mid \bar{\boldsymbol{m}}^{(j)})$$
$$\approx g(\mathrm{Z}_k \mid \bar{\mathrm{M}}^{FoV}(\boldsymbol{x}_k^{(i)})) \tag{4.15}$$

The weight of particle i at timestamp k is then given by this multi-object likelihood multiplied by the previous weight:

$$\omega_k^{(i)} = \omega_{k-1}^{(i)} \cdot \tilde{g}(\mathrm{Z}_k \mid \bar{\mathrm{M}}^{FoV}(\boldsymbol{x}_k^{(i)})). \tag{4.16}$$

As long as out-of-sequence measurements do not occur, this update procedure may be executed sequentially based on different measurements (potentially originating from different sensors) along with their associated landmark maps. If out-of-sequence measurements do only occur rarely, the easiest approach is to simply skip those measurements. If, however, they do occur more often, another option is to apply retrodiction techniques. That would require to store the states of all particles, their weights and the measurements for a certain time span that corresponds to the maximum possible delay. With all this cached data it is possible to easily integrate out-of-sequence measurements and then re-apply the cached updates. If the cost for this is too high, algorithms that alleviate that cost (e.g., [LOC10]) may be considered.

4.4.4 Resampling

The resampling is the most generic part of the particle filter algorithm, since it only operates on the particle weights: The standard low variance sampler method described in Section 2.2 does neither care for the actual state vector nor for how the weights were obtained, as long as each and every particle has a valid weight. Thus, the concrete application does not make any difference to this resampling procedure. Resampling is executed whenever the number of effective particles (cf. listing 1) is below the threshold ν_P^{min}. For the evaluation it was set to $\nu_P^{min} = 0.1 \cdot \nu_P$.

4.5 Evaluation

The proposed localization scheme was evaluated on three different scenarios: on an urban road, on a rural road and on a route that combines both kinds of roads. The latter was also driven by a highly automated vehicle, without any driver intervention using the proposed localization algorithm. The evaluation was conducted using

$\nu_P = 1500$ particles. All results were obtained by running each of the respective recordings of a scenario 50 times in order to account for the non-deterministic behavior of the particle filter. Different recordings were always used for creating the landmark map and for assessing the performance of the localization algorithm.
All scenarios were evaluated with respect to the overall achieved pose accuracy. This accuracy is described by the mean error averaged over the whole route and the respective standard deviation. The ground truth pose was provided by the ADMA (see Section 3.1.6). Further, all runs that exhibited a position error above $t_{err} = 5\,\mathrm{m}$ at any time were marked as "failed" and were excluded from the calculation of the mean error. The reason for this is that in those test runs in which the error exceeded this threshold, the localization algorithm typically was not able to find a pose close to the correct pose again thereafter. Therefore, such a test run would totally bias the mean errors and also severely affect the respective standard deviations. Hence, excluding those test runs from the error calculation – but nevertheless highlighting those failures occurred – actually leads to much more conclusive results.
For the evaluations, measurement updates were only executed if the speed of the vehicle was above a certain threshold. This measure avoids updating the filter with more or less the same measurements over and over again when having stopped at a traffic light. This could especially have undesirable effects when having stopped directly behind another vehicle that occludes most of the sensors' FOV: If then false positive measurements occur in unfavorable positions, this could have negative effects on the filtering performance. If the updates were executed uninterruptedly, these false positive measurements would become overly important and all particles could possibly converge to a very small region of the state space in order to ideally match those measurements. If the true pose was not contained in that region (which is not unlikely), it would take quite some time until the particles would represent the pose of the vehicle in a sensible manner after the vehicle started accelerating again. Further, only the central laser scanner (cf. Section 3.1.3), the forward facing camera (cf. Section 3.1.4), and the two rearward facing mid-range radars (cf. Section 3.1.3) were used.

4.5.1 Urban Road

The proposed localization scheme has been evaluated on a section of a typical inner city road. The major difference of this scenario, in comparison with the other two scenarios described below, is that it facilitates a radar-only localization using the proposed landmarks. The reason for this is that, in contrast to the more rural roads of the other scenarios, the urban road is hemmed with infrastructure elements that reflect the radio waves very well. Hence, it is the only scenario for which the radar based localization is evaluated. More details on the scenario are given in the following paragraph before the reached localization accuracy is presented.

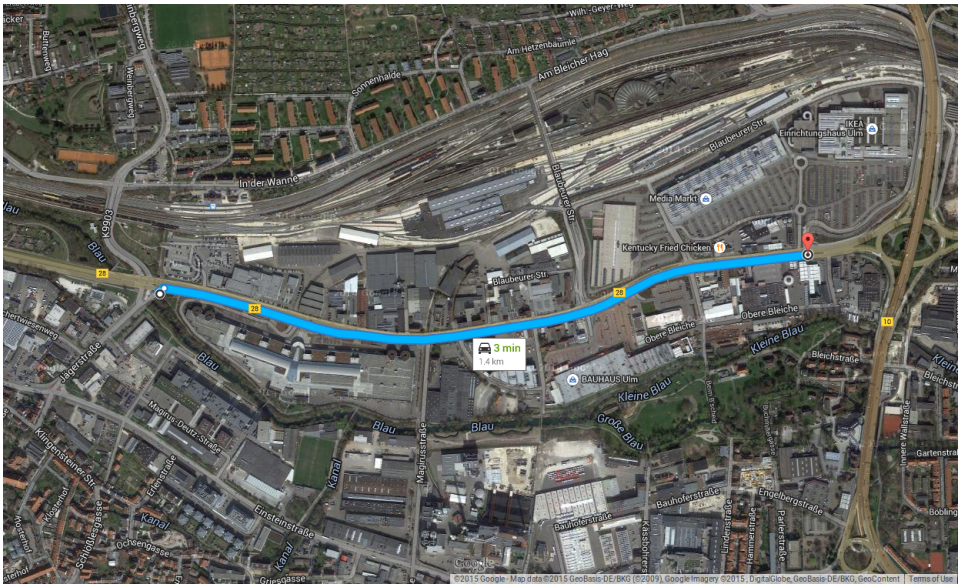


Figure 4.7: This route follows Blaubeurer Str., which is a road with two to three lanes per direction in the city of Ulm. Many infrastructure elements that reflect radio waves very well can be found alongside the road. This fact facilitates radar-only localization.[3]

Scenario Description The route basically follows the Blaubeurer Straße in Ulm for a little less than 1.4 km, as depicted in Figure 4.7. This urban road is one major gateway to the city of Ulm. It consists of multiple lanes in each direction. At the time the recordings were made, the lane markings on this street were relatively new and well detectable. Further, a lot of lamp posts, traffic signs and traffic lights can be found along that road which are, on the one hand, very likely to produce good radar detections and, on the other hand, generate almost ideal MSER in the grid map. Since the route is also relatively straight and contains only two very slight bends, this scenario more or less offers ideal conditions for the proposed localization scheme. There is only one aspect that is not ideal on this route: Other traffic participants which, for one, do occlude landmarks and which also incessantly require the vehicle to decelerate and to accelerate. This is obviously not ideally modeled by the CTRV motion model that is used in the prediction of the particles.
The map that was used for the evaluation contained 531 image MSER landmarks, 401 laser scanner grid map MSER landmarks, 218 radar landmarks for the left radar and 347 landmarks for the right radar.

[3]Source: maps.google.com, ©2015 Google-Map data ©2015 GeoBasis-DE/BKG (©2009), Google Imagery ©2015, DigitalGlobe, GeoBasis-DE/BKG, GeoContent.

Results The evaluation for each of the three types of landmarks was carried out separately. Further, any combination of those landmarks was evaluated. The overall performance for the different configurations is shown in Table 4.1. As can be seen, localization using the camera or the laser scanner based approach (or the combination of both) yields very good performance with standard deviations below 0.12 m and 0.14 deg. Further, also the combination of any of those two sensors with the radar results in a useful performance.
The course of the pose estimation error when using the laser scanner and the camera is depicted in Figure 4.8. It can be seen, that the algorithm estimates the pose very accurately in this admittedly not very challenging scenario. Further, the standard deviation that is obtained from the combined results of the 50 Monte-Carlo runs is also very small indicating that a high localization accuracy can always be expected.
The camera-only approach leads to some inaccuracies in the longitudinal domain. These can be explained by the repeated accelerating and decelerating in this scenario that entails a varying pitch angle of the vehicle. This change in the pitch angle is not compensated for. Consequently, this leads to relatively large errors in estimating the longitudinal position.
When the image MSERs were complemented with radar detections, the effect of the pitch angle vanished. When the radar data had been combined with the grid map MSERs a very good performance was reached as well, although the localization failed once due to a very unfortunate initialization of the particles.
Using the radar alone led to a valid pose estimation in only less than half of the runs. In the successful runs the performance was a lot worse than with the data from the other sensors. However, it has to be stated that the MSER features were exactly tailored to the localization task and to the respective sensor data. In contrast to that, the radar detections were originally designed to find vehicles. If raw radar signals were used instead of these detections, it should be possible to yield much better performance. Nevertheless, even with these simple detections a relatively good localization accuracy can be reached in this scenario.

Sensors	lat_{err} in m	lon_{err} in m	ψ_{err} in deg	*Failed*
Radar	-0.168 ± 0.796	0.122 ± 0.250	0.230 ± 0.662	29
Camera	0.016 ± 0.043	-0.021 ± 0.361	0.065 ± 0.107	0
Laser	-0.066 ± 0.119	-0.053 ± 0.037	0.067 ± 0.124	0
Radar, Camera	0.016 ± 0.050	0.052 ± 0.175	0.067 ± 0.321	0
Radar, Laser	-0.070 ± 0.132	-0.067 ± 0.097	0.066 ± 0.295	1
Camera, Laser	-0.034 ± 0.085	-0.052 ± 0.041	0.022 ± 0.137	0
All	-0.039 ± 0.094	-0.066 ± 0.087	0.033 ± 0.228	0

Table 4.1: Average localization accuracy (i.e., mean and standard deviation of the errors) obtained using three different sensors and all possible combinations of those sensors.

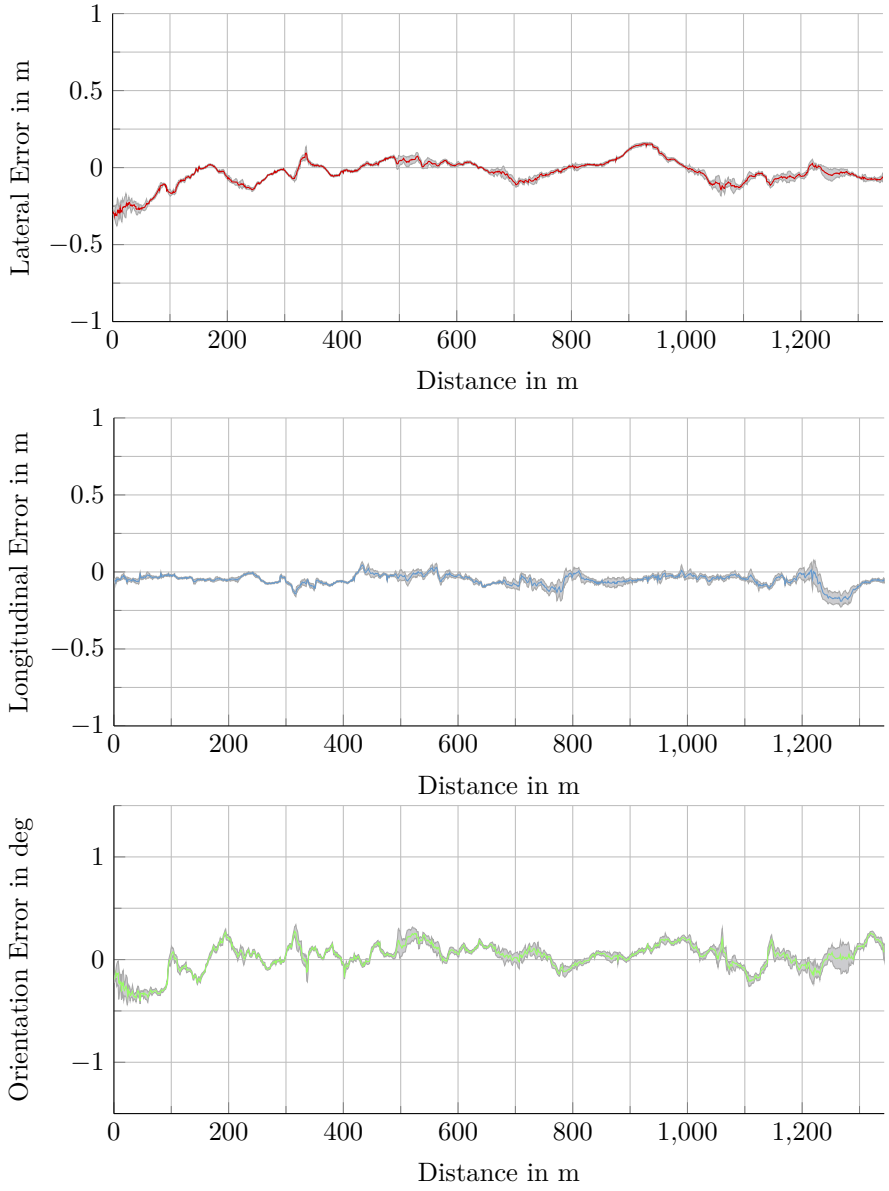


Figure 4.8: Mean pose errors along with the standard deviation (depicted in gray) of the localization algorithm on the Blaubeurer Straße in Ulm using the image MSER and grid map MSER as features.

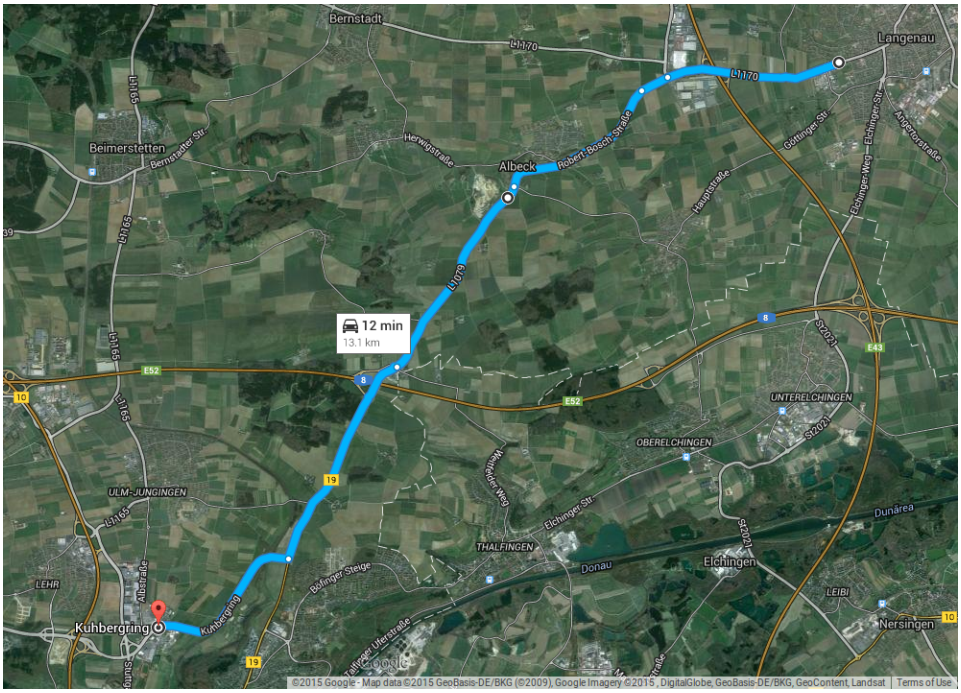


Figure 4.9: The route from Langenau to the Kuhbergring in Ulm for which the accuracy of the localization algorithm was evaluated.[4]

4.5.2 Rural Road

The localization algorithm's performance has also been evaluated on a German rural road. Out of the three evaluated scenarios, this is the most challenging one since it contains several demanding aspects, as described below. Further, this route is much longer than the other two routes.

Scenario Description This route starts at the city limits of Langenau and follows the rural road to Ulm. The allowed maximum speed on that route ranges from 50 km/h to 100 km/h. During the course of that route, several driving situations occur that do not match the assumptions that are made by the measurement model and the motion model: First of all, several roundabouts and traffic lights have to be passed. Obviously, the actual motion of the vehicle is only described insufficiently

[4]Source: maps.google.com, ©2015 Google-Map data ©2015 GeoBasis-DE/BKG (©2009), Google Imagery ©2015, DigitalGlobe, GeoBasis-DE/BKG, GeoContent, Landsat.

by the constant velocity and constant yaw rate assumption, while passing through a roundabout or when stopping at a red traffic light. Furthermore, the route contains multiple steep roads which do not match the flat world assumption that is made in the feature extraction stage. Finally, there is also one segment of the road (approximately from km 8.2 to km 8.6) where almost no landmarks are to be found, since the laser scanner's sight is blocked by a guarding rail and also almost all existing lane markings are continuous, and hence not used as features.
The landmark map created for this route contained 2125 image MSER landmarks and 3486 laser scanner grid map MSER landmarks. This corresponds to as little as one camera landmark every 6.1 m and one laser scanner landmark every 3.7 m, respectively.

Results The localization algorithm was evaluated for two different settings: Using the grid map MSER alone and using both types of MSER features. No evaluation for the camera-only approach was made since the image features were not continuously available along the route. The overall performance is depicted in Table 4.2.
The first noticeable fact is that the localization never failed when the features from the laser scanner grid map were used along with the features from the camera images. In contrast to that, localizing the vehicle was not done successfully in approximately one third of the trials when the grid map MSERs were used alone. The failures occurred always near the roundabout at approximately km 8.6. At that point of the route, almost no features had been available for several hundred meters (starting at km 8.2). In addition, when the vehicle passed through the roundabout its actual motion violated the model assumptions severely which led to an erroneous motion prediction. Due to the missing updates these errors could not be corrected. However, in two thirds of the test runs enough particles still existed close to the real position in order to not exceed the maximum error threshold t_{err}. As soon as new landmarks entered the sensor's FOV after the roundabout had been passed, the pose estimate also almost immediately shifted back to the correct pose. Thus, the overall performance of the laser scanner only approach is still relatively good, when only the successful runs are considered.
Nevertheless, the pose estimation accuracy is significantly higher when the data from both sensors is fused. Especially the standard deviation of the lateral error and the

Sensors	lat_{err} in m	lon_{err} in m	ψ_{err} in deg	*Failed*
Laser	-0.099 ± 0.262	-0.136 ± 0.154	-0.003 ± 0.268	17
Camera, Laser	-0.061 ± 0.118	-0.144 ± 0.132	-0.023 ± 0.174	0

Table 4.2: Average performance of the presented localization algorithm for each of the two sensor setups (i.e., mean and standard deviation of the errors), evaluated on a route of 13 km length.

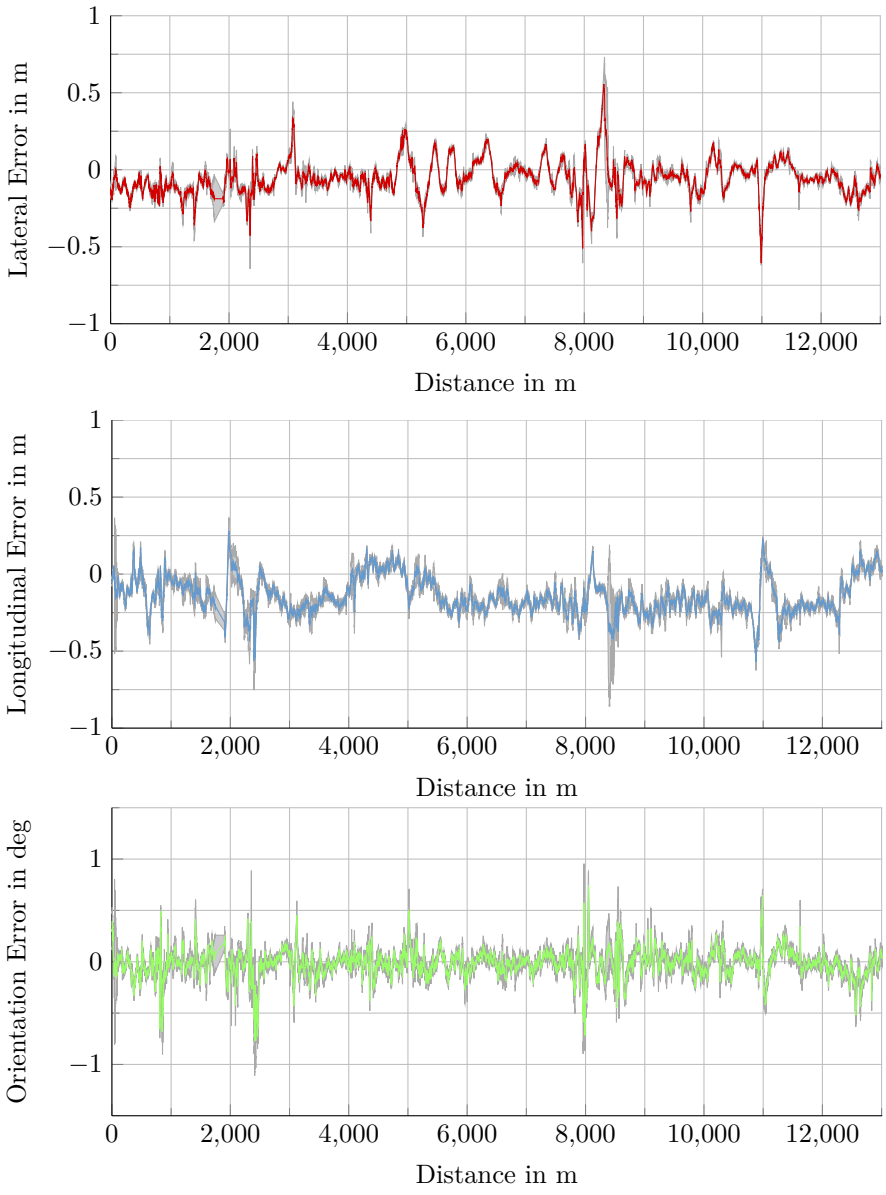


Figure 4.10: Average pose errors of the localization algorithm on the route from Langenau to Kuhbergring, Ulm, using the image MSER and grid map MSER as features. The standard deviation is shaded in gray.

orientation error – which are both much more crucial than the longitudinal error – are significantly smaller. Figure 4.10 reveals that, in this case, the lateral error is almost always below 0.25 m. The few critical situations at which the mean error reaches or exceeds approximately 0.5 m do correspond to the positions of the roundabouts.
One aspect of these results that might seem conspicuous, is the offset of the estimated position, especially in the longitudinal direction. A major part of the offset may be explained by the fact that the local grid map used for localization was not aligned to the global grid map that had been used to create the landmark map. Thus, features extracted from the local grid map may exhibit a shift of several centimeters in longitudinal and lateral direction when compared to the global landmarks (the grid cell size had been set to 0.15 m) which then may result in an offset of the position estimation. Further, the image MSER features are very strong lateral and orientational features. This can also be seen from the significantly smaller standard deviations. However, due to their uncertain longitudinal distance information, they are not very useful for eliminating an offset in the longitudinal direction.
Overall, it can be stated that, when only regarding the demonstrated localization accuracy, a vehicle could possibly drive this route autonomously. The only critical error margins are reached at roundabouts, where the road is usually a bit wider and no oncoming traffic exists.

4.5.3 Autonomous Drive on the Berliner Ring

The localization algorithm was originally designed with the aim in mind to drive a certain public route autonomously without having to rely on DGPS. This route and the respective localization results are presented in the following.

Scenario Description The route starts at the Institute of Measurement, Control and Microtechnology at Ulm University and then follows the Albert-Einstein-Allee westwards. In this section, the Albert-Einstein-Allee does not have any lane markings at all and thus only very few image MSER features can be found (e.g., on gully covers). However, the roadside is hemmed with wooden posts, trees and other cylindrical infrastructure elements. Therefore, a lot of grid map MSER features can be detected. At the end of the Albert-Einstein-Allee the route follows the Berliner Ring northwards. This part of the Berliner Ring is similar to a rural road with a maximum allowed speed of 70 km/h. The lane markings on that road generate reliably detectable MSER features. Further, alongside the road many trees and the compulsory reflector posts make good grid map MSER landmarks. Then, at the intersection which is covered by lane markings, the route continues westwards on the Albert-Einstein-Allee. Parts of this segment of the road do have lane markings. Further, even multiple crosswalks which make very good landmarks, do exist. In

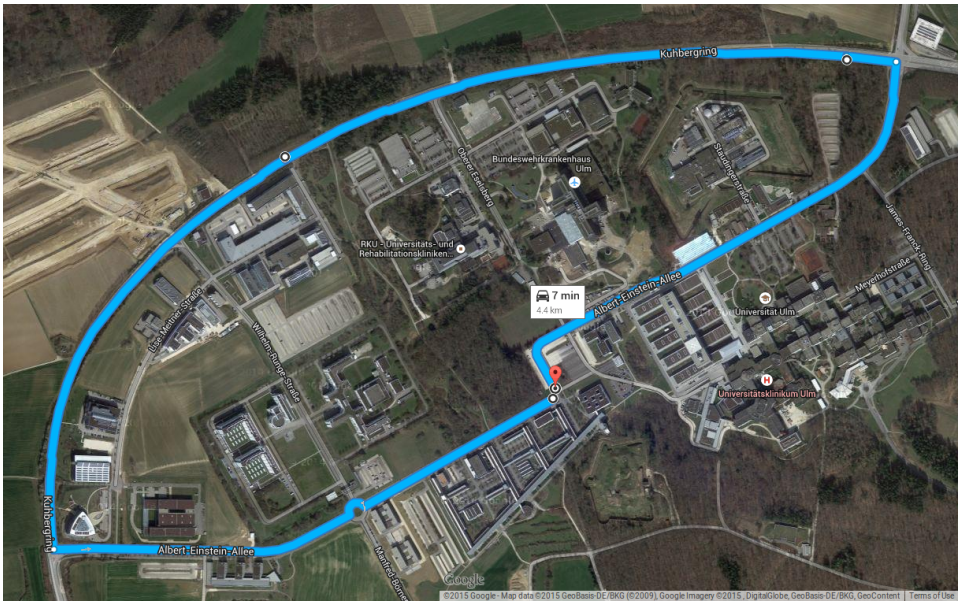


Figure 4.11: The round-trip that has been selected for the first autonomous drive at Ulm University.[5]

addition to that, again a lot of infrastructure (e.g., traffic signs, rocks, lamp posts) are to be found alongside the road – except for a short piece (from km 3.5 to km 3.7). The landmark map of this route was constructed using two different recordings, each offering ideal conditions for the respective sensor. This map contained 685 image MSER landmarks and 1345 laser scanner grid map MSER landmarks in total.
The fact that one longer section of the Albert-Einstein-Allee does not have any lane markings renders an image MSER only localization impossible. Therefore, only the laser scanner-only approach and the fusion of both sensors were evaluated.

Results The performance of the localization algorithm on the route selected for the autonomous drive has been evaluated under different environmental conditions. It obviously had to be ensured that the localization accuracy was always high enough to allow for driving the route autonomously (see Figure 4.12). Localization accuracy may be impaired whenever the mapped landmarks cannot be detected anymore. One reason for not detecting one or more landmarks any longer is for example that the map is outdated (i.e., objects that have been used as landmarks were removed

[5]Source: maps.google.com, ©2015 Google-Map data ©2015 GeoBasis-DE/BKG (©2009), Google Imagery ©2015, DigitalGlobe, GeoBasis-DE/BKG, GeoContent.

(a) Bright sunshine leads to blooming and creates strong shadows. Both may change the MSER. Further, medium height grass blurs the contours in the grid map.

(b) High grass obscures almost all potential grid map MSER landmarks.

(c) Ideal conditions for localizing the vehicle with the proposed approach.

Figure 4.12: Example of the effects of environmental conditions on the input data of the feature extraction algorithms.

or are occluded by other objects that did not exist when the map was created). Another typical reason is that adverse weather conditions lead to reduced sensor performance. The following two environmental conditions were evaluated in order to investigate these phenomena:

- Bright sunshine: The MSER algorithm cannot detect the features precisely in the image due to blooming effects. Further, in the presence of bright sunshine shadows enclose bright MSER-like regions on the tarmac which then may lead to false positive detections.

- High grass: Vegetation may reflect the laser beams, thereby blocking the line of sight between the vehicle's laser scanner and potential landmarks completely. Therefore, many landmarks cannot be detected anymore in the laser scanner grid map.

The recordings used for evaluating the performance on this route were made over a time span of four months. Thus, ample of change in the vegetation (and also some in the infrastructure) alongside the route did happen, as can also be seen in Figure 4.12. As expected, these environmental changes did have effects on the localization accuracy, see Table 4.3.
When comparing the performance under ideal conditions with the performance in the presence of bright sunlight and grass of medium height, one can see that the position estimation does not change very much. However, the standard deviation of the orientation error is significantly lower under ideal conditions. The reason for this is that the image MSER features are not detected very precisely under the bright sunlight, and hence are not very helpful in estimating the orientation exactly. Further, the medium height grass does have a blurring effect on the contours in the grid map which also leads to a much worse orientation estimation.

Sensors (Cond.)	lat_{err} in m	lon_{err} in m	ψ_{err} in deg	*Failed*
Laser (1)	0.076 ± 0.231	-0.095 ± 0.191	-0.135 ± 0.516	0
Camera, Laser (1)	0.049 ± 0.196	-0.111 ± 0.209	-0.159 ± 0.617	0
Laser (2)	0.058 ± 0.412	0.044 ± 0.366	-0.024 ± 0.458	11
Camera, Laser (2)	0.077 ± 0.199	0.013 ± 0.320	-0.064 ± 0.373	0
Laser (3)	0.013 ± 0.234	0.150 ± 0.213	0.093 ± 0.328	0
Camera, Laser (3)	0.010 ± 0.199	0.122 ± 0.169	0.046 ± 0.341	0

Table 4.3: Reached localization accuracy (i.e., mean and standard deviation of the errors) under three different environmental conditions: high grass occludes most gird map features (1), bright sunlight and medium hight grass (2), ideal conditions (3).

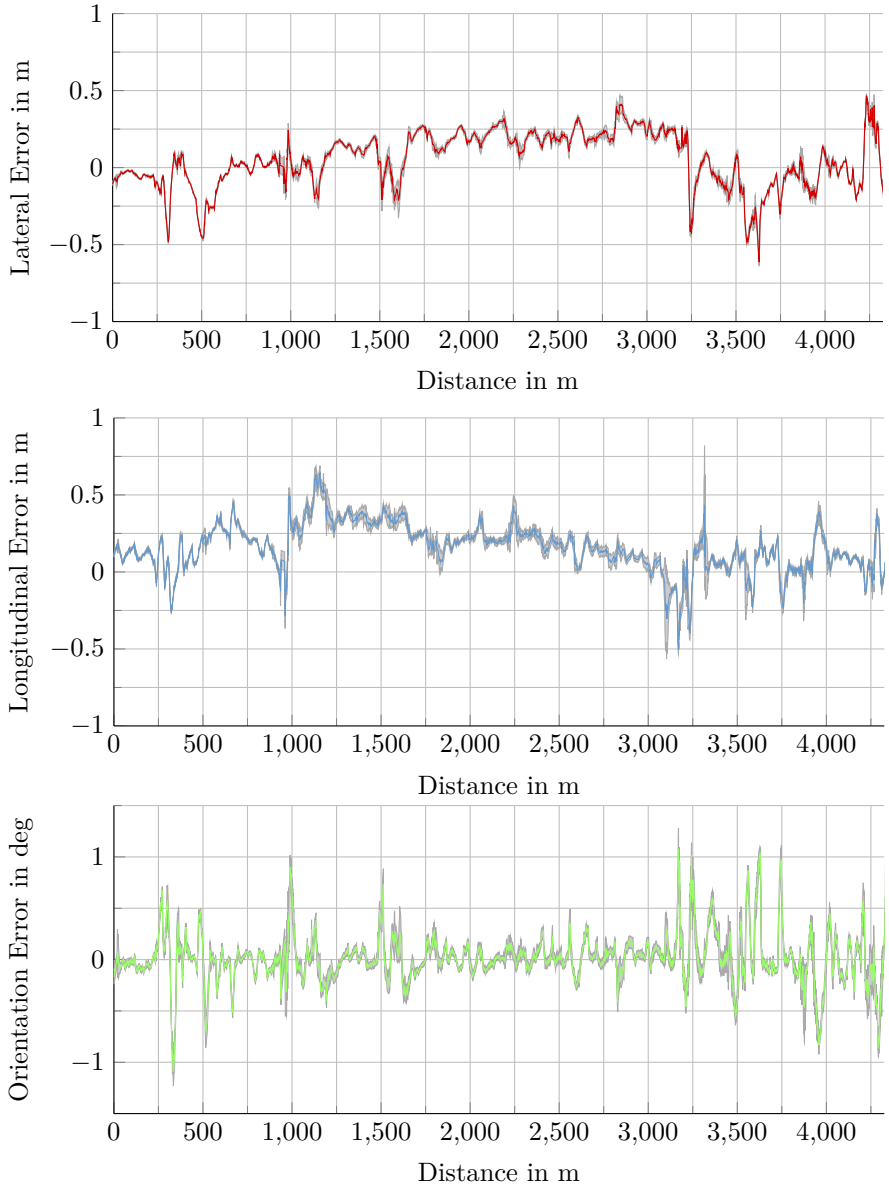


Figure 4.13: Average pose errors of the localization algorithm on the round-trip Albert-Einstein-Allee/Berliner-Ring using the image MSER and grid map MSER as features.

The mean pose error and the respective standard deviations for those evaluations in which the vegetation alongside the road had been really high reveal the expected result: The laser scanner only approach evidently performed much worse than in the other two cases. The standard deviation of the lateral error was above 0.4 m which renders maneuvering the autonomous vehicle safely on its own lane infeasible. Furthermore, in more than one-fifth of the test runs the localization failed entirely, i.e., the position error exceeded $t_{err} = 5\,\text{m}$. Thus, it becomes obvious again that relying on one sensor alone does entail the risk of a total system failure, which obviously has to be avoided when driving autonomously on a publicly accessible road. In combination with the image MSERs, the error margins of all three domains became much smaller and none of the runs failed entirely. In all test cases where the camera and the laser scanner were used in combination, the standard deviation of the lateral error was below 0.2 m and the standard deviation of the longitudinal error was 0.32 m or significantly less.
Figure 4.13 shows the courses of the error of the estimated lateral and longitudinal position along with the course of the error of the estimated orientation exemplary for the ideal conditions test case. The offsets of up to 0.15 m may again predominantly be due to the misalignment of the global grid map used for mapping and the local grid map used for localization. With this overall quality of the position estimation, driving autonomously on this route is possible. Further, given the very good orientation estimation it is possible to assign other traffic participants to their respective lanes. This facilitates a sensible trajectory planning for the autonomous vehicle (e.g., when deciding whether the trajectory should follow another object or if this object should be passed).

4.5.4 Conclusions

The presented MCL algorithm reliably estimates the vehicle's 2D pose using MSER features detected in camera images and MSER features extracted from an occupancy grid map which is created using laser scanner data. The pose estimation error is small enough to allow for autonomous driving on a public route with a highly automated vehicle. This has been proven under different environmental conditions and, more importantly, during several public presentations.
Further, also the localization performance on other routes has been evaluated: It has been proven that, even though the features were designed to match the conditions of the route of the autonomous drive perfectly, the localization accuracy on a normal rural road is not worse – it actually is better. Hence, driving autonomously on a rural road should be possible – at least from a localization perspective.
In addition to this, the presented approach allows for an easy exchange or extension of the features that are utilized, as demonstrated by the evaluation in Section 4.5.1, where additional radar detections are used.

Another great benefit of the presented localization algorithm or, more precisely, the associated mapping scheme, is that it allows for generating new landmark maps on the fly: Simply driving the route and localizing the vehicle with a reference system (ADMA) is enough to obtain a completely new map of landmarks. Further, it is also possible to only replace the landmarks of a certain type, e.g. only the image MSER landmarks. Thanks to this flexibility, a new map of landmarks can be created almost immediately. This quality of the proposed mapping procedure has already proven to be very valuable, when roadworks changed the road layout over several hundred meters entirely. These roadworks had begun just shortly before the first public presentation of the autonomous vehicle was scheduled at Ulm University.
Nevertheless, there is still some room for improvements: First of all, allowing continuous lines to be used as features would certainly boost the performance in those cases where only the data from the camera is used. Further, this would significantly increase the scenarios in which the localization algorithm reaches a satisfactory accuracy. Another aspect that could be improved are the features that are created from the laser scanner data. They are well tailored to the roadsides of rural roads and inner city courses, but cannot be used on highways. The reason for this is that the guarding rails on highways do almost entirely reflect the laser beams and therefore no other objects can be detected. Again, allowing continuous features (i.e., the guarding rails) also in the grid map would be a sensible improvement. Nevertheless, another strong longitudinal feature would be required to facilitate the use on highways.
Finally, a pitch angle estimator would clearly improve the overall performance, since the pitching contradicts the flat world assumption that is made in both feature extraction algorithms.

Chapter 5

Simultaneous Localization and Mapping with RFS

The Simultaneous Localization And Mapping (SLAM) problem has been a popular research topic in the robotics domain for decades, starting in 1986 [BD06; DB06; SC86]. Its ultimate goal is to localize a moving robot precisely on an – a priori entirely unknown – map while consistently generating this particular map. This obviously is a non-trivial task since:

- Localization, i.e., finding the robot's pose by estimating its relative pose to known stationary landmarks, requires a map of these landmarks.

- Mapping, i.e., perceiving new stationary landmarks and storing their (global) position, requires an exact knowledge of the robot's pose.

When taking into account that real sensor data is always non-perfect, i.e., it is subject to noise and drift, this problem constitutes a classical "chicken or the egg" causality dilemma (cf. e.g., [MT07]): The errors in the sensor data used to estimate the ego motion of the vehicle (e.g., odometry data or data from inertial measurement units), lead to an erroneous estimation of the vehicle's pose. Then again, measurements of the landmark locations are distorted by two factors: The measurement noise of the sensor used to perceive the landmark (e.g., cameras, laser scanners or radar) and the corrupted pose estimate. Due to this correlation, as discovered by Smith et al. in 1986 [SC86], the correct path of the vehicle has to be estimated in order to be able to estimate the correct map (i.e., both has to be done simultaneously). Although, in theory, SLAM is a solved problem [DB06], it still remains one of the greatest challenges when it comes to a practical realization.
In recent years, this direction of research has become more relevant to the driver assistance research community. The reason for this is that many currently developed ADAS aim at achieving a higher degree of vehicle automation. Thus, the borders

between highly automated vehicles and robots begin to fade and the problems that have to be solved are very similar: autonomously navigating service robots and highly automated cars have much in common. SLAM is exactly one of these problems that have to be solved in order to build truly autonomous systems.
First of all, this chapter presents the mathematical formulation of the SLAM problem. After that, a short review of the state of the art in vector-based SLAM is given. Then, a more recent approach is described, the so called Random Finite Set (RFS) SLAM. RFS-SLAM has, up until now, usually been implemented using Probability Hypothesis Density (PHD) filters, as e.g., in [AMV13]. Thereafter, a novel Labeled Multi-Bernoulli (LMB) filter [RVVD14] based implementation is introduced and adapted to the problem of doing SLAM. Both of the RFS-based approaches are compared and evaluated. Finally, the integration of a database (see Section 5.6) into the SLAM framework as some sort of "long-term learning map storage facility", is proposed.

5.1 Formulation of the SLAM Problem for a Road Vehicle

The core of the SLAM problem is to infer the trajectory or pose of a vehicle on a map while creating this map. For a mathematically sensible description of the problem, first of all the necessary definitions have to be given. Thereafter, the mathematical formulation of the problem is described.

5.1.1 Definitions

Since a road vehicle usually moves only on the road surface, the 2D pose of the vehicle is a sufficient representation of its location for most applications. Thus, its state $\boldsymbol{x}_k$ at time step k could be described by using a global UTM position and an orientation angle between the vehicle heading and the UTM east axis as in the previous chapter (cf. Chapter 4 and Section 3.2.2). However, this would require an initial pose estimate in the global UTM coordinate frame. In order to keep things a bit clearer a pose in the xy-plane is used instead in this chapter:

$$\boldsymbol{x}_k = \begin{bmatrix} x_k & y_k & \psi_k \end{bmatrix}^T . \tag{5.1}$$

Hence, the pose of the vehicle is always relative to its initial pose $\boldsymbol{x}_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$. Constraining the state space $\mathbb{X}$ to valid poses in the xy-plane is sensible for the purpose of doing SLAM for a road vehicle – nevertheless, the equations presented in

the remainder of this chapter also hold true for entirely different state spaces.
A control input $\boldsymbol{u}_k$ that, in conjunction with a motion model, describes the motion of the vehicle between time step $k-1$ and time step k is required. Again, this control input is given by a velocity in x-direction v_k (i.e., the velocity in direction of the vehicle heading) and a yaw rate (cf. Section 4.4.2):

$$\boldsymbol{u}_k = \begin{bmatrix} v_k & \dot{\psi}_k \end{bmatrix}^T. \tag{5.2}$$

Further, the set of measurements at time step k, $\mathcal{Z}_k$, consists of all the observations $\boldsymbol{z}_k^{(j)}$ with $j = 1, ..., n_{Z_k}$ at time step k ($n_{Z_k} = |\mathcal{Z}_k|$). These observations can either be defined by their respective 2D coordinates in the vehicle coordinate frame

$$\boldsymbol{z}_k^{(j)} = \begin{bmatrix} x_k^{(j)} & y_k^{(j)} \end{bmatrix}^T \tag{5.3}$$

or as range and bearing measurements comprising the distance r and the angle ϕ

$$\boldsymbol{z}_k^{(j)} = \begin{bmatrix} r_k^{(j)} & \phi_k^{(j)} \end{bmatrix}^T. \tag{5.4}$$

For the sake of completeness, sets on the history from time step 0 up to time step k of the respective items are also defined:

- $\boldsymbol{x}_{0:k} = [\boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_k]$ contains all vehicle poses and thus represents the complete trajectory of the vehicle.
- $\boldsymbol{u}_{1:k} = [\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_k]$ comprises all the control inputs.
- $\mathcal{Z}_{0:k} = [\mathcal{Z}_0, \mathcal{Z}_1, ..., \mathcal{Z}_k]$ covers all the observations that have been made.

Finally, the map $\mathcal{M}_k = \{\boldsymbol{m}^1, ..., \boldsymbol{m}^{n_{\mathcal{M}_k}}\}$ is a set comprising all the landmarks $\boldsymbol{m}^{(i)}$, that could have been observed up until time step k (i.e., those landmarks that have passed through the FOV of the sensors). Each of the landmarks is assumed time invariant and is uniquely described by its state, i.e., a position in the xy-plane:

$$\boldsymbol{m}^{(i)} = \begin{bmatrix} x^{(i)} & y^{(i)} \end{bmatrix}^T \in \mathbb{M}, \tag{5.5}$$

where $\mathbb{M}$ is the space of feasible coordinates.

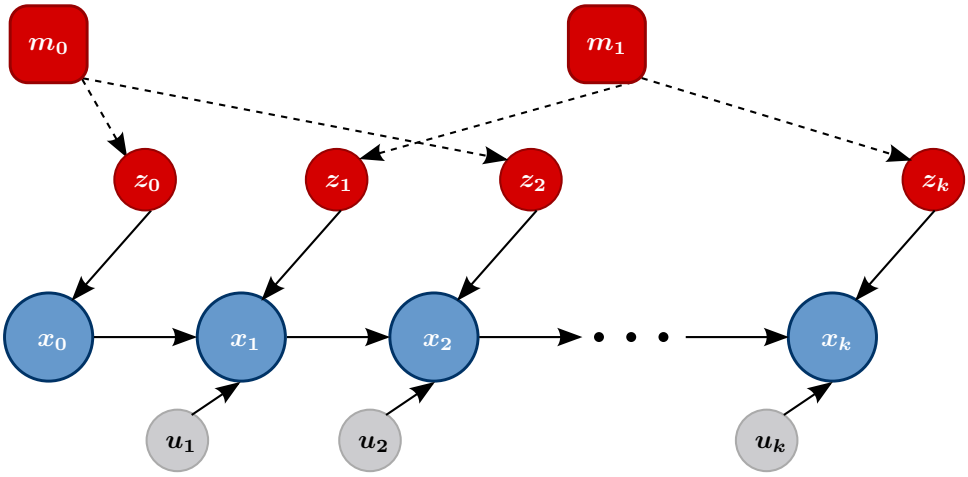


Figure 5.1: In each time step the vehicle is in a state $\boldsymbol{x}$ and makes an observation $\boldsymbol{z}$ of a landmark $\boldsymbol{m}$.[1] The state transition of the vehicle is governed by the control input $\boldsymbol{u}$ (cf. depiction in [MT07]).

5.1.2 Probabilistic Formulation of SLAM

The aim of simultaneous localization and mapping is to optimally estimate the map $\mathcal{M}$ alongside with the vehicle's trajectory $\boldsymbol{x}_{0:k}$ up to time step k (or respectively its pose $\boldsymbol{x}_k$ as in EKF-SLAM). In general, this can be expressed probabilistically (cf. eg., [DB06]) by the probability distribution

$$p(\boldsymbol{x}_{0:k}, \mathcal{M} \mid \mathcal{Z}_{0:k}, \boldsymbol{u}_{1:k}, \boldsymbol{x}_0). \tag{5.6}$$

An abstract SLAM scenario is depicted in Figure 5.1: A vehicle moves along a path (blue circles) and makes observations (red boxes) of landmarks (red circles). Thereby, the vehicle state in each time step, may be calculated by a probabilistic function of the previous time step and the control input (gray circles). The motion model describes exactly that relation between the previous state $\boldsymbol{x}_{k-1}$, the control input $\boldsymbol{u}_k$ and the new state $\boldsymbol{x}_k$. Thus, the transition density for the application of the motion model to the vehicle can be expressed as

$$f_+(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}, \boldsymbol{u}_k). \tag{5.7}$$

[1] For reasons of clarity and comprehensibility only one measurement and one landmark are used in each time step in this description. Obviously, measurements of multiple objects are possible and desirable in each time step, given an appropriate sensor.

If the vehicle used for SLAM is the same as the one used for the localization, the same dead reckoning principle (see Section 4.4.2) may be applied for the prediction of the vehicle state, given its control input.
Further, the measurement model that basically describes how landmarks are perceived by the vehicle's sensor can also be regarded as a probabilistic function. In this case, measuring a feature depends on the current state of the vehicle $\boldsymbol{x}_k$ and the actual landmark map $\mathcal{M}_k$. Therefore, the measurement likelihood function is denoted as

$$g(\mathcal{Z}_k \mid \boldsymbol{x}_k, \mathcal{M}_k). \tag{5.8}$$

Then, one can derive a recursive update rule for the posterior, i.e., a Bayes filter for SLAM where the landmark map is assumed to be static (cf. e.g., [MVAW08]):

$$p_+(\boldsymbol{x}_{0:k}, \mathcal{M}_k \mid \mathcal{Z}_{0:k-1}, \boldsymbol{u}_{1:k}, \boldsymbol{x}_0) = \int f_+(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}, \boldsymbol{u}_k) p(\boldsymbol{x}_{0:k-1}, \mathcal{M}_{k-1} \mid \mathcal{Z}_{0:k-1}, \boldsymbol{u}_{1:k-1}, \boldsymbol{x}_0) d\boldsymbol{x}_{k-1}, \tag{5.9}$$

$$p(\boldsymbol{x}_{0:k}, \mathcal{M}_k \mid \mathcal{Z}_{0:k}, \boldsymbol{u}_{1:k}, \boldsymbol{x}_0) = \frac{g(\mathcal{Z}_k \mid \boldsymbol{x}_k, \mathcal{M}_k) p_+(\boldsymbol{x}_{0:k}, \mathcal{M}_k \mid \mathcal{Z}_{0:k-1}, \boldsymbol{u}_{1:k}, \boldsymbol{x}_0)}{\int\int g(\mathcal{Z}_k \mid \boldsymbol{x}_k, \mathcal{M}_k) p_+(\boldsymbol{x}_{0:k}, \mathcal{M}_k \mid \mathcal{Z}_{0:k-1}, \boldsymbol{u}_{1:k}, \boldsymbol{x}_0) d\boldsymbol{x}_k \delta\mathcal{M}_k}. \tag{5.10}$$

The denominator of equation (5.10) involves integration over the map which in case of a set representation implies calculating a set integral (denoted by $\delta\mathcal{M}_k$). If a vector representation is utilized, it is replaced by a "normal" integral.
However, this Bayes filter is in general computational intractable and can only be solved with the help of several assumptions and approximations [LIA14].

5.2 State of the Art

Since SLAM research has been going on for many years now, a wide range of publications exists. In the following, only a very small excerpt, i.e., some very significant approaches or approaches that are particularly relevant for this work, are summarized and compared. A more thorough overview of SLAM research can be found in e.g., [DB06] and [BD06].

5.2.1 Vector-Based SLAM

The state of the map that is estimated during SLAM has to be represented in a sensible way. The standard way to do this is to stack the states of all landmarks in

some sort of vector. Further, also the measurements have to be somehow pooled, which is again usually done by relying on a vector representation. Most of the approaches aiming to solve the SLAM problem are therefore so-called vector-based approaches. Two of these approaches are outlined in the following: EKF-SLAM and the Factored solution to SLAM (FastSLAM). While the first one is one of the earliest SLAM algorithms and nevertheless still very popular, the latter is a more recent – and thus a more advanced – representative of the SLAM algorithms.
As a consequence of the vector-based representation, the set of measurements at time step k, $\mathcal{Z}_k$, is replaced by the vector $\boldsymbol{\Upsilon}_k$ within this section. The set of landmarks $\mathcal{M}_k$ is replaced by the vector $\boldsymbol{\Omega}_k$, assuming that all landmarks have a probability of existence equal to 1.0.

EKF-SLAM

The idea to use an Extended Kalman filter to tackle the SLAM problem under Gaussian assumptions originates from the pioneering work of Smith and Cheeseman in 1986 [SC86]. Since then many researchers have developed more advanced versions of the EKF-SLAM algorithm (cf. [TBF05]). Nevertheless, all of these algorithms do have the same foundation: An EKF is used to estimate the SLAM posterior (5.6), which is represented by a multivariate Gaussian:

$$p(\boldsymbol{x}_k, \boldsymbol{\Omega}_k \mid \boldsymbol{\Upsilon}_{0:k}, \boldsymbol{u}_{1:k}, \underline{\mathrm{A}}_{0:k}) = \mathcal{N}(\boldsymbol{s}_k; \boldsymbol{\mu}_k, \underline{\Sigma}_k) \tag{5.11}$$

$$\boldsymbol{s}_k = \{\boldsymbol{x}_k, \boldsymbol{m}^1, ..., \boldsymbol{m}^{n_{M_k}}\} \tag{5.12}$$

Note that EKF-SLAM requires an explicit association between landmarks and measurements (denoted as association matrix $\underline{\mathrm{A}}_k$) and that it estimates the vehicle pose $\boldsymbol{x}_k$ instead of its path $\boldsymbol{x}_{0:k}$ (in contrast to the other approaches presented in this chapter).
In EKF-SLAM, the state $\boldsymbol{s}_k$ comprises the pose of the vehicle and the positions of the landmarks. Thus, the mean $\boldsymbol{\mu}_k$ describes the most likely state of the map as well as the state of the vehicle (i.e., all landmark positions and the 2D pose of the vehicle). The covariance matrix $\underline{\Sigma}_k$ holds all correlations between each pair of state variables (i.e., its size is $2n_{M_k} + 3$ by $2n_{M_k} + 3$) which already constitutes one major disadvantage of this algorithm. The quadratic complexity in the number of landmarks makes the evaluation of (5.19) inefficient for large numbers of landmarks.
Since the EKF [SSM62] linearizes the motion model and the measurement model at the most-likely state of the system, it handles the underlying non-linearities in an approximative fashion which works best if the models are only "slightly" non-linear. As e.g., described in [MT07], given the non-linear motion model $f(\boldsymbol{\mu}_{k-1}, \boldsymbol{u}_k)$ and the non-linear measurement model $g(\boldsymbol{s}_k, \underline{\mathrm{A}}_k)$ (which again incorporates the data

association $\underline{\mathrm{A}}_k$), the update equations for the EKF are as follows:

$$\tilde{\boldsymbol{\mu}}_k = f(\boldsymbol{\mu}_{k-1}, \boldsymbol{u}_k), \tag{5.13}$$

$$\tilde{\underline{\Sigma}}_k = \underline{\Sigma}_{k-1} + \underline{\mathrm{Q}}_k, \tag{5.14}$$

$$\underline{\mathrm{G}}_k = \nabla_{\boldsymbol{s}_k} g(\boldsymbol{s}_k, \underline{\mathrm{A}}_k) \mid_{\boldsymbol{s}_k = \tilde{\boldsymbol{\mu}}_k; \underline{\mathrm{A}}_k = \underline{\mathrm{A}}_k}, \tag{5.15}$$

$$\underline{\mathrm{S}}_k = \underline{\mathrm{G}}_k \tilde{\underline{\Sigma}}_k \underline{\mathrm{G}}_k^T + \underline{\mathrm{R}}_k, \quad \tilde{\boldsymbol{\Upsilon}}_{\underline{\mathrm{A}}_k} = g(\tilde{\boldsymbol{\mu}}_k, \underline{\mathrm{A}}_k), \tag{5.16}$$

$$\underline{\mathrm{K}}_k = \tilde{\underline{\Sigma}}_k \underline{\mathrm{G}}_k^T \underline{\mathrm{S}}_k^{-1}, \tag{5.17}$$

$$\boldsymbol{\mu}_k = \tilde{\boldsymbol{\mu}}_k + \underline{\mathrm{K}}_k (\boldsymbol{\Upsilon}_k - \tilde{\boldsymbol{\Upsilon}}_{\underline{\mathrm{A}}_k}), \tag{5.18}$$

$$\underline{\Sigma}_k = (\underline{\mathrm{I}} - \underline{\mathrm{K}}_k \underline{\mathrm{G}}_k) \tilde{\underline{\Sigma}}_k. \tag{5.19}$$

$$\tag{5.20}$$

Here, the predicted quantities are indicated by a tilde over the respective symbols (i.e., the predicted measurements $\tilde{\boldsymbol{\Upsilon}}_{\underline{\mathrm{A}}_k}$, the predicted mean $\tilde{\boldsymbol{\mu}}_k$ and the associated covariance $\tilde{\underline{\Sigma}}_k$). Further, $\underline{\mathrm{Q}}_k$ is the linearized noise covariance of the motion model and $\underline{\mathrm{R}}_k$ the respective linearized noise covariance of the measurement model. Finally, $\underline{\mathrm{G}}_k$ denotes the Jacobian of the measurement function, $\underline{\mathrm{S}}_k$ is the innovation covariance and $\underline{\mathrm{K}}_k$ describes the Kalman gain.
The association between measurements and landmarks, $\underline{\mathrm{A}}_k$, is usually found using some sort of maximum likelihood heuristic, which can be problematic in some cases. Since the EKF-SLAM formulation only tracks one association hypothesis it is prone to fail if this hypothesis is incorrect [MT07]. Other than that it is still a good approach to solving SLAM problems.

FastSLAM

The FastSLAM algorithm by Montemerlo et al. [Mon03; MTKW02] can be recognized as one of the major breakthroughs in SLAM research. The great insight has been that "the SLAM problem can be factored into a set of independent landmark estimation problems conditioned on an estimate of the robot's path" [MT07, p. 28]. This observation enabled the use of particle filters to solve the SLAM problem. In a naive approach the particle state would consist of $2n_{M_k} + 3$ state variables, rendering it impracticable. In contrast, in FastSLAM the one huge estimation problem is split in many low dimensional estimation problems: A particle filter is used to estimate the path posterior $p(\boldsymbol{x}_{0:k} \mid \boldsymbol{\Upsilon}_{0:k}, \boldsymbol{u}_{1:k}, \underline{\mathrm{A}}_k, \boldsymbol{x}_0)$. Hence, the particles' states are hypotheses for the vehicles' poses. Further, each of the particles holds n_{M_k} EKFs – each of them tracking exactly one landmark, i.e., estimating the landmark posterior conditioned on the path $p(\boldsymbol{m}^{(j)} \mid \boldsymbol{x}_{0:k}, \boldsymbol{\Upsilon}_{0:k}, \boldsymbol{u}_{1:k}, \underline{\mathrm{A}}_k)$. The FastSLAM algorithm is an example of a "Rao-Blackwellized" particle filter, which, as a general concept, has already

been presented earlier e.g., by Doucet et al. [DFMR00; DGA00].
In contrast to EKF-SLAM, FastSLAM estimates the posterior over the maps and the vehicle paths, not the posterior over the maps and the vehicle pose [MT07]. However, since the only pose explicitly required by the update equations is the pose from the previous time step, the rest of the previous path can be discarded. The full SLAM posterior in FastSLAM is given by (cf. [MT07]):

$$p(\boldsymbol{x}_{0:k}, \boldsymbol{\Omega}_k \mid \boldsymbol{\Upsilon}_{0:k}, \boldsymbol{u}_{1:k}, \underline{\mathrm{A}}_{0:k}, \boldsymbol{x}_0) = p(\boldsymbol{x}_{0:k} \mid \boldsymbol{\Upsilon}_{0:k}, \boldsymbol{u}_{1:k}, \underline{\mathrm{A}}_{0:k}, \boldsymbol{x}_0) \prod_{j=1}^{n_{M_k}} p(\boldsymbol{m}^{(j)} \mid \boldsymbol{x}_{0:k}, \boldsymbol{\Upsilon}_{0:k}.\boldsymbol{u}_{1:k}, \underline{\mathrm{A}}_{0:k}), \quad (5.21)$$

The filtering process that generates this posterior follows the standard particle filtering algorithm (cf. Section 2.2), with the exception of the additional update of the landmark EKFs within the particle filter's update process. Obviously, the crucial part of this algorithm is to determine the particle weights. They are calculated by multiplying the weights of each of the observations j:

$$\omega_k^{(j)} = \frac{1}{\sqrt{\mid 2\pi \underline{\mathrm{S}}^{(j)}_{\underline{\mathrm{A}}_k,k} \mid}} \exp\{-\frac{1}{2}(\boldsymbol{z}_k^{(j)} - \tilde{\boldsymbol{z}}_k^{(j)})^T [\underline{\mathrm{S}}^{(j)}_{\underline{\mathrm{A}}_k,k}]^{-1} (\boldsymbol{z}_k^{(j)} - \tilde{\boldsymbol{z}}_k^{(j)})\} \quad (5.22)$$

with $\tilde{\boldsymbol{z}}_k^{(j)}$ being the predicted measurement and $\underline{\mathrm{S}}^{(j)}_{\underline{\mathrm{A}}_k,k}$ being the innovation covariance matrix (cf. [MT07]). This step inevitably requires some sort of feature management component in order to find the associations between predicted measurements and actual measurements.
Since FastSLAM scales logarithmically with the number of landmarks, it is much better suited for larger environments than EKF-SLAM. Further it is also superior in handling situations in which data association is ambiguous [MT03]. Getting rid of the explicit correlation of landmarks of course also has its drawbacks. The primary disadvantage of FastSLAM, when compared to EKF-SLAM, is that loop closing gets much more difficult [TBF05]. In SLAM, loop closing refers to the process of detecting that the vehicle has previously already been near the current position and then using that knowledge to correct the current state estimate. In contrast to EKF-SLAM, FastSLAM does not maintain direct correlations in the map anymore but only considers them through the diversity in the particle sets. Therefore, loop closing performance depends on the number of particles which again has an impact on the computational effort.

5.2.2 RFS-Based SLAM

Within the last decade Finite Set Statistics (FISST) has become a popular and powerful tool in the multi object tracking domain. Based on the work of Mahler [GMN97; Mah07], several implementations of the multi-object Bayes filter where the multi-object state is represented by a random finite set, have been realized. Mullane et al. [MVAW08] were the first to apply this approach to the SLAM problem. Since then, primarily variations of the PHD filter have been implemented in FISST-based SLAM publications [AMV13; AVMM14; ILA14; LCS12; LIA14; LNP+14; MVAV11b; MWR+14].
In contrast to the previously presented SLAM approaches, random finite set-based SLAM does not rely on vectors to represent the map and the measurements. Instead, as the name already implies, random finite sets are used. This obviously leads to entirely different filtering approaches, as described in Section 5.3. When doing SLAM, the number of detections in each time step is in fact a random variable and also the value of each of the measurements is random. Thus, a random finite set of measurements at time step k, Z_k, is defined. This RFS encapsulates all possible combinations of feature detections $\boldsymbol{z}_k^{(j)}$, i.e., from detecting nothing $\mathrm{Z}_k = \emptyset$ up to making n_{Z_k} detections $\mathrm{Z}_k = \{\boldsymbol{z}_k^{(1)}, ..., \boldsymbol{z}_k^{(n_{Z_k})}\}$. Finally, the map M_k is also modeled as an RFS comprising all the landmarks $\boldsymbol{m}^{(j)}, j = 1, ..., n_{M_k}$ that could have been observed up unto time step k (i.e., those landmarks that have passed through the field of view of the sensor). This representation directly facilitates a consideration of the existence or non-existence of landmarks.

5.2.3 Advantages of RFS-Based SLAM

As already illustrated in, e.g., [AMV13], representing measurements and landmarks as RFSs is more appropriate than using vectors. Using vectors would imply an order of the represented elements, which obviously does not really exist. So, if two vehicles pass through the same environment but along different routes, they would produce different maps (see Figure 5.2), although the landmarks are actually the same. In contrast, as the RFS naturally encapsulates all possible permutations of its elements, the feature association problem is entirely circumvented. This is not only beneficial when comparing maps of two different routes through the same area, but also in each time step when measurements and features are associated, cf. [MVAV11b] and Figure 5.3. Thus, one of the major problems of vector-based SLAM simply does not exist in RFS-SLAM.
Additionally, in a vector representation the dimension of the measurement vector in each time step is fixed, which neglects the possibility of false alarms and missed detections. The concept of measurement statistics, i.e., a false alarm rate and

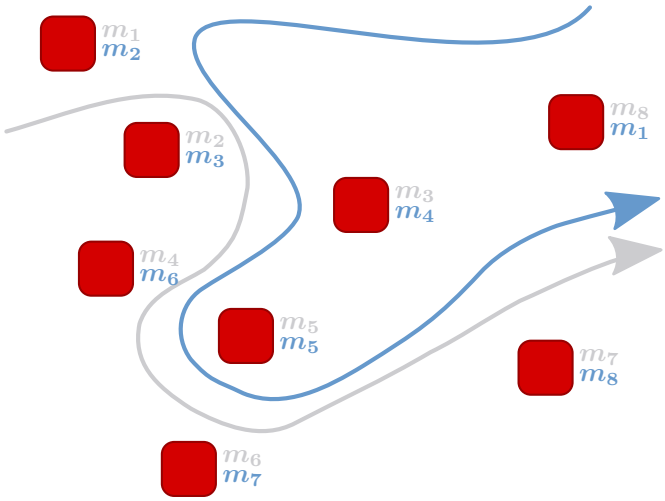


Figure 5.2: The order in which landmarks are measured should not matter, since landmarks do not posses an inherent ordering. Nevertheless, when using vector-based approaches the landmarks are stacked in a vector which constitutes an order. Thus, when two vehicles pass through the same environment along different paths, the results are two different maps (blue vs. gray indices indicate order of landmarks in the picture; cf. depiction in [AMV13]).

the probability of detecting a feature given the robot's pose, is usually ignored in vector-based formulations. On the contrary, the RFS representation enables the integrated, joint handling of the spatial uncertainty along with the uncertainty in feature existence. That is, the number of existing features is directly estimated throughout the filtering process respecting the false alarm rate and the probability of detection [AVMM14]. These are obviously great benefits, regarding the facts that when real sensors are used:

- measurements are noisy
- clutter measurements / false alarms do exist
- missed detections do occur

As opposed to this, in vector-based implementations, some sort of (usually heuristic) data association and map management component is inevitably necessary to be able to cope with these challenges. These components are not integrated in the Bayesian estimator which renders such an approach much more fragile.

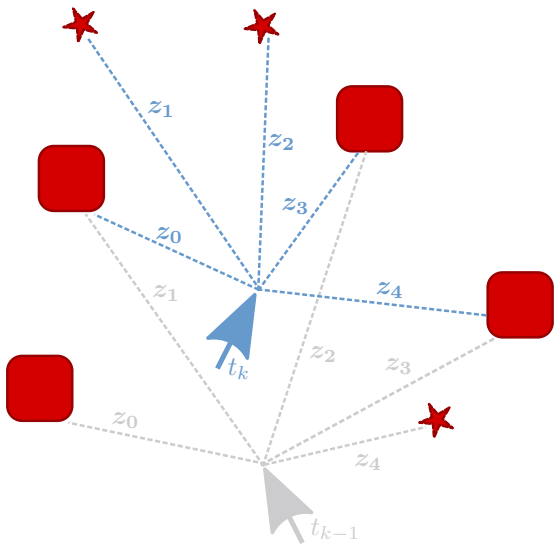


Figure 5.3: Feature to measurement association problem: The indices of the measurements that correspond to one feature (red boxes) may vary between time steps. This is due to the vehicle movement (i.e., the FOV changes) and the occurrence of missed detections and false measurements (red stars). Therefore, the association problem has to be solved when vector-based approaches are used (cf. depiction in [AMV13]).

5.3 Rao-Blackwellized-RFS-SLAM

This section describes the Rao-Blackwellized SLAM formulation for random finite sets, i.e., RB-RFS-SLAM. First of all, the foundations of RFS-based SLAM are described. Then, an approach using a PHD filter, which was presented by Mullane et al. [MVAV11b], is outlined in Section 5.3.2.

5.3.1 RFS-SLAM Foundations

The RFS-SLAM algorithms described in the following make use of Rao-Blackwellized particle filters, as it is also done in FastSLAM. Similarly to FastSLAM the PDF on the vehicle's trajectory is estimated by a particle filter. However, map estimation, and hence also the weighting of the particles, is carried out very differently than in FastSLAM.

The major concern of the following sections is the estimation of the map conditioned on the trajectory. The map and the trajectory may obviously be different for every particle. However, the explicit display of the index of the particle is omitted in the following whenever possible for the sake of clarity. Further, the respective probability densities that are used extensively are abbreviated as follows: The posterior probability density of the map, $\pi(\mathrm{M}_k \mid \mathrm{Z}_{0:k}, \boldsymbol{x}_{0:k})$, is abbreviated by π_{M} whenever possible. Further, the predicted probability density of the map, $\pi_{+}(\mathrm{M}_k \mid \mathrm{Z}_{0:k-1}, \boldsymbol{x}_{0:k})$ is abbreviated by $\pi_{\mathrm{M}+}$.

Map State Transition

As already mentioned, the map state is modeled as an RFS in RB-RFS-SLAM. Since the vehicle moves permanently, the known map is evidently subject to change while the landmarks themselves are assumed to be stationary. Thus, the map is said to be pseudo-stationary and the real feature map evolves according to the equation:

$$\mathrm{M}_k = \mathrm{M}_{k-1} \cup \mathrm{B}_k. \tag{5.23}$$

This modeling explicitly considers the creation of new landmarks within the sensor's field of view through the "Birth-RFS" B_k. This "Birth-RFS" models the landmarks being in the sensor's FOV at time step k that have not been in M_{k-1} (i.e., $(\mathrm{M} \setminus \mathrm{M}_{k-1}) \cap \mathrm{FOV}(\boldsymbol{x}_k)$, where M is the RFS representing the entire map and $\mathrm{FOV}(\boldsymbol{x}_k)$ is the sensor's FOV given the vehicle's pose). The map state transition density is then given by [AMV13]:

$$f_{\mathrm{M}}(\mathrm{M}_k \mid \mathrm{M}_{k-1}, \boldsymbol{x}_k) = \sum_{\mathrm{W} \subseteq \mathrm{M}_k} f_{\mathrm{M}}(\mathrm{W} \mid \mathrm{M}_{k-1}) f_{\mathrm{B}}(\mathrm{M}_k \setminus \mathrm{W} \mid \boldsymbol{x}_k), \tag{5.24}$$

with $f_{\mathrm{M}}(\mathrm{W} \mid \mathrm{M}_{k-1})$ being the transition density of the features having been in the FOV up until time step $k-1$ and $f_{\mathrm{B}}(\mathrm{M}_k \setminus \mathrm{W} \mid \boldsymbol{x}_k)$ being the density of the "Birth-RFS" B_k. Thus, it is basically an abstract adaption of the general multi-object transition density described in equation (2.28).

Measurement Modeling

First of all, measurements are not considered to be always true positive measurements, as it is the case with most other SLAM approaches. As already presented in [AMV13;

MVAV11b], this becomes obvious in the modeling of the measurement RFS:

$$\mathrm{Z}_k = \bigcup_{\boldsymbol{m} \in \mathrm{M}_k} \mathrm{D}_k(\boldsymbol{m}, \boldsymbol{x}_k) \cup \mathrm{C}_k(\boldsymbol{x}_k), \tag{5.25}$$

where the RFS of clutter measurements is denoted as $\mathrm{C}_k(\boldsymbol{x}_k)$ and the RFS of a feature measurement is $\mathrm{D}_k(\boldsymbol{m}, \boldsymbol{x}_k)$. It is modeled as a Bernoulli-RFS (cf. Section 2.3.1 and [MVAV11b]) and describes a feature measurement being generated by a landmark $\boldsymbol{m}$. Hence, $\mathrm{D}_k(\boldsymbol{m}, \boldsymbol{x}_k)$ may either represent a single measurement or may be empty. That is, with a probability of $1 - p_D(\boldsymbol{m}, \boldsymbol{x}_k)$ the RFS $\mathrm{D}_k(\boldsymbol{m}, \boldsymbol{x}_k)$ is empty:

$$\mathrm{D}_k(\boldsymbol{m}, \boldsymbol{x}_k) = \emptyset, \tag{5.26}$$

where $p_D(\boldsymbol{m}, \boldsymbol{x}_k)$ is the probability of detecting landmark $\boldsymbol{m}$ when being positioned as described through the pose $\boldsymbol{x}_k$. Moreover, for each $\boldsymbol{m} \in \mathrm{M}_k$ and $\boldsymbol{z} \in \mathrm{Z}_k$, the RFS $\mathrm{D}_k(\boldsymbol{m}, \boldsymbol{x}_k)$ is a singleton:

$$\mathrm{D}_k(\boldsymbol{m}, \boldsymbol{x}_k) = \{\boldsymbol{z}\}, \tag{5.27}$$

with the probability density $p_D(\boldsymbol{m}, \boldsymbol{x}_k) g(\boldsymbol{z} \mid \boldsymbol{m}, \boldsymbol{x}_k)$. Thereby, the likelihood that the vehicle's sensor measures $\boldsymbol{z}$ based on $\boldsymbol{m}$ and $\boldsymbol{x}_k$ is given by $g(\boldsymbol{z} \mid \boldsymbol{m}, \boldsymbol{x}_k)$. The feature measurement RFS therefore directly incorporates the handling of spatial uncertainty (through $g(\boldsymbol{z} \mid \boldsymbol{m}, \boldsymbol{x}_k)$) and the uncertainty of detection [AMV13]. Thus, in combination with the consideration of clutter measurements (by $\mathrm{C}_k(\boldsymbol{x}_k)$) this approach obviously enables the integrated handling of all kinds of measurement uncertainties.
Then, the probability density that the measurement RFS Z_k is produced (cf. [MVAV11b]), given the map and the pose is defined by:

$$g(\mathrm{Z}_k \mid \mathrm{M}_k, \boldsymbol{x}_k) = \sum_{\mathrm{W} \subseteq \mathrm{Z}_k} g_{\mathrm{D}}(\mathrm{W} \mid \mathrm{M}_k, \boldsymbol{x}_k) g_{\mathrm{C}}(\mathrm{Z}_k \setminus \mathrm{W}). \tag{5.28}$$

Again, this multi-object likelihood is an abstract reformulation of the general multi-object likelihood described by equation (2.32), where summing over the hypotheses now corresponds to summing over all subsets of Z_k. Spurious measurements of the sensor are considered for each subset W by the probability density $g_{\mathrm{C}}(\mathrm{Z}_k \setminus \mathrm{W})$, which is then the density of the clutter RFS C_k (cf. [MVAV11b]). Further, the density of the RFS of observations D_k, is denoted as $g_{\mathrm{D}}(\mathrm{W} \mid \mathrm{M}_k, \boldsymbol{x}_k)$, which describes the likelihood that a set of measurements W is actually received from the elements of the landmark map also considering the respective uncertainties as mentioned above (cf. [MVAV11b]).

RFS Bayes Recursion Formulation

Similar to FastSLAM, the full SLAM posterior is factored into the density of the map conditioned on the trajectory and the density of the trajectory:

$$\pi(\boldsymbol{x}_{1:k}, \mathrm{M}_k \mid \mathrm{Z}_{0:k}, \boldsymbol{u}_{1:k}, \boldsymbol{x}_0) = p(\boldsymbol{x}_{1:k} \mid \mathrm{Z}_{0:k}, \boldsymbol{u}_{1:k}, \boldsymbol{x}_0)\, \pi(\mathrm{M}_k \mid \boldsymbol{x}_{0:k}, \mathrm{Z}_{0:k}). \tag{5.29}$$

The Bayes recursion can then be equivalently executed by jointly propagating the posterior density of the map conditioned on the trajectory and the posterior density of the trajectory [MVAV11b]. Based on this, a Rao-Blackwellized (RB) particle filter implementation can be executed by using the following equations. The posterior density of the vehicle's trajectory is (cf. [MVAV11a]):

$$p(\boldsymbol{x}_{1:k} \mid \mathrm{Z}_{0:k}, \boldsymbol{u}_{1:k}, \boldsymbol{x}_0) = g(\mathrm{Z}_k \mid \mathrm{Z}_{0:k-1}, \boldsymbol{x}_{0:k}) \times \frac{f_{\boldsymbol{x}}(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}, \boldsymbol{u}_k) p(\boldsymbol{x}_{1:k-1} \mid \mathrm{Z}_{0:k-1}, \boldsymbol{u}_{1:k-1}, \boldsymbol{x}_0)}{g(\mathrm{Z}_k \mid \mathrm{Z}_{0:k-1})}, \tag{5.30}$$

using the measurement likelihood function

$$g(\mathrm{Z}_k \mid \mathrm{Z}_{0:k-1}, \boldsymbol{x}_{0:k-1}) = \int \pi(\mathrm{Z}_k, \mathrm{M}_k \mid \mathrm{Z}_{0:k-1}, \boldsymbol{x}_{0:k}) \delta \mathrm{M}_k. \tag{5.31}$$

Further, the map posterior density is given by (cf. [MVAV11a]):

$$\pi_{\mathrm{M}} = \frac{g(\mathrm{Z}_k \mid \mathrm{M}_k, \boldsymbol{x}_k) \pi_{\mathrm{M}+}}{g(\mathrm{Z}_k \mid \mathrm{Z}_{0:k-1}, \boldsymbol{x}_{0:k})}, \tag{5.32}$$

with the predicted probability density of the map being

$$\pi_{\mathrm{M}+} = \int f_{\mathrm{M}}(\mathrm{M}_k \mid \mathrm{M}_{k-1}, \boldsymbol{x}_k) \pi(\mathrm{M}_{k-1} \mid \mathrm{Z}_{0:k-1}, \boldsymbol{x}_{0:k-1}) \delta \mathrm{M}_{k-1}. \tag{5.33}$$

As this full recursion includes the calculation of set integrals, it is computationally intractable. Therefore, some approximations have to be made. They are described in the following sections for RB-PHD-SLAM, and RB-LMB-SLAM respectively.

5.3.2 RB-PHD-SLAM

As the name RB-PHD-SLAM already implies, an RB formulation is employed again. The consequence of this is that the filter estimates multiple independent map PHDs, each conditioned on the vehicle trajectory represented by the respective particle and its according weight. Thus, the PHD filter is solely used to estimate these maps.

The PHD Filter for SLAM

The estimated map of each particle at time step k is approximated by its respective PHD, v, and represented as Gaussian mixture [MVAV11b]:

$$v_k(\boldsymbol{m} \mid \mathrm{Z}_{0:k}\boldsymbol{x}_{0:k}) = \sum_{j}^{J_k} w_k^{(j)} \mathcal{N}\left(\boldsymbol{m}; \boldsymbol{\mu}_k^{(j)}, \underline{\mathrm{P}}_k^{(j)}\right), \tag{5.34}$$

with the Gaussians being defined by their weights $w_k^{(j)}$, mean values $\boldsymbol{\mu}_k^{(j)}$ and covariances $\underline{\mathrm{P}}_k^{(j)}$.
If it is assumed that the map (and the clutter measurements) follows a multi-object Poisson distribution [LIA14; MVAV11a] (implying that the features of the map are assumed to be Independent and Identically Distributed (i.i.d.) and that their cardinality follows a Poisson distribution), then it is possible to recover the probability density of the map from the PHD intensity function [AMV13]. Thus, with these assumptions a computationally tractable implementation of the map estimation process is possible: The predicted and posterior maps are approximated by Poisson RFSs (cf. Section 2.3.1) utilizing the first order moments of the respective RFSs (i.e., their PHDs $v_+(\boldsymbol{m} \mid \mathrm{Z}_{0:k-1}, \boldsymbol{x}_{0:k})$ and $v(\boldsymbol{m} \mid \mathrm{Z}_{0:k}, \boldsymbol{x}_{0:k})$, cf. [MVAV11b]):

$$\pi_{\mathrm{M}+} \approx \frac{\prod\limits_{\boldsymbol{m}\in\mathrm{M}_k} v_+(\boldsymbol{m} \mid \mathrm{Z}_{0:k-1}, \boldsymbol{x}_{0:k})}{\exp(\int v_+(\boldsymbol{m} \mid \mathrm{Z}_{0:k-1}, \boldsymbol{x}_{0:k}) d\boldsymbol{m})} \tag{5.35}$$

and

$$\pi_{\mathrm{M}} \approx \frac{\prod\limits_{\boldsymbol{m}\in\mathrm{M}_k} v(\boldsymbol{m} \mid \mathrm{Z}_{0:k}, \boldsymbol{x}_{0:k})}{\exp(\int v(\boldsymbol{m} \mid \mathrm{Z}_{0:k}, \boldsymbol{x}_{0:k}) d\boldsymbol{m})}. \tag{5.36}$$

Given these approximations, the essence of the PHD filter, i.e., its predictor- corrector-equations (cf. Section 2.3.2) adapted for estimating the SLAM-state can be used and are described in the following. First of all, the predicted map in RB-PHD-SLAM is calculated by expressing the map state transition equation (5.24) in terms of the PHDs of the RFSs, i.e.:

$$v_+(\boldsymbol{m} \mid \mathrm{Z}_{0:k-1}, \boldsymbol{x}_{0:k}) = v_{k-1}(\boldsymbol{m} \mid \mathrm{Z}_{0:k-1}\boldsymbol{x}_{0:k-1}) + b(\boldsymbol{m} \mid \mathrm{Z}_{0:k-1}\boldsymbol{x}_{0:k}), \tag{5.37}$$

with the $b(\boldsymbol{m} \mid \mathrm{Z}_{0:k-1}\boldsymbol{x}_{0:k})$ being the PHD of the "Birth-RFS" [MVAV11b].
Using the measurement likelihood $g(\boldsymbol{z} \mid \boldsymbol{m}, \boldsymbol{x}_k)$, the probability of detection $p_D(\boldsymbol{m}, \boldsymbol{x}_k)$ and the PHD of the clutter RFS $\mathrm{C}_k(\boldsymbol{x}_k)$, denoted as $\kappa_k(\boldsymbol{z} \mid \boldsymbol{x}_k)$, the PHD-corrector

is then given by [MVAV11b]:

$$v(\boldsymbol{m} \mid \boldsymbol{x}_{0:k}) = v_+(\boldsymbol{m} \mid \boldsymbol{x}_{0:k}) \Bigg[1 - p_D(\boldsymbol{m}, \boldsymbol{x}_k) + \sum_{\boldsymbol{z} \in \mathrm{Z}_k} \frac{p_D(\boldsymbol{m}, \boldsymbol{x}_k) g(\boldsymbol{z} \mid \boldsymbol{m}, \boldsymbol{x}_k)}{\kappa_k(\boldsymbol{z} \mid \boldsymbol{x}_k) + \int p_D(\boldsymbol{\xi}, \boldsymbol{x}_k) g(\boldsymbol{z} \mid \boldsymbol{\xi}, \boldsymbol{x}_k) v_+(\boldsymbol{\xi} \mid \boldsymbol{x}_{0:k}) d\boldsymbol{\xi}} \Bigg]. \tag{5.38}$$

Despite the approximations introduced along with the PHD filter, the calculation of the SLAM posterior (5.29) still requires set integration, which is numerically intractable. Particularly, the computation of equation (5.30) is numerically intractable, since it involves the computation of the set integral of equation (5.31) (cf. [MVAV11b]).
However, solving equation (5.32) for $g(\mathrm{Z}_k \mid \mathrm{Z}_{0:k-1}, \boldsymbol{x}_{0:k-1})$ results in

$$g(\mathrm{Z}_k \mid \mathrm{Z}_{0:k-1}, \boldsymbol{x}_{0:k-1}) = \frac{g(\mathrm{Z}_k \mid \mathrm{M}_k, \boldsymbol{x}_k) \pi_+(\mathrm{M}_k \mid \mathrm{Z}_{0:k}, \boldsymbol{x}_{0:k})}{\pi(\mathrm{M}_k \mid \mathrm{Z}_{0:k}, \boldsymbol{x}_{0:k})}, \tag{5.39}$$

and it can be seen, that the map M_k does not occur on the left hand side, but in the nominator and denominator of the right hand side. This, according to [MVAV11b], leads to the conclusion, that the map is only a dummy variable. Thus, as the left hand side is independent of M_k, equation (5.39) can be evaluated for any arbitrary choice of M_k. Based on this finding, it is possible to calculate the measurement likelihood conditioned only on the trajectory in closed form.

Particle Weighting

As with FastSLAM, the vehicle transition density is chosen as the proposal density in RB-PHD-SLAM [MVAV11b]. Therefore, the weight of particle i can similarly be calculated by:

$$\omega_k^{(i)} = g(\mathrm{Z}_k \mid \mathrm{Z}_{0:k-1}, \boldsymbol{x}_{0:k}^{(i)}) \omega_{k-1}^{(i)}. \tag{5.40}$$

The actual calculation of the particle weights nevertheless differs severely from the weight calculation in FastSLAM since it is not possible to approximate the measurement likelihood under an EKF-Framework when using random finite sets. However, the finding that the likelihood of equation (5.39) can be evaluated for any arbitrary choice of M_k allows for different strategies to determine the likelihood $g(\mathrm{Z}_k \mid \mathrm{Z}_{0:k-1}, \boldsymbol{x}_{0:k}^{(i)})$. The following three strategies are the most common and have been published in [MVAV11b] and [LIA13] respectively:

- The empty map strategy: All features are considered Poisson distributed clutter, i.e., $\mathrm{M}_k = \emptyset$.

- The single-feature strategy: All but one single feature are considered to be clutter for the calculation of the particle weights, i.e., $\mathrm{M}_k = \{\boldsymbol{m}^*\}$. This feature could be chosen by selecting the one with the highest measurement likelihood or the lowest uncertainty.

- The multi-feature strategy: M_k is chosen such that it includes all estimated landmarks (up to an upper bound) being in the FOV of the vehicle. This is much more computationally demanding than the other two approaches. Details on the necessary approximations for keeping the algorithm computationally tractable can be found in [LIA13].

Although equation (5.39) may be evaluated for an arbitrary map, the different strategies do not lead to the same exact results. The reason for this is that the actual map is approximated and not exactly represented (cf. equations (5.35), (5.36)) [LIA13; MVAV11b]. Thus, the difference in the quality of the approximations that are made by the different approaches may lead significantly different filter performances. According to Leung et al. [LIA13] the last option, as one might intuitively expect, outperforms the other approaches. Therefore, also multiple features have been used for the evaluation presented in Section 5.5.

Finally, all the weighted particles of RB-PHD-SLAM together directly constitute a representation of the estimated path and map: In RB-PHD-SLAM it is possible to determine the posterior of the map by averaging over all feature maps. That is, as explained in [MVAV11b], the expectation of the trajectory-conditioned PHDs already delivers the posterior PHD of the landmark map:

$$v_k(\boldsymbol{m}) = \mathbb{E}[v_k(\boldsymbol{m} \mid \boldsymbol{x}_{0:k})]. \tag{5.41}$$

This is a natural result of the PHD formulation, and is therefore entirely mathematically motivated. In contrast to that, in FastSLAM either the particle with the highest weight is selected to represent the estimated path and the respective landmark map or some kind of heuristic is used to determine an average feature map.

5.4 RB-LMB-SLAM

This section details the novelty presented in this thesis: The Rao-Blackwellized Labeled Multi-Bernoulli SLAM. In the following it will be shown how LMB filters can be used to estimate the feature map conditioned on the vehicle's trajectory. The factorization of the SLAM posterior (equation (5.29)), requires the estimation of the multi-object probability density $\pi(\mathrm{M}_k \mid \boldsymbol{x}_{0:k}, \mathrm{Z}_{0:k}, \boldsymbol{u}_{1:k})$, which can be done by using an LMB filter. Hence, this chapter focuses exactly on this part of the estimation problem. Similar to FastSLAM and RB-PHD-SLAM a Rao-Blackwellized particle filter is used. Thus, this part of the algorithm will not be described again (see Sections 2.2, 4.4, 5.2.1 and 5.3.2). Nevertheless, the weighting of the particles, which obviously has a severe effect on the performance of the algorithm, and which again is highly dependent on the underlying map estimation, is also described in this section (see Section 5.4.2). Further, it is shown that the LMB filter, in contrast to the PHD approach, directly provides an exact solution for determining the particle weights.

5.4.1 The Labeled Multi-Bernoulli Filter for SLAM

The adaption of the LMB filter to be used in RB-LMB-SLAM is described below. The particular steps of the filtering process, as depicted in Figure 5.4, are outlined one by one. The subsequent description and also the presented equations of this specialized LMB filter are closely along the lines of the work of Reuter, who originally presented the LMB filter for multi-object tracking [Reu14; RVVD14]. The derivations of the general LMB filter prediction and update steps can be found in [Reu14].

LMB Representation

In the LMB filter for SLAM, the predicted and posterior multi-object densities are represented by LMB RFSs which are described by their respective parameter set:

$$\boldsymbol{\pi}_{+}(\mathbf{M}_k \mid \boldsymbol{x}_{0:k}, \mathrm{Z}_{0:k-1}) = \{(r_{+}^{(\ell)}, p_{+}^{(\ell)})\}_{\ell \in \mathbb{L}_{+}} \tag{5.42}$$

$$\boldsymbol{\pi}(\mathbf{M}_k \mid \boldsymbol{x}_{0:k}, \mathrm{Z}_{0:k}) = \{(r^{(\ell)}, p^{(\ell)})\}_{\ell \in \mathbb{L}}, \tag{5.43}$$

where the labeled multi-object state $\mathbf{M}_k$ is a finite set on the space $\mathbb{M} \times \mathbb{L}$, which is spanned by the state space of the landmarks $\mathbb{M}$ and the discrete space of labels $\mathbb{L}$. Further, $r^{(\ell)}$ denotes the existence probability of an object and $p^{(\ell)}$ its spatial distribution. For the sake of compactness, either the parameter set representations or the abbreviations $\boldsymbol{\pi}_{\mathrm{M}+}$ and $\boldsymbol{\pi}_{\mathrm{M}}$ are used in the following. The explicit display

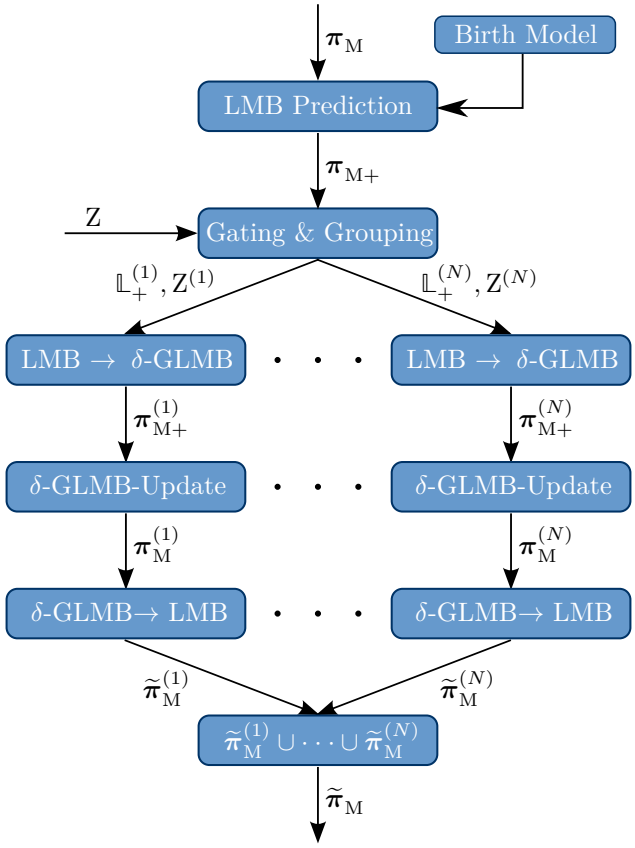


Figure 5.4: The LMB filtering process (cf. depiction in [RVVD14]).

of the conditioning on the states $\boldsymbol{x}_{0:k}$ and the measurements $\mathrm{Z}_{0:k}$ is omitted (but nevertheless valid) whenever possible. For the same reason, the time index k is omitted as well whenever it is not inevitably necessary. Similar to RB-PHD-SLAM, a GM implementation is used to model the spatial distribution $p^{(\ell)}(\cdot)$. Thus, the posterior probability density of any of the labeled Bernoulli tracks, with label $\ell \in \mathbb{L}$, is given by:

$$p^{(\ell)}(\boldsymbol{m}) = \sum_{j=1}^{J^{(\ell)}} w^{(\ell,j)} \mathcal{N}\left(\boldsymbol{m}; \boldsymbol{\mu}_{k-1}^{(\ell,j)}, \underline{\mathrm{P}}_{k-1}^{(\ell,j)}\right). \tag{5.44}$$

The mean value of the jth Gaussian component is denoted by $\boldsymbol{\mu}_{k-1}^{(\ell,j)}$ and its respective estimation error covariance is $\underline{\mathrm{P}}_{k-1}^{(\ell,j)}$.

Prediction

As described earlier, the predicted map follows the map state transition as described by equation (5.24). Thus, the predicted LMB distribution is composed of the set of surviving landmarks and the newly created landmarks (cf. [Reu14]):

$$\boldsymbol{\pi}_{\mathrm{M}+} = \left\{ \left(r_{+,S}^{(\ell)}, p_{+,S}^{(\ell)} \right) \right\}_{\ell \in \mathbb{L}} \cup \left\{ \left(r_{B}^{(\ell)}, p_{B}^{(\ell)} \right) \right\}_{\ell \in \mathbb{B}}, \tag{5.45}$$

where $\mathbb{B}$ is the label space of the new born tracks and $\mathbb{B} \cap \mathbb{L} = \emptyset$.
Within the context of landmark map estimation, the general case of of the LMB prediction can be simplified. In the original formulation, when using a GM implementation, the state transition is required to follow a linear Gaussian process model with state transition matrix $\underline{\mathrm{F}}$ and process noise covariance $\underline{\mathrm{Q}}$, i.e., $f_+(\boldsymbol{m}|\boldsymbol{\xi}) = \mathcal{N}\left(\boldsymbol{m}; \underline{\mathrm{F}}\boldsymbol{\xi}, \underline{\mathrm{Q}}\right)$ (cf. [Reu14]). However, since the landmarks are assumed to be stationary, the state transition matrix equals the identity matrix $\underline{\mathrm{F}} = \mathbb{1}$. Using the survival probability p_S, the predicted existence probability and the predicted spatial distribution of the surviving track ℓ are given by

$$r_{+,S}^{(\ell)} = r^{(\ell)} \cdot p_S, \tag{5.46}$$

$$p_{+,S}^{(\ell)}(\boldsymbol{m}) = p^{(\ell)}(\boldsymbol{m}). \tag{5.47}$$

The survival probability $p_S < 1$ is constant for the tracks which are definitely within the sensor's FOV. For those in the border area of the FOV this probability may be chosen to increase towards $p_S = 1$, which then holds for all tracks outside of the FOV. Although, the predicted spatial distribution is defined to be the same as the prior distribution, adding a small additive Gaussian noise to all landmarks being in the FOV of the sensor during the prediction step might be beneficial for the robustness of the algorithm. This is similarly done in the implementation of the RB-PHD-SLAM [LI14].
Further, the second part of equation (5.45) has to be taken into consideration as well: the new born tracks. An adaptive approach that uses the measurements of the previous time step Z_{k-1} to create the birth distribution of the current time step k, is employed here (cf. [Reu14]):

$$\boldsymbol{\pi}_B(\mathbf{B}_k \mid \boldsymbol{x}_{0:k}, \mathrm{Z}_{0:k-1}) = \left\{ r_B^{(\alpha_i)}(\boldsymbol{m}|\boldsymbol{z}_i), p_B^{(\alpha_i)}(\boldsymbol{m}|\boldsymbol{z}_i) \right\}_{i=1}^{|\mathrm{Z}_{k-1}|}.$$

The labels α_i originate from the label space of new tracks $\mathbb{B} = \{\alpha_i : i \in \{1, ..., |\mathrm{Z}_{k-1}|\}\}$. In one possible implementation, these labels could be obtained by creating a unique identifier from the combination of the measurement's timestamp and the measured relative position. This way, it would also be ensured that each and every particle's

LMB filter uses the same labels for the same landmark.
When creating the birth distribution using the measurements of the previous time step, those measurements that have not contributed in any hypotheses during the update in the previous time step (cf. the paragraph about the update below) should obviously be more likely to create a birth component than those that have been used in all hypotheses with a high weight. To achieve this, the probability that a measurement $\boldsymbol{z} \in \mathrm{Z}^{(i)}$, ($\mathrm{Z}^{(i)}$ contains the measurements being in group i, cf. the paragraph about grouping below), has been assigned to an existing track is calculated by (cf. [Reu14]):

$$r_{U,k-1}(\boldsymbol{z}) = \sum_{(\mathcal{I}_+,\theta)\in\mathcal{F}(\mathbb{L}_{+,k-1}^{(i)})\times\Theta_{\mathcal{I}_+}^{(i)}} 1_\theta(\boldsymbol{z}) w_{k-1}^{(\mathcal{I}_+,\theta)}(\mathrm{Z}_{k-1}^{(i)}). \tag{5.48}$$

As described by the equation, in order to obtain such a probability all hypothesis weights $w_{k-1}^{(\mathcal{I}_+,\theta)}(\mathrm{Z}_{k-1}^{(i)})$ in which the measurement $\boldsymbol{z}$ is assigned to an existing track (ensured by the inclusion function $1_\theta(\boldsymbol{z})$) are simply summed up. Thus, the sum runs over $(\mathcal{I}_+,\theta)$, i.e., the hypotheses θ and their associated sets of predicted labels $\mathcal{I}_+$ that are within $\mathcal{F}(\mathbb{L}_{+,k-1}^{(i)}) \times \Theta_{\mathcal{I}_+}^{(i)}$, which again describes the space that is spanned by all finite subsets of the predicted labels within group i at the previous time step $\mathcal{F}(\mathbb{L}_{+,k-1}^{(i)})$ and the space of all possible association hypotheses for the tracks and measurements within that group, $\Theta_{\mathcal{I}_+}^{(i)}$. More details on the respective quantities, e.g., how the hypothesis weights are calculated (cf. equation (5.66)), are described in the following sections.
Equation (5.48) only considers measurements that have been assigned to a group. Those measurements that have not been assigned to any group have definitely not been used during the update process and consequently have zero probability of having been associated to any track, i.e., $r_{U,k-1}(\boldsymbol{z}) = 0$. Based on this finding and equation (5.48), the birth existence probability $r_B^{(\ell)}$ (with $\ell \in \mathbb{B}$) can be determined by (cf. [Reu14])

$$r_B^{(\ell)}(\boldsymbol{z}) = \min\left(r_B^{\max},\ \frac{1 - r_{U,k-1}(\boldsymbol{z})}{\sum_{\boldsymbol{\xi}\in\mathrm{Z}_{k-1}} 1 - r_{U,k-1}(\boldsymbol{\xi})} \cdot \lambda_B \right), \tag{5.49}$$

where the expected number of new born objects is λ_B. The minimum operator ensures that a maximum birth existence probability $r_B^{\max} \in [0,1]$ is not exceeded.

Finally, the spatial distribution of the new born tracks is given by

$$p_B^{(\ell)} = \sum_{j=1}^{J_B} w_B^{(\ell,j)} \mathcal{N}\left(\boldsymbol{m}; \hat{\boldsymbol{m}}_B^{(\ell,j)}, \underline{\mathrm{P}}_B^{(\ell,j)}\right), \tag{5.50}$$

where the number of Gaussians J_B is a parameter of the birth model and the states $\hat{\boldsymbol{m}}_B^{(\ell,j)}$ as well as the according estimation error covariances $\underline{\mathrm{P}}_B^{(\ell,j)}$ are determined by a transformation of the corresponding measurement $\boldsymbol{z}$ to the state space (e.g. from the polar coordinate system of a sensor to Euclidean xy-coordinates).

Update

The update of the LMB filter requires several steps, since the LMB RFS is not closed with respect to the measurement update [RVVD14]. This is because an LMB RFS is defined as union of independent Bernoulli components. However, updating the tracks cannot be done independently for these components if it is assumed that one measurement is generated by not more than one object: This assumption implies that if a measurement j is associated to one track ℓ_{i^*} (i.e., $\theta(\ell_{i^*}) = j$, with θ being the association map) it may not be associated to another track (i.e. $\theta(\ell_i) \neq j \; \forall \; i \neq i^*$). Hence all tracks and measurements have to be taken into consideration at the same time during the update which renders an independent update impossible. Therefore, the definition of the LMB RFS would be contradicted.
Thus, for executing the measurement update, the predicted LMB RFS is replaced by its δ-GLMB representation first (cf. Section 2.3.1). Then, the full δ-GLMB update is performed, before the updated δ-GLMB RFS is finally approximated by an LMB RFS. Although it is possible to perform the full δ-GLMB update at once, it makes sense to first group the landmarks and measurements with respect to their position in space. The grouping has a very significant performance impact if the landmarks and measurements can be separated into distinct groups, since the updates of the different groups can be run parallel. The grouping again, requires the likelihoods of all track to measurement associations, which are also a prerequisite when calculating the respective updated posterior distributions. Hence, it makes sense to calculate them as a very first step and to store them until they are used again during the actual update. In the following paragraphs, all required steps are described in more detail within the following order: calculation of the association likelihoods, gating and grouping, determination of the δ-GLMB representation of the predicted LMB densities, δ-GLMB-update, LMB approximation of the updated δ-GLMB densities and finally unification of the groups and track extraction.

Association Likelihoods For an association map θ, the likelihood of the track (label) to measurement association is given by the inner product of the predicted spatial distribution and the generalized measurement likelihood (cf. [Reu14]):

$$\eta_{\mathrm{Z},\boldsymbol{x}}^{(\theta)}(\ell) = \left\langle p_+(\cdot,\ell), \phi_{\mathrm{Z},\boldsymbol{x}}(\cdot,\ell;\theta) \right\rangle. \tag{5.51}$$

The generalized measurement likelihood $\phi_{\mathrm{Z},\boldsymbol{x}}(\boldsymbol{m},\ell;\theta)$ which incorporates the possibility of a missed detection (i.e., $\theta(\ell) = 0$) as well as an update with a measurement $\boldsymbol{z}_{\theta(\ell)}$ (i.e., $\theta(\ell) > 0$), is defined as

$$\phi_{\mathrm{Z},\boldsymbol{x}}(\boldsymbol{m},\ell;\theta) = \delta_0(\theta(\ell)) q_D(\boldsymbol{m},\ell \mid \boldsymbol{x}_{0:k}) + (1-\delta_0(\theta(\ell))) \frac{p_D(\boldsymbol{m},\ell) g(\boldsymbol{z}_{\theta(\ell)} | \boldsymbol{m},\ell,\boldsymbol{x}_{0:k})}{\kappa(\boldsymbol{z}_{\theta(\ell)})}. \tag{5.52}$$

In this equation, the probability of a missed detection is denoted as $q_D(\boldsymbol{m},\ell) = 1 - p_D(\boldsymbol{m},\ell \mid \boldsymbol{x}_{0:k})$ and, similar to Section 5.3.1, the spatial likelihood of the measurement $\boldsymbol{z}_{\theta(\ell)}$ and the landmark track labeled ℓ is described by $g(\boldsymbol{z}_{\theta(\ell)}|\boldsymbol{m},\ell,\boldsymbol{x}_{0:k})$. Moreover, the clutter process is again assumed to follow a Poisson distribution and accounted for by $\kappa(\boldsymbol{z}_{\theta(\ell)}) = \lambda_c c(\boldsymbol{z}_{\theta(\ell)})$ which represents the intensity function of such a process with a mean number of λ_c clutter measurements. In the following, the detection probability and the probability for a missed detection are assumed to be state independent [UEW07]. Based on this, the likelihood of the association of measurement $\boldsymbol{z}_{\theta(\ell)}$ to track ℓ expressed in terms of the GM implementation becomes (cf. [Reu14])

$$\eta_{\mathrm{Z},\boldsymbol{x}}^{(\theta)}(\ell) = \frac{p_D}{\kappa(\boldsymbol{z}_{\theta(\ell)})} \sum_{j=1}^{J_+^{(\ell)}} w_+^{(\ell,j)} \mathcal{N}\left(\boldsymbol{z}_{\theta(\ell)}; \boldsymbol{z}_+^{(\ell,j)}, \underline{\mathrm{S}}^{(\ell,j)}\right). \tag{5.53}$$

The predicted measurement $\boldsymbol{z}_+^{(\ell,j)}$ and the innovation covariance $\underline{\mathrm{S}}^{(\ell,j)}$ are determined through the standard EKF (or Kalman filter) equations using the measurement noise $\underline{\mathrm{R}}$ and the (nonlinear) measurement function $h(\cdot)$

$$\boldsymbol{z}_+^{(\ell,j)} = h(\boldsymbol{\mu}_+^{(\ell,j)}, \boldsymbol{x}_+), \tag{5.54}$$

$$\underline{\mathrm{H}} = \left.\frac{\partial h}{\partial \boldsymbol{m}}\right|_{\boldsymbol{\mu}_+^{(\ell,j)}}, \tag{5.55}$$

$$\underline{\mathrm{S}}^{(\ell,j)} = \underline{\mathrm{H}} \mathrm{P}_+^{(\ell,j)} \underline{\mathrm{H}}^{\mathrm{T}} + \underline{\mathrm{R}}. \tag{5.56}$$

Further, the likelihood of associating the track ℓ to the missed detection is given by

$$\eta_{\mathrm{Z},\boldsymbol{x}}^{(\theta)}(\ell) = q_D. \tag{5.57}$$

These association likelihoods $\eta_{\mathrm{Z},\boldsymbol{x}}^{(\theta)}(\ell)$ can be used directly for the grouping described below as well as during the subsequent δ-GLMB update, and should hence be stored temporarily.

Gating and Grouping In order to reduce computation times, the measurements and tracks are grouped according to their position. The updates for the distinct groups then can be run in parallel which is a great benefit when processor architectures with a highly-parallel structure are used (e.g., Graphics Processing Units (GPU)). The partitioning of the predicted LMB RFS $\boldsymbol{\pi}_+$ into mutually exclusive subsets makes use of the gating of the measurements as presented in [Reu14]. Gating aims at finding a reasonable set of measurements $\mathrm{Z}^{(\ell)}$ for each track ℓ, i.e., those measurements that exhibit a spatial likelihood significantly above zero for the given track. In the GM implementation of the RB-LMB-SLAM filter, the decision whether or not a measurement $\boldsymbol{z}_{\theta(\ell)}$ is within the gate of track ℓ, is made based on the minimum of the Mahalanobis Distance (MHD) between all of the j GM components of the track and the measurement:

$$d_{\mathrm{MHD}}^2(\ell) \triangleq \min_{j \in J^{(\ell)}} \left[\left(\boldsymbol{z}_{\theta(\ell)} - \boldsymbol{z}_+^{(\ell,j)} \right)^{\mathrm{T}} \left(\underline{\mathrm{S}}^{(\ell,j)} \right)^{-1} \left(\boldsymbol{z}_{\theta(\ell)} - \boldsymbol{z}_+^{(\ell,j)} \right) \right]. \tag{5.58}$$

Here, the previously determined predicted measurements (cf. equation (5.54)) and innovation covariances (cf. equation (5.56)) are used. Whether a measurement is within the gate of the landmark track or not, is determined by using an application dependent threshold ϑ, i.e., the equation $d_{\mathrm{MHD}}^2(\ell) < \vartheta$ is evaluated. Now, by enumerating the measurements in Z by $\mathrm{Z} = \{1, ..., |\mathrm{Z}|\}$, the measurements being within the gate of track ℓ can be represented by the set $\mathrm{Z}^{(\ell)} \subseteq \mathrm{Z}$. Taking the gating of the measurements as a premise, the groups $\mathbb{L}_+^{(n)}$ of the predicted track labels can be determined, such that the union of all groups contains all the predicted track labels $\mathbb{L}_+$ and that every label only occurs in one group, i.e., $\mathbb{L}_+ = \bigcup_{n=1}^{N} \mathbb{L}_+^{(n)}$ and $n \neq m \implies \mathbb{L}_+^{(n)} \cap \mathbb{L}_+^{(m)} = \emptyset$ (cf. [Reu14]). Consequently, the measurements have to be partitioned accordingly into disjoint sets as well. In that process also the set of measurements not associated to any track, Z^0, has to be taken into consideration:

$$\mathrm{Z} = \mathrm{Z}^0 \cup \bigcup_{n=1}^{N} \mathrm{Z}^{(n)}.$$

A grouping is then given by

$$\mathcal{G}^{(n)} = \left(\mathbb{L}_+^{(n)}, \mathrm{Z}^{(n)} \right). \tag{5.59}$$

These groupings are found by starting off with $|\mathbb{L}_+|$ groupings, one for each track ℓ:

$$\mathcal{G}^{(\ell)} = \left(\{\ell\}, \mathrm{Z}^{(\ell)} \right). \tag{5.60}$$

Then, all groupings i, j sharing at least one measurement, i.e. $\mathrm{Z}^{(i)} \cap \mathrm{Z}^{(j)} \neq \emptyset$ have to be merged according to

$$\mathcal{G}^{(i)} \cup \mathcal{G}^{(j)} = \left(\mathbb{L}_+^{(i)} \cup \mathbb{L}_+^{(j)}, \mathrm{Z}^{(i)} \cup \mathrm{Z}^{(j)} \right). \tag{5.61}$$

This procedure, as originally described in [Reu14], then results in N groups which also constitutes a partitioning of the predicted LMB distribution

$$\boldsymbol{\pi}_{\mathrm{M}+} = \bigcup_{i=1}^{N} \left\{ \left(r_{+,i}^{(\ell)}, p_{+,i}^{(\ell)} \right) \right\}_{\ell \in \mathbb{L}_+^{(i)}}. \tag{5.62}$$

For the purpose of doing SLAM, where stationary landmarks are tracked, a sensible gating is usually possible. Thus, the grouping procedure leads to good results.

δ-GLMB Representation In order to execute the update, the δ-GLMB representations of the groups of predicted LMB distributions described in equation (5.62) are required. For the grouping $\mathcal{G}^{(i)}$ which contains the track labels $\mathbb{L}_+^{(i)}$ the predicted δ-GLMB density which is equivalent to the respective predicted LMB density is given by (cf. [Reu14]):

$$\boldsymbol{\pi}_{\widetilde{\mathrm{M}}+}^{(i)} = \Delta(\widetilde{\mathbf{M}}_+^{(i)}) \sum_{\mathcal{I}_+ \in \mathcal{F}(\mathbb{L}_+^{(i)})} w_{+,i}^{(\mathcal{I}_+)} \delta_{\mathcal{I}_+}(\mathscr{L}(\widetilde{\mathbf{M}}_+^{(i)})) \left[p_+ \right]^{\widetilde{\mathbf{M}}_+^{(i)}}, \tag{5.63}$$

where the discrete index set Ξ of the general δ-GLMB formulation (cf. equation 2.26) is omitted as it is an empty set. While the LMB representation only describes one hypothesis for the predicted landmark map, this δ-GLMB representation now features multiple hypotheses as can be deduced from the sum over all subsets $\mathcal{I}_+ \in \mathcal{F}(\mathbb{L}_+^{(i)})$. Each of these hypotheses is only validated for one concrete realization of the RFS $\widetilde{\mathbf{M}}_+^{(i)}$. This is done by the generalized Kronecker delta function which only evaluates to 1 for the case where the set of labels $\mathcal{I}_+$ exactly matches the set of labels that are extracted from the predicted map state of the group $\widetilde{\mathbf{M}}_+^{(i)}$ by the projection $\mathscr{L}(\cdot)$. The respective weight for each set of track labels $\mathcal{I}_+$ used in equation (5.63) is

determined by:

$$w_{+,i}^{(\mathcal{I}_+)} = \prod_{j \in \mathbb{L}_+^{(i)}} \left(1 - r_+^{(j)}\right) \prod_{\ell \in \mathcal{I}_+} \frac{1_{\mathbb{L}_+^{(i)}}(\ell) r_+^{(\ell)}}{1 - r_+^{(\ell)}}, \tag{5.64}$$

where the second product basically considers the existence of landmarks and the first product considers the non-existence of landmarks (i.e., it cancels out with the denominator of the second product for those labels being in the subset $\mathcal{I}_+$ and only multiplies the probabilities of non-existence for those labels being in $\mathbb{L}_+^{(i)}$ but not in $\mathcal{I}_+$). Finally, the respective multi-object state of the map $\widetilde{\mathbf{M}}_+^{(i)}$ is described through

$$\widetilde{\mathbf{M}}_+^{(i)} = \left\{ \boldsymbol{m} : \mathscr{L}(\boldsymbol{m}) \in \mathbb{L}_+^{(i)} \; \forall \; \boldsymbol{m} \in \mathbf{M} \right\}.$$

For groups with relatively few elements – as it should be the case when doing SLAM with a limited set of meaningful features – all the hypotheses for all cardinalities $n = 0, 1, ..., |\mathbb{L}_+^{(i)}|$ of the δ-GLMB distributions can be generated in a brute-force way. However, if the number of tracks in the groups $|\mathbb{L}_+^{(i)}|$ gets higher one may resort to hypotheses sampling [Reu14], since the number of hypotheses per grouping is $2^{|\mathbb{L}_+^{(i)}|}$.

δ-GLMB Update For a given grouping $\mathcal{G}^{(i)}$, a track to measurement association hypothesis within that group is described by the unique mapping between track labels and measurements $\theta : \mathbb{L}_+^{(i)} \to \{0\} \cup \mathrm{Z}^{(i)}$ (with $\theta(i) = \theta(j) > 0 \implies i = j$, i.e., one measurement originates from at most one landmark). Using these association hypotheses, the posterior δ-GLMB distribution of the grouping according to the δ-GLMB update is given by (cf. [Reu14]):

$$\boldsymbol{\pi}_{\widetilde{\mathrm{M}}}^{(i)} = \Delta(\widetilde{\mathbf{M}}^{(i)}) \sum_{(\mathcal{I}_+, \theta) \in \mathcal{F}(\mathbb{L}_+^{(i)}) \times \Theta_{\mathcal{I}_+}^{(i)}} w^{(\mathcal{I}_+, \theta)}(\mathrm{Z}^{(i)}) \delta_{\mathcal{I}_+}(\mathscr{L}(\widetilde{\mathbf{M}}^{(i)})) \left[p^{(\theta)}(\cdot | \mathrm{Z}^{(i)}, \boldsymbol{x}_{0:k}) \right]^{\widetilde{\mathbf{M}}^{(i)}}, \tag{5.65}$$

where $\Theta_{\mathcal{I}_+}$ denotes the space that comprises all unique mappings between any possible subset $\mathcal{I}_+$ of the set of track labels $\mathbb{L}_+^{(i)}$ and the set of measurements: $\theta : \mathcal{I}_+ \to \{0\} \cup \mathrm{Z}^{(i)}$ (again $\theta(i) = \theta(j) > 0 \implies i = j$).
The posterior hypothesis weight for any given set of track labels $\mathcal{I}_+$ within this posterior δ-GLMB distribution is calculated by

$$w^{(\mathcal{I}_+, \theta)}(\mathrm{Z}^{(i)}) = \frac{\delta_{\theta^{-1}(\{0 \cup \mathrm{Z}^{(i)}\})}(\mathcal{I}_+) w_{+,i}^{(\mathcal{I}_+)} [\eta_{\mathrm{Z}^{(i)}, \boldsymbol{x}}^{(\theta)}]^{\mathcal{I}_+}}{\sum_{(\mathcal{I}_+, \theta) \in \mathcal{F}(\mathbb{L}_+^{(i)}) \times \Theta_{\mathcal{I}_+}^{(i)}} \delta_{\theta^{-1}(\{0 \cup \mathrm{Z}^{(i)}\})}(\mathcal{I}_+) w_{+,i}^{(\mathcal{I}_+)} [\eta_{\mathrm{Z}^{(i)}, \boldsymbol{x}}^{(\theta)}]^{\mathcal{I}_+}}, \tag{5.66}$$

taking into account the predicted weights $w_{+,i}^{(\mathcal{I}_+)}$ as calculated by using equation (5.64). Once again, the generalized Kronecker delta function is used such that this weight is only non-zero for the concrete realization where the set of track labels $\mathcal{I}_+$ matches those labels used in the association hypothesis (in this case obtained through the inverted association map of the measurements $\theta^{-1}(\{0 \cup \mathrm{Z}^{(i)}\})$). Further, the association likelihoods $\eta_{\mathrm{Z}^{(i)},\boldsymbol{x}}^{(\theta)}$ that have been calculated in the first step of the update process (cf. equation 5.53) are considered. Although, the association likelihoods have originally been calculated without the groupings (i.e., $\eta_{\mathrm{Z},\boldsymbol{x}}^{(\theta)}$) they may now be used again, as they are equivalent:

$$\eta_{\mathrm{Z}^{(i)},\boldsymbol{x}}^{(\theta)}(\ell) = \eta_{\mathrm{Z},\boldsymbol{x}}^{(\theta)}(\ell)$$

Finally, the denominator in equation 5.66 simply normalizes the weights.
For groups with a large number of elements, an enormous amount of hypotheses weights $w^{(\mathcal{I}_+,\theta)}(\mathrm{Z}^{(i)})$ have to be calculated. In this case, a truncation of the distribution (5.65) may be required in order to not violate computing time restrictions of the algorithm. As already proposed by Vo and Vo [VV13], a ranked assignment algorithm (e.g., Murty's algorithm [Mur68]) which only evaluates the most significant hypotheses, can be used to achieve this.
Another integral part of updating the N predicted δ-GLMB distributions, is determining the measurement updated posterior spatial distributions of the landmark tracks $p^{(\theta)}(\cdot|\mathrm{Z}^{(i)}, \boldsymbol{x}_{0:k})$ used in equation (5.65). In general they are given by (cf. [Reu14]):

$$p^{(\theta)}(\boldsymbol{m}, \ell \mid \mathrm{Z}_{0:k}, \boldsymbol{x}_{0:k}) = \frac{p_+(\boldsymbol{m}, \ell \mid \mathrm{Z}_{0:k-1}, \boldsymbol{x}_{0:k})\phi_{\mathrm{Z},\boldsymbol{x}}(\boldsymbol{m}, \ell; \theta)}{\eta_{\mathrm{Z},\boldsymbol{x}}^{(\theta)}(\ell)}, \tag{5.67}$$

where the generalized measurement likelihood $\phi_{\mathrm{Z},\boldsymbol{x}}(\boldsymbol{m}, \ell; \theta)$ of equation (5.52) is used again. Further, the normalization constant $\eta_{\mathrm{Z},\boldsymbol{x}}^{(\theta)}(\ell)$ is the association likelihood that has already been determined earlier and has also been used for determining the hypotheses weights (cf. equations (5.53) and (5.66)).
Within the Gaussian Mixture implementation, the posterior spatial distributions are given by

$$p^{(\theta)}(\boldsymbol{m}, \ell|\mathrm{Z}_{0:k}, \boldsymbol{x}_{0:k}) = \sum_{j=1}^{J_+^{(\ell)}} w^{(\ell,j,\theta)}(\mathrm{Z}_k)\mathcal{N}\left(\boldsymbol{m}; \boldsymbol{\mu}_k^{(\ell,j,\theta)}, \underline{\mathrm{P}}_k^{(\ell,j)}\right), \tag{5.68}$$

with the weights of each of the respective $j \in J_+^{(\ell)}$ Gaussian components being calculated by

$$w^{(\ell,j,\theta)}(\mathrm{Z}_k) = \frac{\frac{1}{\kappa(\boldsymbol{z}_{\theta(\ell)})} p_D w_+^{(\ell,j)} \mathcal{N}\left(\boldsymbol{z}_{\theta(\ell)}; \boldsymbol{z}_+^{(\ell,j)}, \underline{\mathrm{S}}^{(\ell,j)}\right)}{\eta_{\mathrm{Z},\boldsymbol{x}}^{(\theta)}(\ell)}. \tag{5.69}$$

Further, the necessary posterior mean values $\boldsymbol{\mu}_k^{(\ell,j,\theta)}$ and the covariances of the estimation error $\underline{\mathrm{P}}_k^{(\ell,j)}$ are obtained using the Kalman filter equations (EKF respectively, cf. equations (5.54)-(5.56)):

$$\boldsymbol{\mu}_k^{(\ell,j,\theta)}(\mathrm{Z}) = \boldsymbol{\mu}_+^{(\ell,j)} + \underline{\mathrm{K}}^{(\ell,j)} \left(\boldsymbol{z}_{\theta(\ell)} - \boldsymbol{z}_+^{(\ell,j)}\right), \tag{5.70}$$

$$\underline{\mathrm{K}}^{(\ell,j)} = \underline{\mathrm{P}}_+^{(\ell,j)} \underline{\mathrm{H}}^{\mathrm{T}} \left[\underline{\mathrm{S}}^{(\ell,j)}\right]^{-1}, \tag{5.71}$$

$$\underline{\mathrm{P}}_k^{(\ell,j)} = \underline{\mathrm{P}}_+^{(\ell,j)} - \underline{\mathrm{K}}^{(\ell,j)} \underline{\mathrm{S}}^{(\ell,j)} \left[\underline{\mathrm{K}}^{(\ell,j)}\right]^{\mathrm{T}}. \tag{5.72}$$

LMB Approximation Based on the updated δ-GLMB distribution, the respective updated LMB distribution has to be determined. To obtain the LMB representation of the tracks, the $i = 1, ..., N$ updated δ-GLMB distributions $\boldsymbol{\pi}_{\widetilde{\mathrm{M}}}^{(i)}$ have to be approximated according to

$$\boldsymbol{\pi}_{\widetilde{\mathrm{M}}}^{(i)} \approx \widetilde{\boldsymbol{\pi}}_{\mathrm{M}}^{(i)} = \left\{\left(r^{(\ell)}, p^{(\ell)}\right)\right\}_{\ell \in \mathbb{L}_+^{(i)}},$$

such that the LMB RFS matches the δ-GLMB RFS with respect to its unlabeled PHD (cf. [Reu14]). To achieve this, the existence probability $r^{(\ell)}$ has to be equal to the sum of the weights $w^{(\mathcal{I}_+,\theta)}(\mathrm{Z})$ of all those hypotheses in which the landmark track with the label ℓ exists, i.e.:

$$r^{(\ell)} = \sum_{(\mathcal{I}_+,\theta) \in \mathcal{F}(\mathbb{L}_+) \times \Theta_{\mathcal{I}_+}} w^{(\mathcal{I}_+,\theta)}(\mathrm{Z}) 1_{\mathcal{I}_+}(\ell) \tag{5.73}$$

Further, the spatial distribution $p^{(\ell)}$ of the track labeled ℓ is consequently given by

$$p^{(\ell)}(\boldsymbol{m}) = \frac{1}{r^{(\ell)}} \sum_{(\mathcal{I}_+,\theta) \in \mathcal{F}(\mathbb{L}_+) \times \Theta_{\mathcal{I}_+}} w^{(\mathcal{I}_+,\theta)}(\mathrm{Z}) 1_{\mathcal{I}_+}(\ell) p^{(\theta)}(\boldsymbol{m}, \ell \mid \mathrm{Z}_{0:k}, \boldsymbol{x}_{0:k}), \tag{5.74}$$

using the posterior spatial distributions $p^{(\theta)}(\boldsymbol{m}, \ell \mid \mathrm{Z}_{0:k}, \boldsymbol{x}_{0:k})$ that have previously been calculated (cf. equations 5.67 or respectively 5.68).

Unification and Track Extraction To approximate the full multi-object posterior by one LMB RFS, the N approximations $\widetilde{\boldsymbol{\pi}}^{(i)}$ are unified:

$$\boldsymbol{\pi}_{\mathrm{M}} \approx \widetilde{\boldsymbol{\pi}}_{\mathrm{M}} = \bigcup_{i=1}^{N} \left\{ \left(r^{(\ell)}, p^{(\ell)} \right) \right\}_{\ell \in \mathbb{L}_{+}^{(i)}} . \tag{5.75}$$

This LMB RFS finally constitutes the posterior distribution of the estimated landmark map conditioned on the trajectory of one particle.
However, whenever a filtering process is used, one does not always want to output each and every track, especially not those which have not been reliably validated by multiple update steps. This also holds true when doing SLAM: if the particle with the highest weight is used to represent the filter output, one does not usually want to have all newly created – and therefore possibly false positive – tracks displayed. The fact that the LMB directly facilitates an intuitive track extraction scheme comes in very handy. Since all tracks provide an existence probability $r^{(\ell)}$, one can simply apply a thresholding mechanism on that value, so that the set of extracted tracks is described by

$$\widehat{\mathbf{M}} = \{(\hat{\boldsymbol{m}}, \ell) | r^{(\ell)} > \vartheta\},$$

given the threshold ϑ. Further, also a hysteresis can be used in order to enable a continuous output of tracks that already have been confirmed earlier but have then been missed during the last few consecutive updates. To this end, the maximum existence probability $r_{\max}^{(\ell)}$ of any track is stored as well. If the current probability of existence $r^{(\ell)}$ is above a lower threshold ϑ_l and the existence probability has previously exceeded an upper threshold ϑ_u, the track will be part of the set of tracks that constitute the output of the filtering algorithm:

$$\widehat{\mathbf{M}} = \{(\hat{\boldsymbol{m}}, \ell) | r_{max}^{(\ell)} > \vartheta_u \wedge r^{(\ell)} > \vartheta_l\}. \tag{5.76}$$

Consequently, such a threshold can not only be used to prevent tracks from contributing to the output, but also to enable a pruning scheme, which deletes all tracks that have an existence probability below a further threshold ϑ_d.

5.4.2 Particle Weighting

As already mentioned before, the weight of a particle in an RB-RFS-SLAM filter can be determined by evaluating the likelihood $g(\mathrm{Z}_k \mid \mathrm{Z}_{0:k-1}, \boldsymbol{x}_{0:k})$, i.e., the particle weight is given by $\omega_k = g(\mathrm{Z}_k \mid \mathrm{Z}_{0:k-1}, \boldsymbol{x}_{0:k}) \cdot \omega_{k-1}$. As can be seen from the Bayes recursion of the map (5.32), this likelihood is the normalizing constant with respect to the map estimation. In contrast to the RB-PHD-SLAM filter, the RB-LMB-SLAM

filter offers a straightforward way to calculate the particle weights, due to the fact that "the update equation of the δ-GLMB filter provides an analytic solution for the normalization constant of the multi-object Bayes filter" [Reu14, p. 123]. The normalizing constant for each of the group updates following equation (5.65) is given by the denominator of the hypotheses weights (5.66):

$$d^{(i)}(\mathrm{Z}^{(i)}) = \sum_{(\mathcal{I}_+,\theta)\in\mathcal{F}(\mathbb{L}_+^{(i)})\times\Theta_{\mathcal{I}_+}^{(i)}} \delta_{\theta^{-1}(\{0\cup\mathrm{Z}^{(i)}\})}(\mathcal{I}_+) w_+^{(i)}(\mathcal{I}_+)[\eta_{\mathrm{Z}^{(i)},\boldsymbol{x}}^{(\theta)}]^{\mathcal{I}_+}. \tag{5.77}$$

These denominators are missing the clutter factor $\pi_C(\mathrm{Z}^{(i)})$ since it has been canceled out, due to its appearance in the nominator and denominator (see [VV13] for the derivation the δ-GLMB-update). Nevertheless, the clutter factor has to be considered in order to obtain the actual normalization constant. The full normalization constant of the LMB filter is therefore given by the product of these N group-normalization constants and the clutter factor $\pi_C(\mathrm{Z})$

$$d(\mathrm{Z}) = \pi_C(\mathrm{Z}) \cdot d^{(1)}(\mathrm{Z}^{(1)}) \cdots d^{(N)}(\mathrm{Z}^{(N)}), \tag{5.78}$$

where the clutter term is calculated using

$$\pi_C(\mathrm{Z}) = e^{-\lambda_c} \prod_{j=1}^{|\mathrm{Z}|} \kappa(\boldsymbol{z}). \tag{5.79}$$

Thus, the particle weights in RB-LMB-SLAM originate from a principled, analytical solution to the required likelihood and hence this approach is superior to the approximations that are used for finding the particle weights in RB-PHD-SLAM.
Finally, in RB-LMB-SLAM the estimated map and trajectory of the vehicle can also be determined by averaging over the maps and trajectories of the single particles. If the labels of the individual tracks are generated in such a way that each particle's LMB filter uses the same label for the same measurement (and therefore landmark) as all the other particles do (cf. Section 5.4.1), it is possible to calculate the mean map over all particles using the respective average existence probability and spatial distribution for each and every labeled track.
Another approach to calculate the average map, would be to approximate the LMB RFSs of the particles by their first order moments, i.e., their PHDs. Then, the average map can be calculated as already described in Section 5.3.2.

5.5 Evaluation

The simulation environment used for the evaluation of the RB-LMB-SLAM has been designed to allow for a comparison between the RB-LMB-SLAM and the reference implementation of the RB-PHD-SLAM published by Leung and Inostroza (version 1.0, publicly available at [LI14]). Hence, all required parameters (measurement model, measurement statistics, motion model,...) match those of the reference implementation as closely as possible. The details of the simulation environment are described in the following subsection, prior to the presentation of the simulation results. In [LIA14], the authors already demonstrated that RB-PHD-SLAM is superior to FastSLAM – in particular in the presence of a lot of clutter measurements. Hence, this evaluation suffices in comparing the proposed RB-LMB-SLAM to the RB-PHD-SLAM.

5.5.1 Setup

For this evaluation, an omni-directional vehicle with a sensor that produces range-bearing-measurements is simulated. The trajectory of the vehicle begins at $(0, 0)$ (cf. Figure 5.5 or Figure 5.6). The vehicle starts its journey in the direction of the positive x-axis and the positive y-axis. The winding path towards the end point near $(10.38, 0.74)$ comprises 3000 simulation steps and includes several bends. The course of the vehicle allows for revisiting mapped features. The ground truth map consists of a total of 36 landmarks spread randomly around the ground truth path of the vehicle.

Motion Model

The omni-directional vehicle used for this simulation moves in the xy-plane, thus its pose at time k is described by

$$\boldsymbol{x}_k = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}_k .$$

The predicted pose of the vehicle can be calculated on the basis of its pose at time $k-1$ using

$$\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k) + \boldsymbol{\delta} = f\left(\begin{bmatrix} x \\ y \\ \psi \end{bmatrix}_{k-1}, \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \psi \end{bmatrix}_k \right) + \boldsymbol{\delta},$$

with the additive white noise vector $\boldsymbol{\delta}$, the control input $\boldsymbol{u}_k$ and the function f which implements the translational and rotational displacements. Using the rotation matrix

$$\underline{\mathrm{C}} = \begin{bmatrix} \cos(\psi_{k-1}) & -\sin(\psi_{k-1}) \\ \sin(\psi_{k-1}) & \cos(\psi_{k-1}) \end{bmatrix},$$

the predicted position is thus given by

$$\begin{bmatrix} x \\ y \end{bmatrix}_k = \begin{bmatrix} x \\ y \end{bmatrix}_{k-1} + \underline{\mathrm{C}} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}_k + \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix}.$$

Finally, the predicted orientation is determined using

$$\psi_k = \psi_{k-1} + \Delta\psi_k + \delta_\psi.$$

Given this motion model, it would be possible that the vehicle moved in arbitrary ways. Nevertheless, the main direction of motion was set up to be towards the vehicle's positive x-axis for this simulation, i.e., the major contribution of the control input was chosen to be a forward motion.

Measurement Model

The simulated vehicle is equipped with a sensor which has a field of view of 360 deg and is able to measure objects within the range from 0.5 m to 2.5 m. The detection $\boldsymbol{z}^{(j)}$ of landmark $\boldsymbol{m}^{(i)} = \begin{bmatrix} x_{m^{(i)}} & y_{m^{(i)}} \end{bmatrix}^T$ at time step k is given by:

$$\boldsymbol{z}_k^{(j)} = \begin{bmatrix} r^{(j)} \\ \phi^{(j)} \end{bmatrix} = h(\boldsymbol{x}_k, \boldsymbol{m}^{(i)}) + \boldsymbol{e} = \begin{bmatrix} \sqrt{(x_{m^{(i)}} - x_k)^2 + (y_{m^{(i)}} - y_k)^2)} \\ \arctan\left(\frac{y_{m^{(i)}} - y_k}{x_{m^{(i)}} - x_k}\right) - \psi_k \end{bmatrix} + \boldsymbol{e}$$

using the polar coordinates $r^{(j)}, \phi^{(j)}$ and the measurement noise $\boldsymbol{e}$ with zero mean and covariance $\underline{\mathrm{R}}$.

5.5.2 Simulation Parameters

For the evaluation of the two RFS-SLAM approaches, a simulation over 3000 time steps which corresponds to 240 seconds at an update rate of 12.5 Hz, was conducted. Since the particle filter in general is a non-deterministic approach, 50 Monte-Carlo runs for each of the approaches were made. During these runs the number of particles was set to 200 and the resampling mechanism of the particle filter was triggered, whenever the effective number of particles fell below a threshold of 20 particles. The

simulated vehicle moved with an average speed of $0.11 \pm 0.05 \, \frac{\mathrm{m}}{\mathrm{s}}$ in x-direction and $0.00 \pm 0.018 \, \frac{\mathrm{m}}{\mathrm{s}}$ in y-direction through the map. The rotational rate of this randomly generated trajectory was on average $-0.1719 \pm 8.02 \, \frac{\mathrm{deg}}{\mathrm{s}}$. The noisy odometry based on this trajectory was created with a rate matching the measurement rate (i.e., 12.5 Hz) and assuming a standard deviation of 0.0022 m for the lateral as well as the longitudinal displacement and a standard deviation of 0.573 deg for the change in orientation.
The simulated measurements were created with a detection probability of $p_D = 0.7$ and a standard deviation of 0.07 m for the range measurements together with 2.86 deg for the bearing measurements, respectively. Further, clutter measurements were added. These false measurements were distributed uniformly over the measurement space. Their cardinality was Poisson distributed with an intensity of 0.8 m^{-2}. Given the area of the sensor's FOV, this resulted in an expected number of approximately 15 clutter measurements per scan.

5.5.3 Comparison of RB-PHD-SLAM and RB-LMB-SLAM

Two aspects are of great interest, when evaluating SLAM algorithms: First of all, as in the previous chapter, the accuracy of the estimated pose (or path), is of major importance. Secondly, the quality of the estimated map is at least as relevant – especially, when thinking about using SLAM to generate maps for highly automated vehicles.

Pose Estimation

The simulation reveals that both RB-RFS-SLAM approaches yield a very good localization accuracy (see Table 5.1 and Figures 5.5 and 5.6). However, the RB-LMB-SLAM outperforms the RB-PHD-SLAM in the estimation of the position. The mean values of the lateral and longitudinal errors are almost zero for the RB-LMB-SLAM. For the RB-PHD-SLAM they are also close to zero but approximately three times as much as with the RB-LMB-SLAM. Further, also the respective standard deviations are nearly twice as much as in the RB-LMB-SLAM runs. Thus, the RB-LMB-SLAM not only exhibits a reduced offset, but it also produces a more precise position estimation. The results for the orientation estimation are not as unambiguous. Again, when the LMB approach is used, the offset in the estimated orientation is low in comparison to the offset of the PHD approach (less than a quarter of that value). However, the standard deviation is higher when the RB-LMB-SLAM is used, i.e., the estimated orientation of the RB-LMB-SLAM deviates on average more from its mean value but this mean value is still closer to the true orientation.

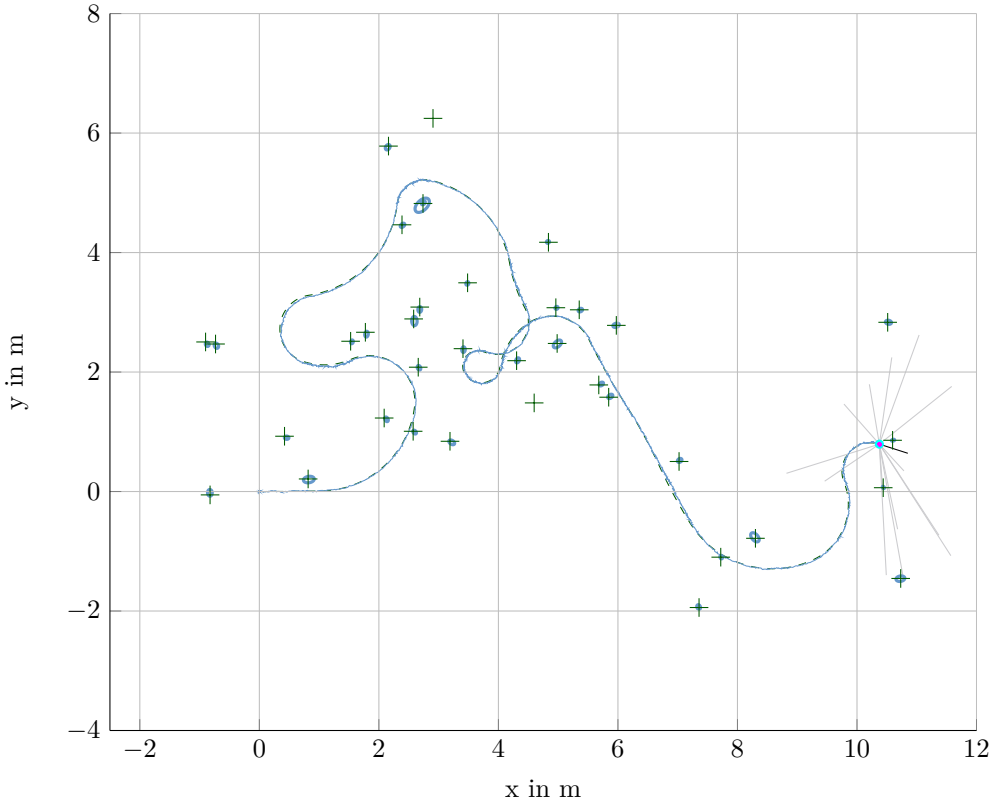


Figure 5.5: Result of one exemplary run of the RB-LMB-SLAM. The estimated trajectory and landmark positions are depicted by the blue curve and the blue ellipses respectively (only components with a existence probability $r > 0.5$ are shown). The ground truth path and landmark positions are colored in dark green. The pose estimate in the final time step is depicted in pink color. The gray lines which originate from the final ground truth position (turquoise dot) illustrate the simulated measurements at the final time step. The trajectory is estimated very precisely and the feature positions almost exactly match the ground truth landmarks except for those two landmarks that are not detected at all (near $(4.6, 1.5)$ and $(2.9, 6.2)$) Further, only one landmark is detected twice (near $(2.7, 4.9)$) and besides that zero false positives exist.

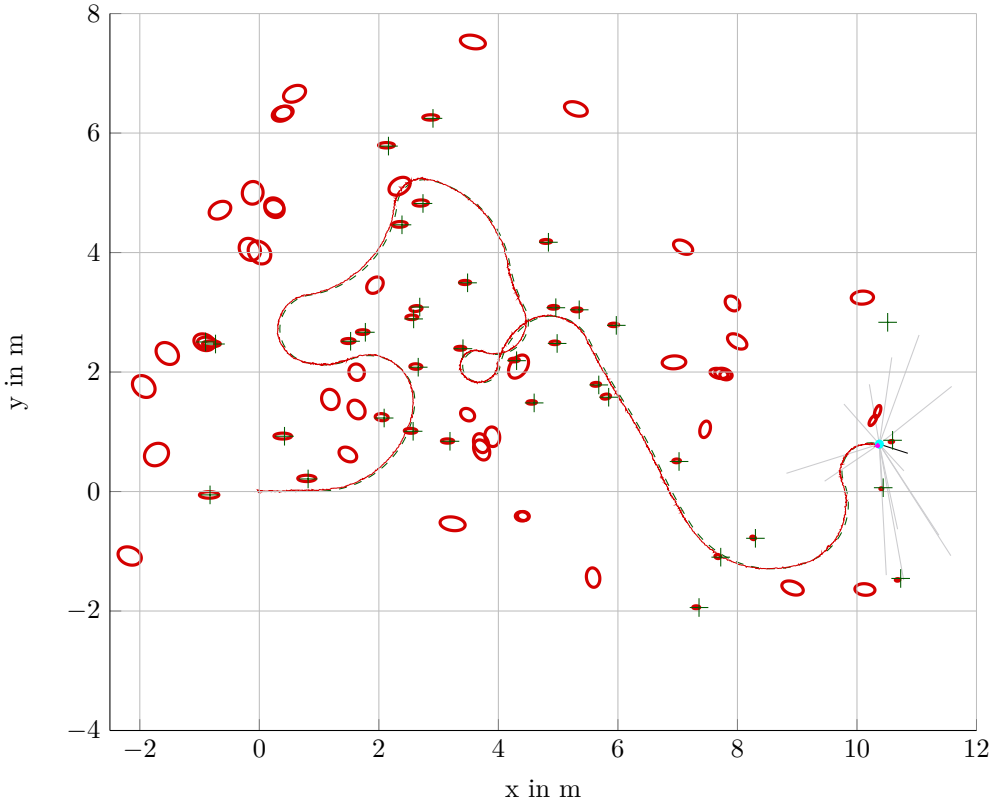


Figure 5.6: Result of one exemplary run of the RB-PHD-SLAM. The estimated trajectory and landmark positions are depicted by the red curve and the red ellipses respectively (only components of the GM with a weight above 0.5 are shown). The ground truth path and landmark positions are colored in dark green. The pose estimate in the final time step is depicted in pink color. The gray lines originating from the final ground truth position (turquoise dot) illustrate the simulated measurements at the final time step. The trajectory is estimated very exactly. The feature positions of those features originating from an actual landmark are very close to those ground truth positions, but many features actually originate from false measurements. Further, for two landmarks multiple features do exist (near $(-0.9, 2.5)$ and $(4.3, 2.2)$). Only one landmark was not detected at all (near $(10.5, 2.8)$).

Algorithm	lat_{err} in m	lon_{err} in m	ψ_{err} in deg
RB-LMB-SLAM	-0.0064 ± 0.0398	0.0084 ± 0.0436	-0.1585 ± 1.1258
RB-PHD-SLAM	0.0225 ± 0.0670	-0.0250 ± 0.0790	0.6941 ± 0.8024

Table 5.1: Average pose estimation error with standard deviation of the LMB- and PHD-SLAM

These results already give an indication of the high quality of the map estimate produced by the LMB filters, since the weights of the particles which are used to represent the trajectory, are calculated based on the map estimation. Hence, a good localization accuracy requires a good underlying map estimation.

Map Estimation

In order to asses the quality of the map estimation, the Optimal Sub-Pattern Assignment (OSPA) distance is used. The OSPA metric was introduced by Schumacher et al. in 2008, see [SVV08], and has since then become a standard tool to evaluate the performance of multi-object tracking algorithms. The OSPA metric does not only take the errors in state estimation (i.e., the estimated landmark positions) into consideration but also assesses the cardinality (i.e., the number of estimated landmarks). Therefore, it is the ideal choice to evaluate the quality of the map estimation. Further, the OSPA distance can be interpreted as per object error – per object means per estimated object if the number of estimated objects is higher than the number of existing objects and per true object otherwise (cf. [SVV08]).
The resulting OSPA distances of the RB-LMB-SLAM (blue) and the RB-PHD-SLAM (red) based on 50 Monte Carlo (MC) runs are depicted in Figure 5.7. It can clearly be seen that the OSPA distance of the RB-LMB-SLAM is almost always lower than the OSPA distance produced by the RB-PHD-SLAM. Furthermore, Figure 5.8 presents the two components contributing to the overall OSPA: the localization error (top) and the cardinality error (bottom). The results of the RB-PHD-SLAM jitter significantly – especially in the cardinality component. This is due to the filter constantly creating new tracks based on false measurements and then again discarding them. In comparison, the curve representing the RB-LMB-SLAM is clearly smoother, although it also oscillates around a mean value, which nevertheless is lower than the mean value of the RB-PHD-SLAM. In the localization component, the RB-LMB-SLAM clearly outperforms the RB-PHD-SLAM. The reason for this is, that the RB-PHD-SLAM often discards tracks of the real landmark and creates new tracks based on false measurements which are close enough to the true landmark position to be taken into consideration when calculating the OSPA distance.
However, the difference in the OSPA distance is not as big as one might expect

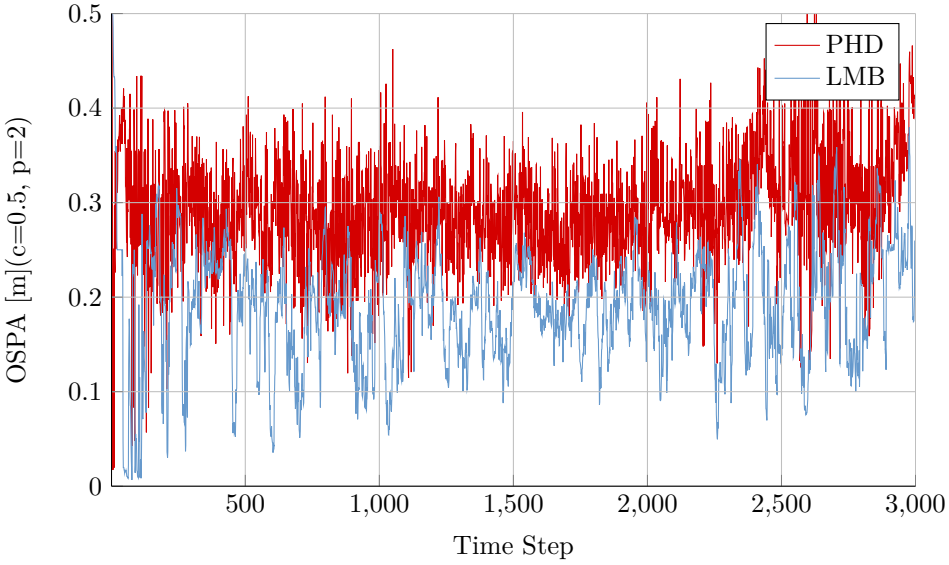


Figure 5.7: Average Optimal Sub-Pattern Assignment (OSPA) distance over 50 runs of RB-PHD-SLAM and RB-LMB-SLAM (cut-off $c = 0.5$, order $p = 2$).

considering the Figures 5.5 and 5.6. The reason for this is that only the landmarks and tracks that are actually in the FOV of the vehicle at any given time step are assessed.
The OSPA distances for all tracks and landmarks that have passed through the vehicle's FOV up until the last time step are given in Table 5.2. The respective OSPA distances clearly indicate that the RB-LMB-SLAM outperforms the RB-PHD-SLAM. Further, when considering the results at the last timestamp it becomes obvious that the RB-PHD-SLAM struggles with the high false measurement rate and produces many false positive features. The average total number of the estimated landmarks

Algorithm	*OSPA* in m	*Cardinality*
RB-LMB-SLAM	0.0901	33.02
RB-PHD-SLAM	0.3138	76.48

Table 5.2: Average OSPA-metric and estimated cardinality at the last time step calculated over the 50 Monte-Carlo runs (all tracks were considered, not only those in the FOV). The ground truth cardinality is 36.

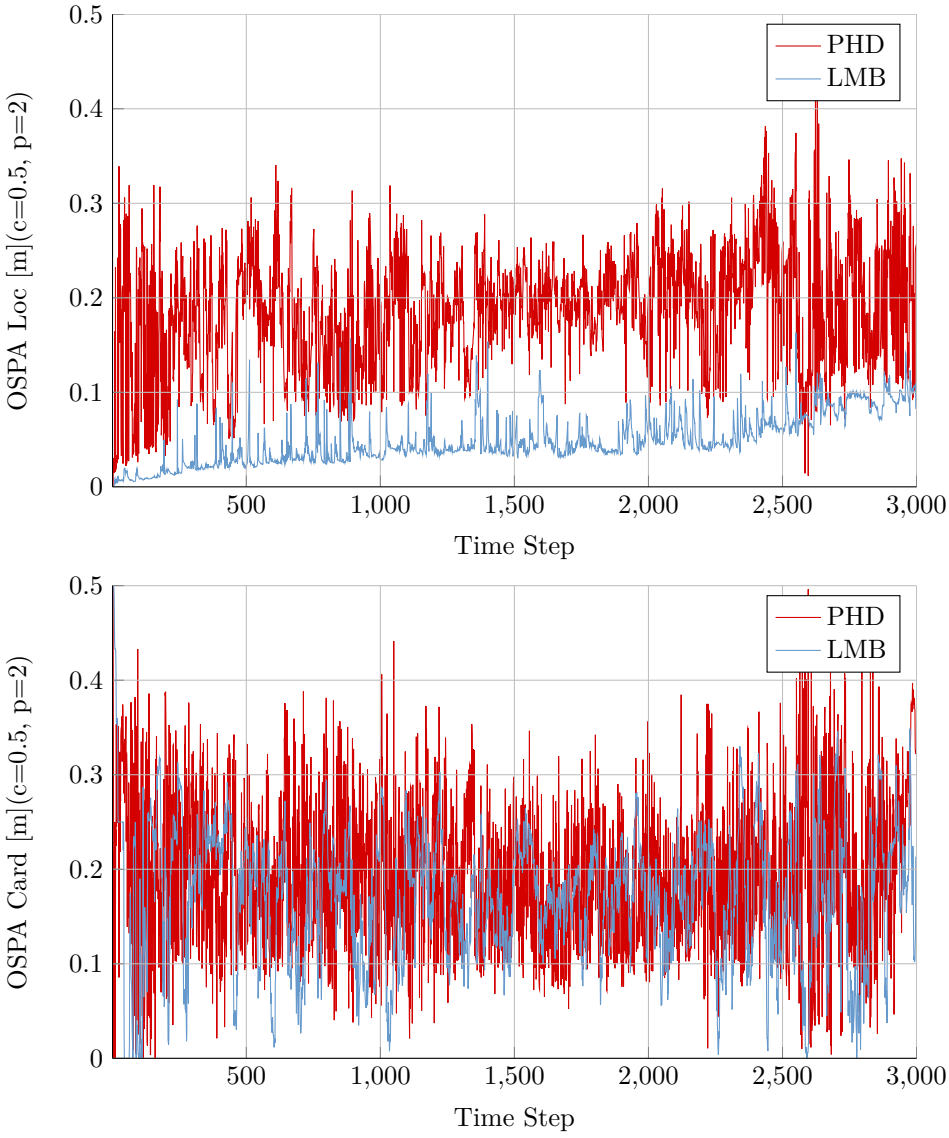


Figure 5.8: Average OSPA components over 50 runs of RB-PHD-SLAM and RB-LMB-SLAM (cut-off $c = 0.5$, order $p = 2$): in the localization component the error of the RB-LMB-SLAM is significantly lower (top); in the cardinality component the RB-LMB-SLAM produces a relatively smooth estimate in comparison to the RB-PHD-SLAM which jitters significantly.

is more than double the amount of the actual count when using RB-PHD-SLAM. The RB-LMB-SLAM is able to estimate the number of landmarks almost correctly, i.e., it only slightly underestimates the number of landmarks. This can also be seen in the exemplary runs presented in Figures 5.5 and 5.6.
The reason for the superior performance of RB-LMB-SLAM in this challenging scenario (high clutter rate, moderate probability of detection) lies in the exact update procedure of the LMB filter which is superior to the strong approximations made by the PHD filter. Further, the LMB representation (i.e., the explicit representation of the track existence and the track labels), is obviously more eligible than the first statistical moment approximation of the PHD filter which completely disregards cardinality. Therefore, the LMB filter is able to track the landmarks continuously opposed to constantly discarding the tracks and creating new tracks over and over again, as the PHD filter does in this scenario.
Overall, it can be stated, that the RB-LMB-SLAM is able to handle this challenging scenario very well and that it yields superior performance in comparison to RB-PHD-SLAM. Further, state of the art algorithms such as FastSLAM would not be able to yield a performance close to this, since they already struggle with lower clutter rates, cf. [LIA14].

5.6 Extension to Long-Term Map Storage

One possible future direction of research using the presented RB-LMB-SLAM algorithm is to facilitate its application in highly automated vehicles. However, if the presented RB-LMB-SLAM algorithm is to be used as a map creation (or map verification and updating) tool for highly automated vehicles, e.g., by using landmarks as presented in Chapter 4, it is no longer possible to keep all tracks in the main memory. Further, as a road vehicle usually moves relatively fast and on a particular route, it makes no sense to keep the tracks in the computer's main memory after they have left the FOV of the vehicle's sensors, since it most likely will take a long time until the vehicle again passes that same location. Hence, the question has to be answered how RB-LMB-SLAM and a map database can be interconnected. A possible answer is outlined in this section. In general, the problem has two sides: On the one hand, the landmarks estimated by the SLAM algorithm have to be integrated into the map database. On the other hand, the map data has also to be integrated as input to the SLAM algorithm. Figure 5.9 depicts the interactions between the SLAM component and a map database: The map database provides the landmarks that are currently in the FOV of the vehicle's sensors. This is done based on a geospatial query that uses the pose estimate produced by the SLAM algorithm. The SLAM algorithm again uses the mapped landmarks to improve its own landmark estimation performance. Further, the tracks of landmarks that have left the FOV (in the figure referred to as

"Completed Track") are fed back into the map database, i.e., an update is performed. To be able to detect situations in which landmarks that exist in the map database were not tracked by the SLAM algorithm, although they should have passed through the FOV, the map database update component also keeps track of the landmarks that were passed to the RB-LMB-SLAM and compares them to the produced tracks. This can be used to decrease the existence probability of such landmarks in the database.
As already discussed in Section 3.3, a database scheme that supports storing geospatial data along with a measure for the confidence in that data, can be used to represent a map of landmarks. Hence it also comprises everything necessary for the use with RB-LMB-SLAM. Although the confidence value is not absolutely necessary when using the map for localization only, it is crucial when such a map is used in combination with RB-LMB-SLAM.

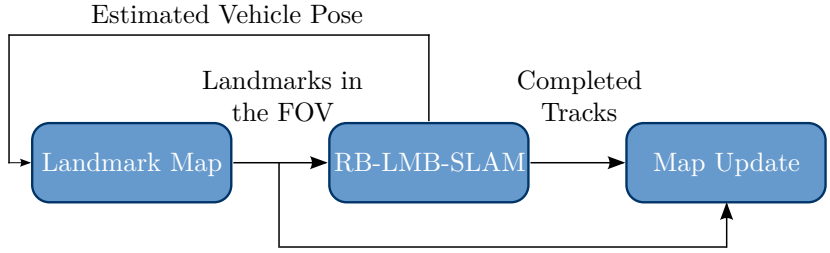


Figure 5.9: The interaction between the SLAM component and a database containing a map of landmarks.

5.6.1 The Map as Input to RB-LMB-SLAM

In the context of using SLAM for creating landmark maps for highly automated vehicles, one can imagine that such vehicles would be equipped with a basic predefined map which they then could augment with new landmark detections. Even if the vehicle started off with an empty map, at some point it would possess a map that is not empty. No matter what the source, this map should obviously be used as an input to the SLAM algorithm, to improve its performance. This can be done in RB-LMB-SLAM as follows. The algorithm uses a birth model to create new tracks during prediction, see Section 5.4.1. This birth model incorporates the measurements from the last time step, especially those that have not contributed significantly in updating existing tracks, to create a birth distribution $\boldsymbol{\pi}_B$. This birth distribution can be modified by also considering the landmarks from the map (i.e., the database) that should be in the sensor's FOV and that do not correspond to existing tracks. These landmarks should not add components to the birth distribution, i.e., they

should not create tracks without a supporting measurement. Instead, the existence probability of the tracks r_B and the spatial distribution p_B of components of the birth distribution should be modified according to their association likelihoods with the landmarks from the database. For example, if a landmark with a high probability of existence and a very low spatial uncertainty existed in the map and a measurement was made very close to that location, one could create a new track with a high initial probability of existence r_B and the spatial distribution p_B could be shifted towards the position of the landmark in the map. Using the described procedure, one can decrease the necessary number of detections to create new tracks, if they are likely to originate from a landmark that already exists in the map. Further, one can allow for more missed detections (especially during the first frames after track creation) since these tracks posses a higher initial existence probability. Finally, the position estimate of tracks which are created using landmarks from the map is also usually better, since it is supported by information that has been gathered through multiple passings of the respective objects.

5.6.2 Updating Maps with RB-LMB-SLAM

Obviously, in order to keep the map in the database up to date, the output of the SLAM algorithm has to be somehow fused with the data already present in the database. Basically, there are three possibilities of how the map has to be updated:

- A track does not correspond to any landmark, i.e., a new landmark has to be created in the database.
- A track possesses a significant likelihood of corresponding to one or more landmarks that have also been passed to the SLAM algorithm, i.e., these landmarks have to be updated.
- A landmark that had been passed to the SLAM algorithm has not been verified by the output of the SLAM algorithm, i.e., the existence probability of that landmark in the database has to be reduced.

The realization of the first case is straightforward – if no corresponding landmark is present in the database, a new one has to be created. The position and spatial uncertainty of this new landmark can be extracted from the spatial distribution $p^{(\ell)}$ of the track labeled ℓ. On the other hand, the existence probability in the database p_{ex} should not be a copy of the existence probability of the track $r^{(\ell)}$, although all tracks posses one, as they are represented by an LMB-RFS, cf. Section 5.4.1. The reason for this is that the information stored in the database should represent "long-term knowledge" and the output of the RB-LMB-SLAM can only represent

"short-term knowledge". Thus, setting or updating existence probabilities in the database should resemble a low-pass filter behavior.
For example, a landmark that was found by the SLAM algorithm with a high existence probability will not necessarily be there the next time the vehicle passes, which again a high p_{ex} would imply. Hence, the initial p_{ex} should be set to a relatively low value (which still may depend on the tracks existence probability).
The second case, where existing landmarks have to be updated by the output of the SLAM algorithm can be processed as follows: First, those landmarks which have been passed to the SLAM component (cf. Figure 5.9) and which have a significant spatial likelihood of corresponding to the track labeled ℓ are selected. Then their positions and existence probabilities are updated using the data from the track ($p^{(\ell)}$ and $r^{(\ell)}$) and the learning rate η. This learning rate is a function of the likelihood of the track corresponding to the landmark being updated and of the number of times this landmark has already been updated, i.e., the more likely a track corresponds to the landmark, the higher the learning rate and the older the landmark (high number of updates) the lower the learning rate. A minimum value for the learning rate is also necessary to always have the ability to react to changes in the environment. Furthermore, when the positions of landmarks are updated in the database, it is necessary to also check if a merge of landmarks is required.
Finally, the last case, which is the main reason why the map update component has to have knowledge about the landmarks that have been passed to the RB-LMB-SLAM (as depicted in Figure 5.9): Landmarks have been passed to the SLAM algorithm, but no corresponding track has been created. In this case the existence probabilities of those landmarks have to be updated with a track existence probability $r = 0.0$, again using the learning rate (neglecting the spatial likelihood). After this update, it has to be checked if the existence probability of the landmarks in the database still is above a certain threshold. If it has fallen below the threshold, the respective landmarks have to be deleted from the database.

Using an approach like the one described above could make it possible to generate and update large maps of frequently driven routes with a standard road vehicle based on RB-LMB-SLAM. Further, if a central database was used, multiple vehicles could contribute to this database whenever they were connected to the Internet. This would facilitate having reliable, up to date landmark maps of wide areas of the road network.

Chapter 6

Conclusions and Future Directions

The central contributions of this work are: A particle filter based localization algorithm for highly automated vehicles and a novel approach to SLAM. The localization algorithm predominantly uses MSER features extracted from the data of two entirely different sensors that complement each other: The first source are camera images in which the MSER algorithm typically detects lane markings as valid features. The second source are occupancy grid maps created using measurements from a laser scanner. The features that are extracted from the grid maps are arbitrary stationary objects that satisfy certain size constraints. The particle filter uses both data sources whenever they are available, but it does not necessarily require data from both sensors. This facilitates a good pose estimation even when one of the sensors does not provide sensible measurements for a certain amount of time. Moreover, the weight function that is used to assess the pose hypotheses represented by the particles makes use of Finite Set Statistics. Hence, the presented method constitutes a novel, mathematically sound approach to implementing this crucial part of Monte Carlo Localization.
Furthermore, the simplicity and reliability of the features allows for fast and efficient mapping – this can be done by simply driving the to-be-mapped-route once and localizing the vehicle with a reference system.
The evaluations presented in Section 4.5 reveal that the proposed approach makes it possible to estimate the pose of the vehicle with high accuracy on rural and urban roads. Furthermore, the localization algorithm has been applied successfully while driving autonomously during multiple public presentations.
However, there are still a few desirable modifications that could be integrated to increase the performance and broaden the range of applications of the localization algorithm in the future. First of all, integrating an estimate of the vehicle's pitch angle would improve the accuracy of the grid map and would further allow for a more precise localization of the image features. Secondly, in order to be able to localize the vehicle on highways, additional features are required. On highways, the laser beams are reflected by other traffic participants and guarding rails almost

exclusively. Hence, no MSER features can be detected in the respective occupancy grid maps which implicates that the algorithm has to rely solely on the camera. Another possible modification of the localization scheme would be to replace the laser scanner with a latest generation high-resolution automotive radar. This would improve the localization performance in adverse weather conditions significantly if the resolution of these new radars was high enough.

The second major contribution of this thesis is the Rao-Blackwellized Labeled Multi-Bernoulli Simultaneous Localization And Mapping algorithm. It represents a novel approach to solving the simultaneous localization and mapping problem which is motivated by recent advances in multi-object tracking. Like other SLAM approaches (e.g., FastSLAM) it is based upon a particle filter where each particle estimates its own map. However, it is the first SLAM algorithm to use the novel LMB filter for that purpose. As the RB-PHD-SLAM approach it also relies on RFS theory. Therefore, it also directly facilitates an integrated handling of measurement statistics, i.e., false alarm rates and detection probability. Nevertheless, it is superior to RB-PHD-SLAM since it does not require any further approximations to determine the particle weights.

With the simulation based evaluation described in Section 5.5, it is demonstrated that, in scenarios with high clutter rates, the RB-LMB-SLAM outperforms the RB-PHD-SLAM, which is especially well known for its good performance in this kind of scenario. Admittedly, as the algorithm has only been evaluated in simulations, it would be interesting to evaluate it with real data. To bring the algorithm to its real purpose, i.e., localizing a road vehicle and creating a map of landmarks online while driving, a real-time capable implementation would be required. As the algorithm is very well suited to be executed in parallel, a GPU implementation would be desirable.

Appendix A

Supervised Theses

Bach, Martin: *Fahrstreifendetektion für ein autonomes Modellfahrzeug unter Verwendung eines Grafikprozessors*. Bachelorarbeit, Universität Ulm, 2013.

Bonfert, Stefan: *Vergleich von Neuronalen Netzen und Hidden Markov Modellen zur Detektion von rechts vor links Verstößen*. Bachelorarbeit, Universität Ulm, 2012.

Debout, Marcel: *Detektion und Lokalisierung von Hindernissen für ein autonomes Modellfahrzeug unter Verwendung der ASUS XtionPro Tiefenbildkamera*. Bachelorarbeit, Universität Ulm, 2012.

Fischer, Alexander: *Detektion mehrerer Fahrstreifen zur Fahrbahnrepräsentation unter Verwendung von Partikelsets*. Diplomarbeit, Universität Ulm, 2012.

Hoch, Philipp: *Evaluation of Classifiers for Real Time Traffic Sign Recognition Systems*. Bachelor Thesis, Ulm University, 2012.

Wallner, Paul: *Multi-Sensor Fahrstreifentracking unter Verwendung von Random Finite Sets*. Diplomarbeit, Universität Ulm, 2013.

Witte, Thomas: *Nutzung eines Monokamerasystems zur Lokalisierung auf Basis von Verkehrszeichen*. Bachelorarbeit, Universität Ulm, 2014.

Appendix B

Publications

Parts of this thesis were pre-released in the author's articles and conference papers.

Deusch, Hendrik; Graf, Regine; Fritzsche, Martin; and Dietmayer, Klaus: *The Use of Spatial Memory for Advanced Driver Assistance Systems: Preventing Stationary ACC False Alarms.* In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 1291–1296, 2013.

Deusch, Hendrik; Nuss, Dominik; Konrad, Paul; et al.: *Improving Localization in Digital Maps with Grid Maps.* In: *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems*, pages 1522–1527, 2013.

Deusch, Hendrik; Reuter, Stephan; and Dietmayer, Klaus: *The Labeled Multi-Bernoulli SLAM Filter.* In: *IEEE Signal Processing Letters*, volume 22, no. 10, pages 1561–1565, 2015.

Deusch, Hendrik; Wiest, Jürgen; Reuter, Stephan; et al.: *A Random Finite Set Approach to Multiple Lane Detection.* In: *Proceedings of the 15th International IEEE Conference on Intelligent Transportation Systems*, pages 270–275, 2012.

Deusch, Hendrik; Wiest, Jürgen; Reuter, Stephan; et al.: *Multi-Sensor Self-Localization Based on Maximally Stable Extremal Regions.* In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 555–560, 2014.

Graf, Regine; Deusch, Hendrik; Fritzsche, Martin; and Dietmayer, Klaus: *A Learning Concept for Behavior Prediction in Traffic Situations.* In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 672–677, 2013.

Graf, Regine; Deusch, Hendrik; Seeliger, Florian; Fritzsche, Martin; and Dietmayer, Klaus: *A Learning Concept for Behavior Prediction at Intersections.* In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 939–945, 2014.

Kunz, Felix; Nuss, Dominik; Wiest, Juergen; et al.: *Autonomous Driving at Ulm University: A Modular, Robust, and Sensor-Independent Fusion Approach.* In: *Proceedings of the IEEE Intelligent Vehicles Symposium.* Accepted for publication, 2015.

Nuss, Dominik; Wilking, Benjamin; Wiest, Jürgen; et al.: *Decision-Free True Positive Estimation with Grid Maps for Multi-Object Tracking.* In: *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems*, pages 28–34, The Hague, Netherlands, 2013.

Wiest, Jürgen; Deusch, Hendrik; Nuss, Dominik; et al.: *Localization Based on Region Descriptors in Grid Maps.* In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 793–799, 2014.

Acronyms

δ-GLMB	Delta-Generalized Labeled Multi-Bernoulli	14
ABS	Anti-lock Blocking System	21
ACC	Autonomous Cruise Control	23
ADAS	Advanced Driver Assistance System	22
ADMA	Automotive Dynamic Motion Analyzer	24
ADTF	Automotive Data and Time-Triggered Framework	25
AMR	Anisotropic Magneto Resistive	21
CAN	Controller Area Network	19
CCD	Charge-Coupled Device	23
CPU	Central Processing Unit	25, 26
CTRV	Constant Turn Rate and Velocity	52
DGPS	Differential Global Positioning System	20
EKF	Extended Kalman Filter	45
ESC	Electronic Stability Control	21
FastSLAM	Factored solution to SLAM	78
FISST	Finite Set Statistics	3, 5, 10, 81
FMCW	Frequency-Modulated Continuous-Wave	24
FOV	Field Of View	16
GLMB	Generalized Labeled Multi-Bernoulli	13
GLONASS	GLObalnaja NAwigazionnaja Sputnikowaja Sistema	24
GM	Gaussian Mixture	18
GNSS	Global Navigation Satellite System	2
GPS	Global Positioning System	2
GPU	Graphics Processing Units	96
GUI	Graphical User Interface	27

i.i.d.	Independent and Identically Distributed	87
IMU	Inertial Measurement Unit	24
LIDAR	LIght Detection And Ranging	21
LMB	Labeled Multi-Bernoulli	3, 12, 74
MC	Monte Carlo	108
MCL	Monte Carlo Localization	iii, 3
MEMS	Micro-Electro-Mechanical System	21
MHD	Mahalanobis Distance	45
MSER	Maximally Stable Extremal Regions	41
NTP	Network Time Protocol	26
OEM	Original Equipment Manufacturer	iii
OGC	Open Geospatial Consortium	32
ORDBMS	Object Relational DataBase Management Systems	32
OSPA	Optimal Sub-Pattern Assignment	108, 109
PDF	Probability Density Function	11
PHD	Probability Hypothesis Density	3, 17, 74
PPS	Pulse Per Second	26
RAM	Random Access Memory	25, 26
RB	Rao-Blackwellized	86
RB-LMB-SLAM	Rao-Blackwellized Labeled Multi-Bernoulli Simultaneous Localization And Mapping	3
RB-PHD-SLAM	Rao-Blackwellized Probability Hypothesis Density Simultaneous Localization And Mapping	iv, 3
RB-RFS-SLAM	Rao-Blackwellized Random Finite Set based Simultaneous Localization And Mapping	5
RFS	Random Finite Set	iii, 3, 74
ROI	Region of Interest	47
RTK	Real Time Kinematic (System)	20
SLAM	Simultaneous Localization And Mapping	iii, vi, 3, 73
SQL	Structured Query Language	32
SURF	Speeded Up Robust Features	39
ToF	Time of Flight	21

USB	Universal Serial Bus	24
UTM	Universal Transverse Mercator	31
WGS84	World Geodetic System 1984	30

List of Symbols

Notations

a	a scalar value
$\boldsymbol{a}$	a vector
$\underline{\mathrm{A}}$	a matrix
$\underline{\mathrm{A}}^{-1}$	inverse of a matrix
$\underline{\mathrm{A}}^{T}$	transpose of a matrix
A	a random finite set
$\mathbf{A}$	a labeled random finite set
$\mathcal{A}$	a set
$\lvert\mathrm{A}\rvert$	cardinality of a random finite set
$\lvert\boldsymbol{a}\rvert$	dimension of a vector
$1_{\mathrm{Y}}(\mathrm{X})$	inclusion function
$\delta_{\mathrm{A}}(\mathrm{B})$	generalized Kronecker delta function
$\langle b, c\rangle$	inner product of the functions b and c
h^{A}	multi-object exponential

Subscripts

B	variable or function referring to new born objects
C	variable or function referring to clutter
k	time step k
M	variable or function referring to the map
Z	variable or function referring to the measurements
$+$	variable or function referring to a prediction
$0:k$	all elements up to time step k
$\boldsymbol{x}$	variable or function referring to the pose

Vectors and Scalars of Particular Importance

e	a UTM easting coordinate
ℓ	a track label

λ	expected value of the Poisson distribution
$\boldsymbol{m}$	an element of the map (i.e., a landmark)
$\bar{\boldsymbol{m}}$	an element of the map transformed to the vehicle coordinate system
$\boldsymbol{\Omega}$	vector of landmarks
n	a UTM northing coordinate
n	number of elements in a set or vector (referring to the entity described by its index)
ν_P	number of particles
$\tilde{\nu}_P^{Eff}$	approximate number of effective particles
$\boldsymbol{s}$	the state vector in EKF-SLAM
$\boldsymbol{u}$	the control input
v	velocity of the vehicle
w	a weight in general (e.g., of a hypothesis or a mixture component)
ω	the weight of a particle specifically
$\boldsymbol{x}^*$	the initial pose of the vehicle used for SLAM
$\boldsymbol{x}$	the (2D) pose of the vehicle
ψ	orientation of the vehicle
$\dot{\psi}$	yaw rate of the vehicle
$\boldsymbol{z}$	a measurement
$\boldsymbol{\Upsilon}$	vector of measurements

Spaces

$\mathbb{C}$	an index space
$\mathcal{F}(\mathbb{X})$	finite subsets of the space $\mathbb{X}$
$\mathcal{F}_n(\mathbb{X})$	finite subsets of the space $\mathbb{X}$ with cardinality n
$\mathbb{M}$	the space of landmark states
$\mathbb{L}$	label space of existing objects
$\mathscr{L}$	projection from $\mathbb{X} \times \mathbb{L}$ to $\mathbb{L}$
Θ	space of association hypotheses
Ξ	a discrete space
$\mathbb{X}$	the state space
$\mathbb{Z}$	the measurement space

Specific Probabilities

p_D	detection probability
q_D	probability of a missed detection

p_S	survival (or persistence) probability
r	existence probability

Densities

$b(\cdot)$	probability hypothesis density of the birth RFS
$\kappa(\cdot)$	a probability hypothesis density of the clutter process
$f(\cdot)$	single-object or multi-object Markov transition density (referring to the entity described by its index)
$\boldsymbol{\pi}(\cdot)$	labeled multi-object probability density
$\pi(\cdot)$	multi-object probability density
$p(\cdot)$	a probability density
$v(\cdot)$	probability hypothesis density

Sets

B	the birth RFS
C	the RFS of clutter measurements
$\mathrm{D}(\boldsymbol{m}, \boldsymbol{x})$	the Bernoulli-RFS of a feature measurement generated by a landmark
M	the RFS of landmarks
Z	the RFS of measurements
$\mathcal{I}$	a set of track labels (within a hypothesis)
$\mathcal{L}$	a set of track labels
$\mathcal{M}$	the set of landmarks
$\bar{\mathcal{M}}^{FoV}(\boldsymbol{x}_k^{(i)})$	the set of landmarks being in the FOV of the vehicle transformed to the vehicle's coordinate system
$\mathcal{P}$	the set of particles
$\mathcal{Z}$	the set of measurements
$\Delta(\mathbf{X})$	distinct label indicator

Likelihoods

$\eta_{\mathrm{Z},\boldsymbol{x}}^{(\theta)}(\ell)$	likelihood for the assignment of track label ℓ to measurement $\theta(\ell)$
$\phi_{\mathrm{Z},\boldsymbol{x}}(\boldsymbol{m}, \ell; \theta)$	the generalized measurement likelihood
$g(\boldsymbol{z} \mid \boldsymbol{m}, \boldsymbol{x})$	the measurement likelihood given the landmarks position and the vehicle pose
$\tilde{g}(\mathrm{Z}_k \mid \bar{\mathrm{M}}^{FoV}(\boldsymbol{x}_k^{(i)}))$	the approximated likelihood for the set of measurements and the transformed set of landmarks in the vehicle's FOV

$g(\mathcal{Z}_k \vert \boldsymbol{x}_k^{(i)})$	the likelihood for the measurements $\mathcal{Z}_k$ given the state $\boldsymbol{x}_k^{(i)}$

Other Symbols

$\underline{\mathrm{A}}$	the association matrix for vector-based SLAM
$\beta(\boldsymbol{z} \mid \boldsymbol{m})$	cost for the association of a landmark and a measurement
ζ^{cam}	the coordinate system of a camera
ζ^{img}	the image coordinate system
ζ^{ls}	the coordinate system of a laser scanner
ζ^{r}	the coordinate system of a radar
ζ^{utm}	the UTM coordinate system
ζ^{veh}	the coordinate system of the vehicle
γ_i	a grid cell
$\mathcal{G}$	a group for the LMB update
$\mathcal{N}(\cdot; \mu, \underline{\mathrm{P}})$	Normal distribution with mean μ and covariance $\underline{\mathrm{P}}$
$\underline{\mathrm{S}}$	the innovation covariance matrix
θ	an association hypothesis
Θ^*	the most likely association hypotheses
$\mathcal{U}(\cdot)$	uniform distribution

Bibliography

[AMGC02] Arulampalam, M.S.; Maskell, S.; Gordon, N.; and Clapp, T.: *A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking*. In: *IEEE Transactions on Signal Processing*, volume 50, no. 2, pages 174–188, 2002.

[AMV13] Adams, Martin; Mullane, John; and Vo, Ba-Ngu: *Circumventing the Feature Association Problem in SLAM*. In: *IEEE Intelligent Transportation Systems Magazine*, volume 5, no. 3, pages 40–58, 2013.

[AVMM14] Adams, M.; Vo, B.; Mahler, R.; and Mullane, J.: *SLAM Gets a PHD: New Concepts in Map Estimation*. In: *IEEE Robotics Automation Magazine*, volume 21, no. 2, pages 26–37, 2014.

[BBD+13] Broggi, Alberto; Buzzoni, Michele; Debattisti, Stefano; et al.: *Extensive Tests of Autonomous Driving Technologies*. In: *IEEE Transactions on Intelligent Transportation Systems*, volume 14, no. 3, pages 1403–1415, 2013.

[BD06] Bailey, Tim and Durrant-Whyte, Hugh: *Simultaneous localization and mapping (SLAM): Part II*. In: *IEEE Robotics & Automation Magazine*, volume 13, no. 3, pages 108–117, 2006.

[BHK11] Badino, H.; Huber, D.; and Kanade, T.: *Visual topometric localization*. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 794–799, 2011.

[BP99] Blackman, S. and Popoli, R.: *Design and Analysis of Modern Tracking Systems*. Artech House Publishers, 1999.

[BSH+03] Bronstein, I.N.; Semendjajew, K.A.; Hackbusch, W.; Schwarz, H.R.; and Zeidler, E.: *Teubner-Taschenbuch der Mathematik*. Ed. by Zeidler, Prof. Dr. E. 2nd Edition, Teubner, 2003.

[CSRL01] Cormen, Thomas H.; Stein, Clifford; Rivest, Ronald L.; and Leiserson, Charles E.: *Introduction to Algorithms*. 2nd Edition, McGraw-Hill Higher Education, 2001.

[DB06] Durrant-Whyte, Hugh and Bailey, Tim: *Simultaneous localization and mapping: part I*. In: *IEEE Robotics & Automation Magazine*, volume 13, no. 2, pages 99–110, 2006.

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

[DFBT99] Dellaert, F.; Fox, D.; Burgard, W.; and Thrun, S.: *Monte Carlo localization for mobile robots*. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. Volume 2, pages 1322–1328, 1999.

[DFMR00] Doucet, Arnaud; Freitas, Nando de; Murphy, Kevin P.; and Russell, Stuart J.: *Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks*. In: *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 176–183, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[DGA00] Doucet, Arnaud; Godsill, Simon; and Andrieu, Christophe: *On sequential Monte Carlo sampling methods for Bayesian filtering*. In: *Statistics and Computing*, volume 10, no. 3, pages 197–208, 2000.

[DMC90] Dickmanns, E.D.; Mysliwetz, B.; and Christians, T.: *An integrated spatio-temporal approach to automatic visual guidance of autonomous vehicles*. In: *IEEE Transactions on Systems, Man and Cybernetics*, volume 20, no. 6, pages 1273–1284, 1990.

[DNK+13] Deusch, Hendrik; Nuss, Dominik; Konrad, Paul; et al.: *Improving Localization in Digital Maps with Grid Maps*. In: *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems*, pages 1522–1527, 2013.

[DWR+14] Deusch, Hendrik; Wiest, Jürgen; Reuter, Stephan; et al.: *Multi-Sensor Self-Localization Based on Maximally Stable Extremal Regions*. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 555–560, 2014.

[Ebe12] Eberhardt, Elvira: *Photograph of the Test Vehicle*. Institute of Measurement, Control and Microtechnology, Ulm University, Ulm, Germany, 2012.

[Elf89] Elfes, A.: *Using occupancy grids for mobile robot perception and navigation*. In: *Computer*, volume 22, no. 6, pages 46–57, 1989.

[EWB05] Erdinc, O.; Willett, P.; and Bar-Shalom, Y.: *Probability hypothesis density filter for multitarget multisensor tracking*. In: *Proceedings of the 8th International Conference on Information Fusion*, pages 1–8, 2005.

[For07] Forssen, P.-E.: *Maximally Stable Colour Regions for Recognition and Matching*. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[GLL+15] Gehrels, Barend; Lalande, Bruno; Loskot, Mateusz; Wulkiewicz, Adam; and Karavelas, Menelaos: *Boost Geometry*. Visited March 14th 2015. 2015. URL: `www.boost.org/doc/libs/1_57_0/libs/geometry`.

[GLM+12] Geiger, A.; Lauer, M.; Moosmann, F.; et al.: *Team AnnieWAY's Entry to the 2011 Grand Cooperative Driving Challenge*. In: *IEEE Transactions on Intelligent Transportation Systems*, volume 13, no. 3, pages 1008–1017, 2012.

[GLW+13] Gabb, Michael; Löhlein, Otto; Wagner, Raimar; et al.: *High-Performance On-Road Vehicle Detection in Monocular Images*. In: *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems*, pages 336–341, 2013.

[GMN97] Goodman, I.R.; Mahler, R.P.; and Nguyen, H.T.: *Mathematics of Data Fusion*. From the series *Theory and decision library: Series B : Mathematical and Statistical Methods*. Springer, 1997.

[GSS93] Gordon, N.J.; Salmond, D.J.; and Smith, A. F M: *Novel approach to nonlinear/non-Gaussian Bayesian state estimation*. In: *IEE Proceedings F (Radar and Signal Processing)*, volume 140, no. 2, pages 107–113, 1993.

[HO11] Hsu, L. and Obe, R.: *PostGIS in Action*. From the series *Manning Pubs Co Series*. 1st editon, Manning Publications, 2011.

[HS97] Heikkilä, Janne and Silven, Olli: *A Four-step Camera Calibration Procedure with Implicit Image Correction*. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, IEEE Computer Society, Washington, DC, USA, 1997.

[ILA14] Inostroza, Felipe; Leung, Keith Y. K; and Adams, Martin: *Semantic Feature Detection Statistics in Set Based Simultaneous Localization and Mapping*. In: *Proceedings of the 17th International Conference on Information Fusion*, 2014.

[Kal60] Kalman, Rudolph Emil: *A New Approach to Linear Filtering and Prediction Problems*. In: *Transactions of the ASME - Journal of Basic Engineering*, volume 82, no. Series D, pages 35–45, 1960.

[KLW94] Kong, Augustine; Liu, Jun S.; and Wong, Wing H.: *Sequential Imputations and Bayesian Missing Data Problems*. In: *Journal of the American Statistical Association*, volume 89, no. 425, pages 278–288, 1994.

[KND12] Konrad, Marcus; Nuss, Dominik; and Dietmayer, Klaus: *Localization in Digital Maps for Road Course Estimation using Grid Maps*. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 87–92, 2012.

[Kry07] Krymmel, Jan: *The UTM zones*. 2007. URL: `http://upload.wikimedia.org/wikipedia/commons/e/ed/Utm-zones.jpg`.

[KSD10] Konrad, Marcus; Szczot, Magdalena; and Dietmayer, Klaus: *Road Course Estimation in Occupancy Grids*. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 412–417, 2010.

[Lat13] Lategahn, Henning: *Mapping and Localization in Urban Environments Using Cameras*. PhD thesis, Institut für Mess- und Regelungstechnik, Karlsruher Institut für Technologie (KIT), 2013.

[LCS12] Lee, Chee Sing; Clark, D.E.; and Salvi, J.: *SLAM with single cluster PHD filters*. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2096–2101, 2012.

[Lei14] Leica: *Leica 3D Disto User Manual*. 3.0. Leica Geosystems AG, 2014. URL: `http://www.leica-geosystems.de/downloads123/cp/disto/3DDisto/manuals/Leica_3D_Disto_UserManual_en.pdf`.

[LI14] Leung, Keith and Inostroza, Felipe: *A C++ Library for Simultaneous Localization and Mapping using Random Finite Sets*. Visited July 07th 2014. 2014. URL: `https://github.com/kykleung/RFS-SLAM`.

[LIA13] Leung, Keith Y. K.; Inostroza, Felipe; and Adams, Martin: *An improved weighting strategy for Rao-Blackwellized Probability Hypothesis Density simultaneous localization and mapping*. In: *Proceedings of the International Conference on Control, Automation and Information Sciences*, pages 103–110, 2013.

[LIA14] Leung, Keith Y. K.; Inostroza, Felipe; and Adams, Martin: *Evaluating Set Measurement Likelihoods in Random-Finite-Set SLAM*. In: *Proceedings of the 17th International Conference on Information Fusion*, 2014.

[LMT07] Levinson, J.; Montemerlo, M.; and Thrun, S.: *Map-Based Precision Vehicle Localization in Urban Environments*. In: *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, 2007.

[LNP+14] Lee, C.; Nagappa, S.; Palomeras, N.; Clark, D.; and Salvi, J.: *SLAM with SC-PHD Filters: An Underwater Vehicle Application*. In: *IEEE Robotics Automation Magazine*, volume 21, no. 2, pages 38–45, 2014.

[LOC10] Liu, X.; Oreshkin, B. N.; and Coates, M. J.: *Efficient delay-tolerant particle filtering through selective processing of out-of-sequence measurements*. In: *Proceedings of the 13th Conference on Information Fusion*, pages 1–8, 2010.

[Lof14] Löffler, Thomas: *Photographs of the Vehicle's Sensors*. Institute of Measurement, Control and Microtechnology, Ulm University, Ulm, Germany, 2014.

[LSZS13] Lategahn, Henning; Schreiber, Markus; Ziegler, Julius; and Stiller., Christoph: *Urban Localization with Camera and Inertial Measurement Unit.* In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 719–724, 2013.

[LT10] Levinson, Jesse and Thrun, Sebastian: *Robust vehicle localization in urban environments using probabilistic maps.* In: *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4372–4378, 2010.

[LY12] Li, Xiaoming and Yang, Yunying: *An experimental comparison of localization accuracy of affine region detectors.* In: *Proceedings of the 5th International Congress on Image and Signal Processing*, pages 411–414, 2012.

[Mah03] Mahler, Ronald: *Multitarget Bayes filtering via first-order multitarget moments.* In: *IEEE Transactions on Aerospace and Electronic Systems*, volume 39, no. 4, pages 1152–1178, 2003.

[Mah04] Mahler, Ronald: *Random sets: Unification and Computation for Information Fusion - a Retrospective Assessment.* In: *Proceedings of the 7th International Conference on Information Fusion, Stockholm, Sweden*, 2004.

[Mah07] Mahler, Ronald: *Statistical Multisource-Multitarget Information Fusion.* Artech House Inc., Norwood, 2007.

[MC08] Miller, Isaac and Campbell, Mark E.: *Particle filtering for map-aided localization in sparse GPS environments.* In: *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1834–1841, 2008.

[MCUP02] Matas, J.; Chum, O.; Urban, M.; and Pajdla, T.: *Robust Wide Baseline Stereo from Maximally Stable Extremal Regions.* In: *Proceedings of the British Machine Vision Conference*, pages 384–393, 2002.

[MMD09] Munz, Michael; Mählisch, Mirko; and Dietmayer, Klaus: *Probabilistische Sensorfusion und integrierte Existenzschätzung für zukünftige Fahrerassistenzsysteme.* In: Stiller, Christoph and Maurer, Markus (editors): *6. Workshop Fahrerassistenzsysteme (FAS2009)*, pages 166–176, 2009.

[MNT04] Madsen, K.; Nielsen, H. B.; and Tingleff, O.: *Methods for Non-Linear Least Squares Problems (2nd ed.)* 2004.

[Mon03] Montemerlo, Michael: *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association.* PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 2003.

[MSW10] Mattern, Norman; Schubert, Robin; and Wanielik, Gerd: *High-accurate vehicle localization using digital maps and coherency images*. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 462–469, 2010.

[MT03] Montemerlo, M. and Thrun, S.: *Simultaneous localization and mapping with unknown data association using FastSLAM*. In: *IEEE International Conference on Robotics and Automation*. Volume 2, pages 1985–1991, 2003.

[MT07] Montemerlo, Michael and Thrun, Sebastian: *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics (Springer Tracts in Advanced Robotics)*. Springer-Verlag Berlin and Heidelberg GmbH & Co. K, 2007.

[MTKW02] Montemerlo, Michael; Thrun, Sebastian; Koller, Daphne; and Wegbreit, Ben: *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem*. In: *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 593–598, AAAI, 2002.

[MTS+05] Mikolajczyk, K.; Tuytelaars, T.; Schmid, C.; et al.: *A Comparison of Affine Region Detectors*. In: *International Journal of Computer Vision*, volume 65, no. 1-2, pages 43–72, 2005.

[Mun57] Munkres, James: *Algorithms for the Assignment and Transportation Problems*. In: *Journal of the Society for Industrial and Applied Mathematics*, volume 5, no. 1, pages 32–38, 1957.

[Mur68] Murty, K.G.: *An Algorithm for Ranking all the Assignments in Order of Increasing Cost*. In: *Operations Research*, volume 16, pages 682–687, 1968.

[MVAV11a] Mullane, J.; Vo, Ba-Ngu; Adams, M.D.; and Vo, Ba-Tuong: *A Random-Finite-Set Approach to Bayesian SLAM*. In: *IEEE Transactions on Robotics*, volume 27, no. 2, pages 268–282, 2011.

[MVAV11b] Mullane, John; Vo, Ba-Ngu; Adams, Martin; and Vo, Ba-Tuong: *Random Finite Sets for Robot Mapping and SLAM - New Concepts in Autonomous Robotic Map Representations*. From the series *Springer Tracts in Advanced Robotics*. Springer, 2011.

[MVAW08] Mullane, J.; Vo, Ba-Ngu; Adams, M.D.; and Wijesoma, W.S.: *A Random Set Formulation for Bayesian SLAM*. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1043–1049, 2008.

[MWR+14] Moratuwage, D.; Wang, D.; Rao, A.; Senarathne, N.; and Wang, H.: *RFS Collaborative Multivehicle SLAM: SLAM in Dynamic High-Clutter Environments*. In: *IEEE Robotics & Automation Magazine*, volume 21, no. 2, pages 53–59, 2014.

[NS08] Nistér, David and Stewénius, Henrik: *Linear Time Maximally Stable Extremal Regions*. In: Forsyth, David; Torr, Philip; and Zisserman, Andrew (editors): *Computer Vision – ECCV 2008*, volume 5303. From the series *Lecture Notes in Computer Science*, pages 183–196, Springer Berlin Heidelberg, 2008.

[NSD14] Nuss, Dominik; Stuebler, Manuel; and Dietmayer, Klaus: *Consistent Environmental Modeling by Use of Occupancy Grid Maps, Digital Road Maps, and Multi-Object Tracking*. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 1371–1377, 2014.

[NWW+13] Nuss, Dominik; Wilking, Benjamin; Wiest, Jürgen; et al.: *Decision-Free True Positive Estimation with Grid Maps for Multi-Object Tracking*. In: *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems*, pages 28–34, The Hague, Netherlands, 2013.

[PMB09] Pink, O.; Moosmann, F.; and Bachmann, A.: *Visual features for vehicle localization and ego-motion estimation*. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 254–260, 2009.

[PPO+12] Petrovskaya, Anna; Perrollaz, Mathias; Oliveira, Luciano; et al.: *Awareness of Road Scene Participants for Autonomous Driving*. In: Eskandarian, Azim (editor): *Handbook of Intelligent Vehicles*, pages 1383–1432, Springer London, 2012.

[PS10] Pink, Oliver and Stiller, Christoph: *Automated map generation from aerial images for precise vehicle localization*. In: *Proceedings of the 13th International IEEE Conference on Intelligent Transportation Systems*, pages 1517–1522, 2010.

[Reu14] Reuter, Stephan: *Multi-Object Tracking Using Random Finite Sets*. PhD thesis, Ulm University, 2014.

[RIW13] Ranganathan, Ananth; Ilstrup, David; and Wu, Tao: *Light-weight localization for vehicles using road markings*. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 921–927, 2013.

[RVVD14] Reuter, Stephan; Vo, Ba-Tuong; Vo, Ba-Ngu; and Dietmayer, Klaus: *The Labeled Multi-Bernoulli Filter*. In: *IEEE Transactions on Signal Processing*, volume 62, no. 12, pages 3246–3260, 2014.

[SC86] Smith, Randall C. and Cheeseman, Peter: *On the Representation and Estimation of Spatial Uncertainly*. In: *International Journal of Robotics Research*, volume 5, no. 4, pages 56–68, 1986.

[Sch13] Schindler, Andreas: *Vehicle self-localization with high-precision digital maps*. In: *IEEE Intelligent Vehicles Symposium Workshops*, pages 134–139, 2013.

[Sed83] Sedgewick, Robert: *Algorithms*. Addison-Wesley Publishing Company, 1983.

[SH09] Skog, I. and Handel, P.: *In-Car Positioning and Navigation Technologies: A Survey*. In: *IEEE Transactions on Intelligent Transportation Systems*, volume 10, no. 1, pages 4–21, 2009.

[Sid03] Sidenbladh, H.: *Multi-target particle filtering for the probability hypothesis density*. In: *Proceedings of the 6th International Conference on Information Fusion*. Volume 2, pages 800–806, 2003.

[SKF13] Schreiber, Markus; Knöppel, Carsten; and Franke, Uwe: *LaneLoc: Lane marking based localization using highly accurate maps*. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 449–454, 2013.

[Sny87] Snyder, John: *Map Projections - A Working Manual*. In: *U.S. Geoligical Survey Professional Paper 1395*, 1987.

[SSM62] Smith, Gerald L.; Schmidt, Stanley F.; and Mc Gee, Leonard A.: *Application of Statistical Filter Theory to the Optimal Estimation of Position and Velocity on Board a Circumlunar Vehicle*. Tech. rep. NASA Ames Research Center, Tech. Rep. TR R-135, 1962.

[SVV08] Schuhmacher, D.; Vo, B.-T.; and Vo, B.-N.: *A Consistent Metric for Performance Evaluation of Multi-Object Filters*. In: *IEEE Transactions on Signal Processing*, volume 56, no. 8, pages 3447–3457, 2008.

[SWE11] Sun, Hao; Wang, Cheng; and El-Sheimy, Naser: *Automatic Traffic Lane Detection for Mobile Mapping Systems*. In: *International Workshop on Multi-Platform/Multi-Sensor Remote Sensing and Mapping*, pages 1–5, 2011.

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

[TMD+06] Thrun, S.; Montemerlo, M.; Dahlkamp, H.; et al.: *Stanley: The Robot that Won the DARPA Grand Challenge*. In: *Journal of Field Robotics*, 2006.

[UEW07] Ulmke, M.; Erdinc, O.; and Willett, P.: *Gaussian mixture cardinalized PHD filter for ground moving target tracking*. In: *Proceedings of the 10th International Conference on Information Fusion*, pages 1–8, 2007.

[Ulm15] Ulm, Stadt: *Official website of the city of Ulm*. 2015. URL: `http://www.ulm.de/ulm/`.

[Urm+08] Urmson, Christopher et al.: *Autonomous driving in urban environments: Boss and the Urban Challenge*. In: *Journal of Field Robotics Special Issue on the 2007 DARPA Urban Challenge, Part I*, volume 25, no. 8, Martin Buehler Karl Lagnemma, Sanjiv Singh (editor), pages 425–466, 2008.

[Ver15] Vermeulen, Jeroen T.: *libpqxx*. 2015. URL: `http://pqxx.org`.

[VM06] Vo, Ba-Ngu and Ma, Wing-Kin: *The Gaussian Mixture Probability Hypothesis Density Filter*. In: *IEEE Transactions on Signal Processing*, volume 54, no. 11, pages 4091–4104, 2006.

[VV13] Vo, Ba-Tuong and Vo, Ba-Ngu: *Labeled Random Finite Sets and Multi-Object Conjugate Priors*. In: *IEEE Transactions on Signal Processing*, volume 61, no. 13, pages 3460–3475, 2013.

[Wal13] Wallner, Paul: *Multi-Sensor Fahrstreifentracking unter Verwendung von Random Finite Sets*. Diplomarbeit, Universität Ulm, 2013.

[WD08] Weiss, Thorsten and Dietmayer, Klaus: *Applications for Driver Assistant Systems Using Online Maps*. In: *Proceedings of the International Workshop on Intelligent Transportation*, 2008.

[WDN+14] Wiest, Jürgen; Deusch, Hendrik; Nuss, Dominik; et al.: *Localization Based on Region Descriptors in Grid Maps*. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 793–799, 2014.

[Wei11] Weiss, Thorsten: *Hochgenaue Positionierung und Kartographie mit Laserscannern für Fahrerassistenzsysteme*. PhD thesis, University of Ulm, 2011.

[WHW12] Winner, Hermann; Hakuli, Stephan; and Wolf, Gabriele (editors): *Handbuch Fahrerassistenzsysteme*. 2nd Edition, Vieweg+Teubner Verlag, 2012.

[WKD05] Weiss, Thorsten; Kämpchen, Nico; and Dietmayer, Klaus: *Precise ego-localization in urban areas using Laserscanner and high accuracy feature maps*. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 284–289, 2005.

[WSD07] Weiss, Thorsten; Schiele, Bruno; and Dietmayer, Klaus: *Robust Driving Path Detection in Urban and Highway Scenarios Using a Laser Scanner and Online Occupancy Grids*. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 184–189, 2007.

[ZBS+14] Ziegler, J.; Bender, P.; Schreiber, M.; et al.: *Making Bertha Drive—An Autonomous Journey on a Historic Route*. In: *IEEE Intelligent Transportation Systems Magazine*, volume 6, no. 2, pages 8–20, 2014.

ISBN 978-3-941543-22-5

e-ISBN 978-3-941543-23-2