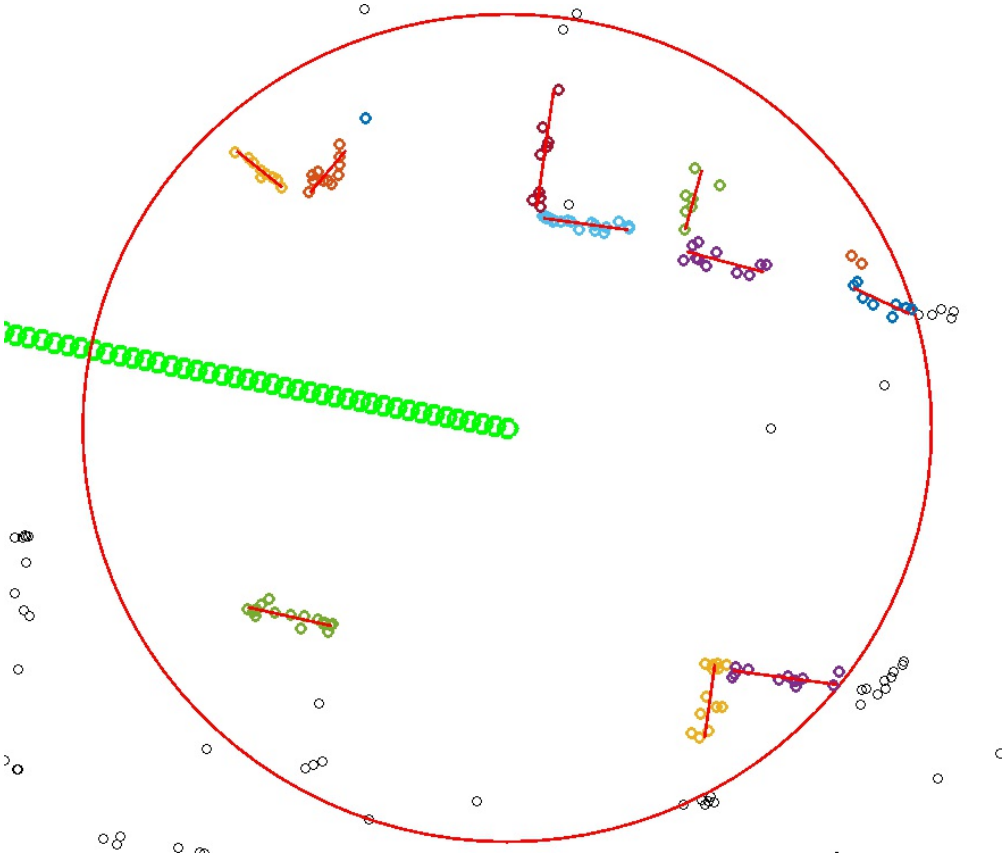


Mapping of Parking Areas using Radar Sensors

with a Cluster-based Landmark Extraction Algorithm and an Extended Kalman Filter

Master's thesis in Algorithms Languages and Logic

Karin Brötjefors, Jacob Gideflod

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

Master's thesis 2018

Mapping of Parking Areas using Radar Sensors

with a Cluster-based Landmark Extraction Algorithm and an Extended Kalman Filter

Karin Brötjefors

Jacob Gideflod

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

Mapping of Parking Areas using Radar Sensors
with a Cluster-based Landmark Extraction Algorithm and an Extended Kalman Filter
Karin Brötjefors
Jacob Gideflod

Supervisor: John Wiedenhoeft, Department of Computer Science and Engineering
Advisor: Andreas Andersson, Aptiv
Examiner: Peter Damaschke, Department of Computer Science and Engineering

Master's Thesis 2018
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Lines extracted from radar data using our landmark extraction algorithm

Typeset in LaTeX
Gothenburg, Sweden 2018

Mapping of Parking Areas using Radar Sensors
with a Cluster-based Landmark Extraction Algorithm and an Extended Kalman Filter

Karin Brötjefors
Jacob Gideflod
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This thesis investigates the possibility of building maps of partially filled parking areas, accurate enough to find empty spaces, using Radio Detection and Ranging (radar) sensors. Existing techniques for solving feature-based Simultaneous Localization and Mapping (SLAM) will be used as basis for map building, but refinements are needed in order to handle noise from radar sensors.

A new landmark extraction algorithm is developed for finding lines and corners within radar data. The algorithm first cluster detections that belong to the same car using a single-linkage clustering, then lines and corners are found within each cluster by a line segmentation algorithm.

The landmarks are used in two different SLAM approaches. The first is a standard SLAM approach using an Extended Kalman Filter (EKF) in combination with single lines as landmarks. The second includes an additional step using an Extended Information Filter (EIF) to maintain correlations between features within more complex landmarks, such as lines and corners of the same car.

Precision and correctness of the algorithms are evaluated in real world scenarios using Light Detection and Ranging (lidar) data in a line by line comparison. Results show that EKF-SLAM maps are noisy, but have most lines located close by cars. It is possible to detect free spaces within the maps, even though noise is present and some lines are too short. Including the EIF correlation step shows promising results for creating less noisy maps, however the landmark extraction limits its performance in dense parking areas. Both approaches can create maps where it is possible to locate available parking spaces.

Keywords: SLAM, Radar, Parking, EKF, Landmark Extraction.

Acknowledgements

First, we would like to thank Aptiv for providing us with the opportunity of doing our master thesis there. It has been a great experience working with their latest technology and meeting talented and helpful coworkers. We especially want to thank our supervisor Andreas Andersson for supporting us during the thesis and providing us with detailed knowledge regarding radar sensors. We would also like to thank Joachim Rukin for helping us form the thesis proposal.

Secondly, we want to thank our supervisor at Chalmers, John Wiedenhoeft, for providing useful ideas and knowledge regarding the development of our algorithms.

Finally, we want to thank our examiner Peter Damaschke for feedback on our thesis.

Karin Brötjefors & Jacob Gideflod, Gothenburg, June 2018

Contents

List of Figures

List of Tables

Acronyms

BFS	Breadth First Search.
EIF	Extended Information Filter.
EKF	Extended Kalman Filter.
KF	Kalman Filter.
lidar	Light Detection and Ranging.
MST	Minimum Spanning Tree.
radar	Radio Detection and Ranging.
SLAM	Simultaneous Localization and Mapping.
SPmodel	Symmetries and Perturbations Model.

1
Introduction

In modern society travelling by car is one of the most commonly used transport methods. In Gothenburg there are on average 445600 vehicle movements every weekday and the car is an important part of many people's everyday life [16]. Modern vehicles are equipped with advanced systems such as adaptive cruise control and automatic parallel parking. These systems aim to minimize accidents and make driving easier. The next step is to make cars more autonomous.

One useful autonomous feature is automated valet parking. This would allow drivers to get out of their cars at a destination and their cars would find parking spaces by themselves. Drivers can save time, and parking spaces can be utilized more efficiently since autonomous cars can park closer to each other. Additionally the number of parking lot accidents caused by human driving errors would be minimized. To achieve automated valet parking cars have to be able to navigate in unknown parking areas and find available places.

1.1 Background

Self-driving cars are under development in many companies and there are already test vehicles on public roads. One of the leading companies is Waymo (previously Google self-driving car project) with four generations of self-driving cars that have driven more than three million miles on public roads [1]. To achieve this their cars are equipped with an advanced sensor system consisting of cameras, Light Detection and Ranging (lidar) and Radio Detection and Ranging (radar) sensors. Radar is a technique that uses radio waves to measure range and range rate to objects. Lidar is a similar technique that instead of radio waves uses laser light, which gives higher accuracy in distance and angular measures. With all this technology there would probably be no match for their cars to navigate in unknown parking areas and find available places. On the other hand all these sensors are expensive and it will take time before they are standard equipment in ordinary cars. However most modern cars are equipped with radar sensors to enable functions such as Blind Spot Warning. Therefore a solution to automated valet parking using radar sensors is desired.

In order to navigate in an unknown parking area a car has to incrementally build a map of its surroundings from sensor observations. This is part of a well known problem within robotics called Simultaneous Localization and Mapping (SLAM). The general SLAM problem is to simultaneously map an environment and compute

an estimate of the location of a vehicle at discrete time steps. One of the properties that makes the problem hard is the uncertainties in both vehicle movements and sensor observations. These uncertainties make it hard to build an accurate map, since the true positions of both the vehicle and the sensor observations are never known. SLAM is considered solved at a theoretical level [2] using a probabilistic approach that can be further read about in section 2.2. Practical realizations of SLAM is however an ongoing research area.

1.2 Related Work

A key question in current research of SLAM is how to represent the map. Dube et al. [3] have successfully made use of a grid map representation of parking areas in combination with radar sensors. A map is represented by a grid consisting of fixed size cells with a probability of being occupied. Each probability is updated based on received radar observations. Their work also presents a classifier that can be used to detect both cross-parked and parallel-parked vehicles within the map.

Schuster et al. [4] describe a different map representation based on several small clusters stored in a tree structure. Each new radar observation is either absorbed into the closest cluster or into a new cluster, if no existing cluster is close enough. To handle false clusters, that occur due to sensor noise, each cluster decays over time. The decay ensures that clusters that seldom absorb radar detections will eventually disappear. This property makes it possible to handle dynamic environments. Schuster et al. [4] demonstrate this property by visiting a parking area several times during a day and showing how a map is maintained even if the configuration of cars has changed.

Both these map representations require a lot of memory, which does not scale well with the size of a map. A more memory efficient approach is a feature-based map, where a map consists of environmental features like lines and corners. This approach has successfully been used by Garulli et al. [5] and Lv et al. [6] in indoor environments with a lidar sensor. Both Garulli and Lv choose to use lines as features, much due to the existence of long straight walls that can easily be observed by lidar sensors. Their SLAM implementations can be described in three steps:

1. Line extraction, which is the process of finding lines from a set of sensor observations. The most common line extraction methods are presented and compared by Nguyen et al. [7].
2. Data association, which is a method for pairing a newly extracted feature to an already existing feature within the map [8].
3. Extended Kalman filtering, which is a method that can be used to update a map and a vehicle position, given an estimated vehicle movement and data association pairs. More details about Extended Kalman Filtering can be found in section 2.2.4.1.

A block of parked cars similarly to an indoor environment contains line-like features, which indicates that it would be possible to use a feature-based SLAM approach to

build a map of a parking block.

1.3 Problem Definition

The aim of this project is to develop an algorithm for incrementally building maps of parking areas using radar sensors. The map should be accurate enough to be able to use for detecting available parking spaces. Existing techniques for solving feature-based SLAM will be used as basis for the map building, but refinements are needed in order to handle noise from radar sensors. The problem can be divided into three sub problems

- Landmark Extraction, which is the problem of finding lines and corners within noisy radar data.
- Data Association, which is the task of pairing a newly extracted landmark with one in the map.
- State Model Update, which is the problem of updating the vehicle state and the map, given new landmarks and their data association.

The algorithm will be designed, implemented and tested on data gathered from real world parking area scenarios. The accuracy of the landmark extraction on radar data will be assessed by comparison to the landmark extraction on more accurate lidar data. Likewise the resulting maps from radar data will be compared to maps from lidar data.

1.4 Delimitations

The project makes the following assumptions and delimitations

- Only scenarios containing parked cars will be considered. Therefore, objects such as trees, poles, rails and other common objects found in parking areas will not be handled.
- Objects in the environment are assumed stationary.
- The ground is assumed to be flat and non-slippery.

1.5 Outline

The thesis is structured in the following way:

Chapter 1 introduces the problem considered in this thesis along with background information and a problem definition.

Chapter 2 provides theory required to understand the rest of the material in this thesis. It is divided into two main topics, landmark extraction and SLAM. The Land-

mark Extraction section contains clustering and line fitting methods that are used to find landmarks within radar data. The Simultaneous Localization and Mapping section explains the probabilistic formulation of SLAM and how different solution methods like the Extended Kalman Filter (EKF) and Extended Information Filter (EIF) works.

Chapter 3 presents the algorithm used to extract landmarks from radar data. That includes descriptions of how different clustering and line fitting methods are used.

Chapter 4 describes two different map building algorithms. Each algorithm's state model and state model update is explained, along with the process of associating a new landmark to a landmark within the current map.

Chapter 5 describes how the developed algorithms are tested and evaluated using lidar data.

Chapter 6 presents test results. First landmark extraction results are presented, where precision of individual landmarks are evaluated, followed by map results where radar maps are compared to lidar maps.

Chapter 7 provides reflections regarding results and suggestions for future work.

Chapter 8 highlights the most important findings of this thesis project and concludes this report.

2

Theory

This chapter presents theory required for understanding methods used for solving SLAM in parking area scenarios. First, graph theory clustering and line fitting techniques used for landmark extraction are presented. Secondly, the theoretical SLAM problem is formulated, followed by an introduction of the distance measure used for data association. Finally, EKF and EIF are introduced as estimation techniques.

2.1 Landmark Extraction

Landmarks are distinguishable features in data that can be recognized at different time steps, such as lines and corners. Clustering algorithms and line fitting methods can be used for extracting landmarks from radar and lidar observations.

2.1.1 Single-linkage Clustering

A single-linkage clustering creates clusters such that all distances between clusters are larger than a threshold d and each point within a cluster has a neighbour closer than d. The distance between clusters is defined by the distance between the two closest points. This clustering can be achieved by creating an Minimum Spanning Tree (MST) and cutting edges longer than threshold d.

A MST is a set of edges that connects all nodes, while minimizing the total edge weight without forming any cycles. A MST can be found by for example Prim's or Kruskal's algorithm in $\mathcal{O}(m \log n)$ time [9], where m is the number of edges and n the number of nodes. Figure 2.1 shows a MST obtained from a complete graph, where points are nodes and weights are the euclidean distances between the points. The time complexity is therefore $\mathcal{O}(n^2 \log n)$, since the number of edges is $m = \frac{n(n-1)}{2}$ in a complete graph. The number of edges, hence the time complexity, can be reduced by performing a Delaunay triangulation on the set of points. A Delaunay triangulation, visualized in figure 2.1, is a triangulation so that no point is inside the circumference of any triangle. The triangulation can be computed in $\mathcal{O}(n \log n)$ and results in a planar graph, which contains $\mathcal{O}(n)$ edges [10]. By running the MST algorithm on the new graph the complexity is reduced to $\mathcal{O}(n \log n)$.

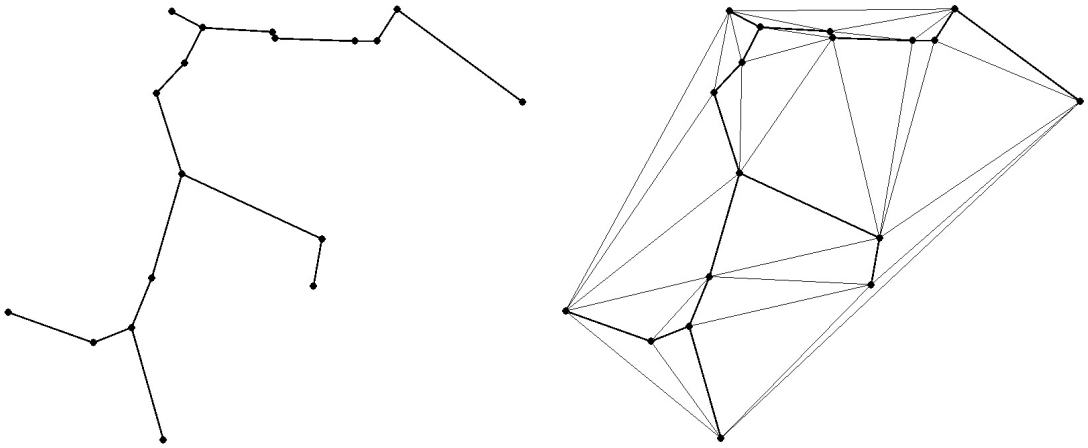


Figure 2.1: *The graph on the left is a MST calculated from the complete graph of the point vertices. The graph on the right is the Delaunay triangulation of the points. The bold subgraph is the MST calculated from the triangulation, hence with a smaller time complexity.*

2.1.2 Graph Diameter

The graph diameter of a MST is in section 3.4.1 used to decrease the number of outliers in clusters. It is defined as the number of nodes on the shortest path between the two most distant nodes [12]. It therefore describes the longest of all shortest paths in a graph. The diameter can be found by first computing heights of all nodes and then finding the node with largest sum of two subtree heights.

2.1.3 Orthogonal Least Square Line Fitting

Lines are fitted to data sets obtained from clustering. A line can be fitted to data points by minimizing the sum of squares of the orthogonal distances from the points to the line [13]. A line in the plane can be represented by

$$\begin{aligned} c + n_1 x + n_2 y &= 0 \\ n_1^2 + n_2^2 &= 1 \end{aligned} \tag{2.1}$$

where the unit vector (n_1, n_2) is normal to the line. The orthogonal distance from a point to a line is

$$r = |c + n_1 x_p + n_2 y_p| \tag{2.2}$$

therefore, the line that minimizes

$$\sum_{i=1}^{n} r_i^2 \tag{2.3}$$

for all points is the desired line. Hence, the problem is to minimize equation (2.3) subject to

$$\begin{pmatrix} 1 & x_{p_1} & y_{p_1} \\ \vdots & \vdots & \vdots \\ 1 & x_{p_n} & y_{p_n} \end{pmatrix} \begin{pmatrix} c \\ n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix}, \tag{2.4}$$

$$n_1^2 + n_2^2 = 1$$

When fitting two orthogonal lines the number of unknowns is only increased by one. This is possible by realizing that the normal vectors of two orthogonal lines are orthogonal. If one line has normal vector (n_1, n_2) then the second line has normal vector $(-n_2, n_1)$. Therefore the unknown parameters to estimate are (c_1, c_2, n_1, n_2). If the set of points $p_1, ..., p_n$ are associated to the first line and another set $q_1, ..., q_m$ to the second line, then the problem in (2.4) is extended for two orthogonal lines as

$$\begin{pmatrix} 1 & 0 & x_{p_1} & y_{p_1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & x_{p_n} & y_{p_n} \\ 0 & 1 & x_{q_1} & y_{q_1} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & x_{q_m} & y_{q_m} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} r_1 \\ \vdots \\ r_{n+m} \end{pmatrix}, \tag{2.5}$$

$$n_1^2 + n_2^2 = 1$$

This scenario is common in parking areas where the type of landmarks are dominated by 90 degree corners of parked cars.

2.2 Simultaneous Localization and Mapping

Estimating a vehicle position while mapping an environment during movement is known as SLAM. The probabilistic problem can be formulated using Bayes' theorem and a solution method is an EKF.

2.2.1 Probabilistic Problem Definition

The probabilistic SLAM problem is to compute the probability distribution of vehicle position and landmark positions at all discrete time steps. Below follows mathematical formulations of the problem, according to Durrant-Whyte et al. [2]. First, the following quantities are defined at time t

- $\mathcal{V}_t$ vehicle location.
- $\mathcal{L}_i$ location of the ith landmark.
- $\mathcal{M}_t = \{\mathcal{L}_1 \dots \mathcal{L}_n\}$ the set of all n landmarks in the state model.
- $\boldsymbol{u}_t$ control input applied at time $t-1$ to make the vehicle drive to $\mathcal{V}_t$.
- $\boldsymbol{z}_m = \boldsymbol{z}_{m,t}$ measurement m of a landmark. Note that subscript t is dropped to simplify notation.
- $\mathcal{U}_{0:t}$ the set of all control inputs up to time t.
- $\mathcal{Z}_{0:t}$ the set of all landmark observations up to time t.

The map is a state vector containing positions of vehicle and landmarks at time t

$$\boldsymbol{x}_t = \begin{bmatrix} \mathcal{V}_t \\ \mathcal{M}_t \end{bmatrix} = \begin{bmatrix} \mathcal{V}_t \\ \mathcal{L}_1 \\ \vdots \\ \mathcal{L}_n \end{bmatrix} \tag{2.6}$$

The size of n increases with time since landmarks are added to the state vector as new landmarks are detected.

The probability distribution to be computed in each time step is

$$P(\mathcal{V}_t, \mathcal{M}_t \mid \mathcal{Z}_{0:t}, \mathcal{U}_{0:t}, \mathcal{V}_0) \tag{2.7}$$

which is the joint posterior density of the vehicle position and landmark locations given radar data, odometry data and the initial vehicle state. The general SLAM solution is to recursively calculate this probability distribution using Bayes theorem. This requires a *vehicle motion model*

$$P(\mathcal{V}_{t+1} \mid \mathcal{V}_t, \boldsymbol{u}_{t+1}) \tag{2.8}$$

that computes expected position of the vehicle, given the old position and control input, to be used as an initial guess. It also requires a *measurement model*

$$P(\boldsymbol{z}_m \mid \mathcal{V}_{t+1}, \mathcal{M}_t) \tag{2.9}$$

that computes expected range and bearing of a landmark provided estimates of vehicle and landmark positions after the vehicle motion model has been applied at time t.

Bayes theorem states that

$$Posterior \propto Likelihood \times Prior \tag{2.10}$$

$$\tag{2.11}$$

The likelihood is in this case the observation model (the likelihood of making a measurement $\boldsymbol{z}_{t+1}$) and the prior is the joint probability density before the new measurement.

The prior is called a time update since it is the probability distribution after the vehicle motion model has been applied to the previous state

$$\begin{aligned} P(\mathcal{V}_{t+1} \mid \mathcal{Z}_{0:t}, \mathcal{U}_{0:t+1}, \mathcal{V}_0) = \\ = \int P(\mathcal{V}_{t+1} \mid \mathcal{V}_t, \boldsymbol{u}_{t+1}) \times P(\mathcal{V}_t, \mathcal{M}_t \mid \mathcal{Z}_{0:t}, \mathcal{U}_{0:t}, \mathcal{V}_0) \mathrm{d}\mathcal{V}_t \end{aligned} \tag{2.12}$$

Equation (2.7) is called a measurement update and can be formulated

$$\begin{aligned} P(\mathcal{V}_{t+1}, \mathcal{M}_{t+1} \mid \mathcal{Z}_{0:t+1}, \mathcal{U}_{0:t+1}, \mathcal{V}_0) = \\ = \frac{P(\mathbf{z}_m \mid \mathcal{V}_{t+1}, \mathcal{M}_t) P(\mathcal{V}_{t+1} \mid \mathcal{Z}_{0:k}, \mathcal{U}_{0:t+1}, \mathcal{V}_0)}{P(\boldsymbol{z}_m \mid \mathcal{Z}_{0:t}, \mathcal{U}_{0:t+1})} \end{aligned} \tag{2.13}$$

This implies that the recursion is carried out in two steps: first the time-update in (2.12) and then the measurement update in (2.13). In order to make a practically usable SLAM solution computation of these equations have to be efficient. Achieving this usually involves selecting a motion model (2.8) and an observation model (2.9) that simplify computation of equation (2.12) and (2.13).

2.2.2 Uncertain Geometric Information

The first step when using a feature-based approach for representing geometric objects is to choose an appropriate model for specifying feature locations and uncertainties. Different geometric features, e.g. lines, endpoints and corners, are represented using different types of parameters. For example, lines can be represented using a point and an angle while endpoints and corners can be represented using a single point.

2.2.2.1 Gaussian Distributions of Feature Parameters

When choosing a model for representing geometric information it is important to consider how characteristics of parameters affect the representation of uncertainty [14]. Gaussian distributions, consisting of means and covariance matrices, are common when representing uncertain information. However, one drawback is that the covariance tends to infinity near singularities and is therefore not suitable for certain location representations. For example, using Gaussian distributions for uncertainties in slope-intersection parameters, $\{(k, m) : y = kx + m\}$, for line representation is not appropriate since k tends to infinity for vertical lines. This does not occur when using Hesse normal form $\{(\rho, \theta) : \rho = x \cos\theta + y \sin\theta\}$, since ρ and θ have finite ranges in a finite world. However, using this form has a different type of drawback. A small uncertainty in orientation θ will result in a large uncertainty in distance ρ.

When choosing model, a problem arises because different features require different parameters. This becomes a problem during parameter estimations since different parameters require different equations.

2.2.2.2 Symmetries and Perturbations Model

The Symmetries and Perturbations Model (SPmodel) by Castellanos et al. [14] is a model for representing geometric features and their uncertainty, avoiding the drawbacks introduced in the previous section. It allows the same parametric representation for lines, edges and corners, and utilizes symmetries between features during pairing and estimation. The idea of the SPmodel is to attach local reference frames to each feature, and to use (x, y, θ) as parameters for each feature. The (x, y)-parameters specify the origin of the reference frame and θ is the angle from the global x-axis to the features x-axis. Reference frames attached to a landmark consisting of two endpoint, two segment and a corner is visualized in figure 2.2.

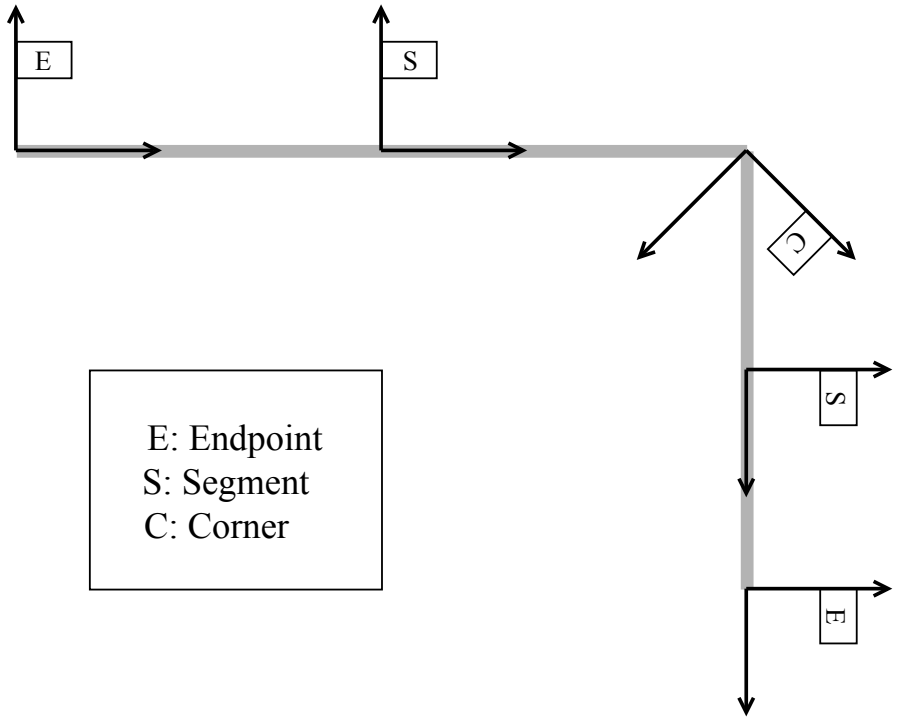


Figure 2.2: *The SPmodel of a landmark consisting of five consecutive features. Each feature has a local reference frame. Segments are aligned with their x-axis, endpoints have their x-axis aligned with the adjacent segments x-axis and corners reference frame has its x-axis along the bisector of the two adjacent segments.*

The reference frame F of a feature can be expressed in the global reference frame G by a location vector $\mathbf{x}_{GF}$ which is both a translation and a rotation

$$\mathbf{x}_{GF} = (x, y, \theta)^\top = \text{Trans}(x, y), \text{Rot}(Z, \theta) \tag{2.14}$$

Using local reference frames for all features make it possible to represent uncertainty by means of a reference frame transformation. If the estimated location has a reference frame specified by location vector $\hat{\mathbf{x}}_{GF}$, then the true feature location $\mathbf{x}_{GF}$ has a reference frame located a small translation and rotation away. This error transformation is represented by a differential location vector $\mathbf{d}_F$, forming relation

$$\mathbf{x}_{GF} = \hat{\mathbf{x}}_{GF} \oplus \mathbf{d}_F \tag{2.15}$$

which is called a *composition*, visualized in figure 2.3 and explained further in Appendix A.3.1.

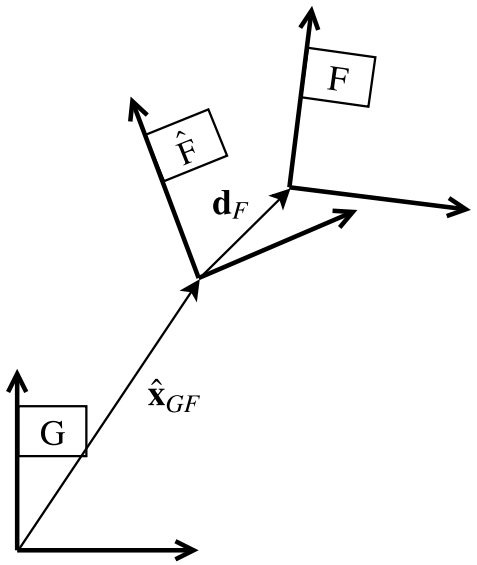


Figure 2.3: *Visualization of the uncertain location of feature F. The true location is a transformation of the reference frames of the estimated location $\hat{\mathbf{x}}_{GF}$ and an error transformation $\mathbf{d}_F$. The error transformation is Gaussian distributed in the parameters (x, y, θ)*

Representing uncertainty in this way makes it possible to remove components of $\mathbf{d}$ that lie along the symmetries of a feature. For example, uncertainty of a line segment can be reduced to (y, θ) since the x-axis is aligned with the segment. This is done using a matrix $\mathbf{B}_F$

$$\mathbf{p}_F = \mathbf{B}_F \mathbf{d}_F \tag{2.16}$$

called the binding matrix of the feature, which forms a new vector called a perturbation vector $\mathbf{p}_F$ for a feature with reference frame F. The perturbation vector is normally distributed with mean $\hat{\mathbf{p}}_F$ and covariance $\mathbf{C}_F$. A perturbation vector is *centered* if the mean is zero, $\hat{\mathbf{p}}_F = 0$. Each type of feature has a specific binding matrix, summarized in table 2.1

Table 2.1: *Binding matrices used in this project*

Feature	Binding Matrix
Vehicle	$\mathbf{B}_V = \mathbf{I}_3$
Endpoint	$\mathbf{B}_E = \mathbf{I}_3$
Segment	$\mathbf{B}_S = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
Corner	$\mathbf{B}_C = \mathbf{I}_3$

The uncertain location of a feature can therefore be represented by

$$\mathbf{L}_{GF} = (\hat{\mathbf{x}}_{GF}, \hat{\mathbf{p}}_F, \mathbf{C}_F, \mathbf{B}_F) \tag{2.17}$$

where

$$\begin{aligned} \mathbf{x}_{GF} &= \hat{\mathbf{x}}_{GF} \oplus \mathbf{B}_F^\top \mathbf{p}_F \\ \hat{\mathbf{p}}_F &= E[\mathbf{p}_F] \\ \mathbf{C}_F &= E[(\mathbf{p}_F - \hat{\mathbf{p}}_F)(\mathbf{p}_F - \hat{\mathbf{p}}_F)^\top] \end{aligned} \tag{2.18}$$

This representation avoids the drawbacks introduced in 2.2.2.1, and utilizes feature symmetries when representing uncertainty. This model also makes it possible to include correlations between features within a landmark, further addressed in section 2.2.4.2.

2.2.3 Data association

In order to update estimated positions of landmarks while moving within a parking area it is necessary to recognize them at different time frames. To achieve this, features at different time frames are paired by measuring the difference in location using a Mahalanobis distance.

2.2.3.1 Pairing Features

Before applying estimation techniques for updating the state vector in equation (2.6), observed features need to be paired with features in the state model. According to Castellanos et al. [14], two features in the SPmodel can be related through a measurement equation

$$\mathbf{f}_m(\mathbf{x}, \mathbf{y}_m) = \mathbf{0} \tag{2.19}$$

where $\mathbf{x}$ represents the location of a feature in the state model and

$$\hat{\mathbf{y}}_m = \mathbf{y}_m + \mathbf{u}_m \;\; ; \;\; \mathbf{u}_m \sim \mathcal{N}(\mathbf{0}, \mathbf{U}_m) \tag{2.20}$$

is an observation. These measurement equations are non-linear due to orientation terms and need to be linearized. This is done using a first order Taylor expansion

$$\begin{aligned}\mathbf{f}_m(\mathbf{x},\mathbf{y}) &= \mathbf{h}_m + \mathbf{H}_m(\mathbf{x}-\hat{\mathbf{x}}) + \mathbf{G}_m(\mathbf{y}_m-\hat{\mathbf{y}}_m)\\ &= \mathbf{0}\end{aligned} \tag{2.21}$$

where

$$\begin{aligned}\mathbf{h}_m &= \mathbf{f}_m(\hat{\mathbf{x}},\hat{\mathbf{y}})\\ \mathbf{H}_m &= \left.\frac{\partial \mathbf{f}_m}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}},\hat{\mathbf{y}}_m}\\ \mathbf{G}_m &= \left.\frac{\partial \mathbf{f}_m}{\partial \mathbf{y}_m}\right|_{\hat{\mathbf{x}},\hat{\mathbf{y}}_m}\end{aligned} \tag{2.22}$$

Therefore, the linearized measurement equation can be formulated

$$\boldsymbol{z}_m = \mathbf{H}_m\mathbf{x} + \mathbf{v}_m \;\;;\;\; \mathbf{v}_m \sim \mathcal{N}(\mathbf{0},\mathbf{R}_m) \tag{2.23}$$

where

$$\begin{aligned}\boldsymbol{z}_m &= -\mathbf{h}_m + \mathbf{H}_m\hat{\mathbf{x}}\\ \mathbf{v}_m &= -\mathbf{G}_m\mathbf{u}_m\\ \mathbf{R}_m &= -\mathbf{G}_m\mathbf{U}_m\mathbf{G}_m^\top\end{aligned} \tag{2.24}$$

Table 2.2: *Binding matrices for paired features used in this project*

Local Feature	Global Feature	Binding Matrix
Endpoint	Endpoint	$\mathbf{B}_{FE} = \mathbf{I}_3$
Segment	Segment	$\mathbf{B}_{FE} = \begin{pmatrix}0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}$
Corner	Endpoint	$\mathbf{B}_{FE} = \begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{pmatrix}$

The SLAM state vector consist of perturbations vectors when using the SPmodel. Therefore perturbation vectors are estimated during the estimation processes described in sections 2.2.4.1 and 2.2.4.2. Measurement equations are therefore formulated to relate perturbation vectors with observations. Let F be a reference frame specifying the location of feature $\mathcal{F}$ in the state model. Similarly, let E be the reference specifying the location of an observed feature $\mathcal{E}$. A measurement equation relating perturbation vectors $\mathbf{p}_F$ and $\mathbf{p}_E$ can be formulated by viewing $\mathcal{E}$ from F, visualized in figure 2.4. If feature $\mathcal{F}$ and $\mathcal{E}$ represent the same feature, their relative location vector $\mathbf{x}_{FE}$ should equal zero, forming measurement equation

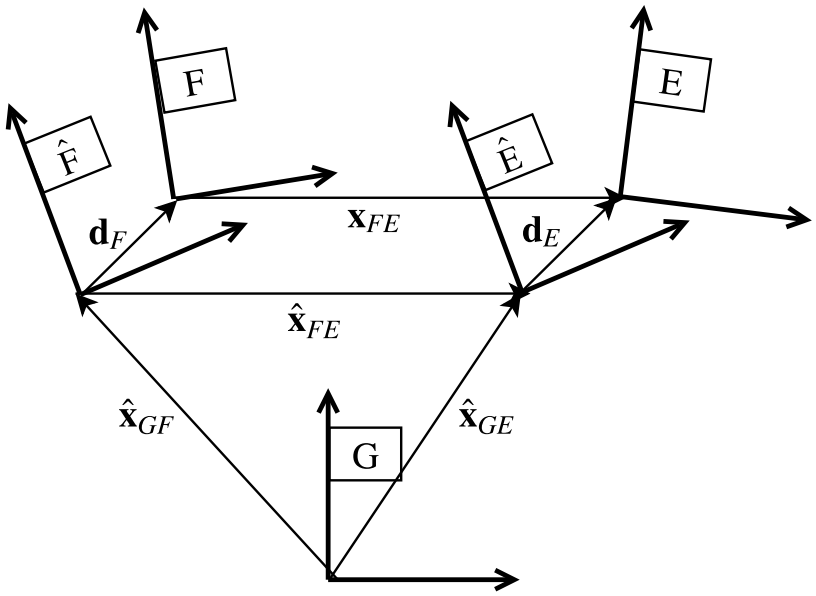


Figure 2.4: *Pairing two features is done using the relative location vector* $\boldsymbol{x}_{FE}$.

$$\mathbf{f}_m(\mathbf{p}_F, \mathbf{p}_E) = \mathbf{B}_{FE}\mathbf{x}_{FE} = \mathbf{0} \tag{2.25}$$

where binding matrix $\mathbf{B}_{FE}$ removes components that lie along symmetries of the pairing. Binding matrices for pairings used in this project are listed in table 2.2. This equation can be expressed using compositions and inversions of transformations according to

$$\begin{aligned}
\mathbf{f}_m(\mathbf{p}_F, \mathbf{p}_E) &= \mathbf{B}_{FE}\mathbf{x}_{FE} \\
&= \mathbf{B}_{FE}(\ominus\mathbf{x}_{GF} \oplus \mathbf{x}_{GE}) \\
&= \mathbf{B}_{FE}(\ominus(\hat{\mathbf{x}}_{GF} \oplus \mathbf{B}_F^\top\mathbf{p}_F) \oplus (\hat{\mathbf{x}}_{GE} \oplus \mathbf{B}_E^\top\mathbf{p}_E)) \\
&= \mathbf{B}_{FE}(\ominus\mathbf{B}_F^\top\mathbf{p}_F \ominus \hat{\mathbf{x}}_{GF} \oplus \hat{\mathbf{x}}_{GE} \oplus \mathbf{B}_E^\top\mathbf{p}_E) \\
&= \mathbf{B}_{FE}(\ominus\mathbf{B}_F^\top\mathbf{p}_F \oplus \hat{\mathbf{x}}_{FE} \oplus \mathbf{B}_E^\top\mathbf{p}_E) \\
&= \mathbf{0}
\end{aligned} \tag{2.26}$$

Linearization of equation (2.30) gives

$$\mathbf{f}_m \approx \mathbf{h}_m + \mathbf{H}_m(\mathbf{p}_F - \hat{\mathbf{p}}_F) + \mathbf{G}_m(\mathbf{p}_S - \hat{\mathbf{p}}_S) = \mathbf{0} \tag{2.27}$$

where the coefficients are derived by Castellanos et al. [14] as

$$
\begin{aligned}
\mathbf{h}_m &= \mathbf{f}(\hat{\mathbf{p}}_F, \hat{\mathbf{p}}_E) \\
&= \mathbf{B}_{FE}(\ominus \mathbf{B}_F^\top \hat{\mathbf{p}}_F \oplus \hat{\mathbf{x}}_{FE} \oplus \mathbf{B}_E^\top \hat{\mathbf{p}}_E) \\
\mathbf{H}_m &= \left.\frac{\partial \mathbf{f}}{\partial \mathbf{p}_F}\right|_{\hat{\mathbf{p}}_F, \hat{\mathbf{p}}_E} \\
&= \mathbf{B}_{FE} \left[\frac{\ominus \mathbf{B}_F^\top \mathbf{p}_F \oplus \hat{\mathbf{x}}_{FE} \oplus \mathbf{B}_E^\top \mathbf{p}_E}{\ominus \mathbf{B}_F^\top \mathbf{p}_F} \cdot \frac{\ominus \mathbf{B}_F^\top \mathbf{p}_F}{\mathbf{B}_F^\top \mathbf{p}_F} \cdot \frac{\mathbf{B}_F^\top \mathbf{p}_F}{\mathbf{p}_F} \right]_{\hat{\mathbf{p}}_F, \hat{\mathbf{p}}_E} \\
&= \mathbf{B}_{FE} \mathbf{J}_{1\oplus}\{\ominus \mathbf{B}_F^\top \hat{\mathbf{p}}_F, \hat{\mathbf{x}}_{FE} \oplus \mathbf{B}_E^\top \hat{\mathbf{p}}_E\} \mathbf{J}_{\ominus}\{\ominus \mathbf{B}_F^\top \hat{\mathbf{p}}_F\} \mathbf{B}_F^\top \\
\mathbf{G}_m &= \left.\frac{\partial \mathbf{f}}{\partial \mathbf{p}_E}\right|_{\hat{\mathbf{p}}_F, \hat{\mathbf{p}}_E} \\
&= \mathbf{B}_{FE} \left[\frac{\ominus \mathbf{B}_F^\top \mathbf{p}_F \oplus \hat{\mathbf{x}}_{FE} \oplus \mathbf{B}_E^\top \mathbf{p}_E}{\ominus \mathbf{B}_E^\top \mathbf{p}_E} \cdot \frac{\mathbf{B}_E^\top \mathbf{p}_E}{\mathbf{p}_E} \right]_{\hat{\mathbf{p}}_F, \hat{\mathbf{p}}_E} \\
&= \mathbf{B}_{FE} \mathbf{J}_{2\oplus}\{\ominus \mathbf{B}_F^\top \hat{\mathbf{p}}_F \oplus \hat{\mathbf{x}}_{FE}, \mathbf{B}_E^\top \hat{\mathbf{p}}_E\} \mathbf{B}_E^\top
\end{aligned}
\tag{2.28}
$$

Jacobians $\mathbf{J}_{\ominus}$, $\mathbf{J}_{1\oplus}$ and $\mathbf{J}_{2\oplus}$ of compositions and inversions are found in Appendix A.3.2. Assuming centered estimations we have $\hat{\mathbf{p}}_F = \mathbf{0}$ and $\hat{\mathbf{p}}_F = \mathbf{0}$, which simplifies the expressions to

$$
\begin{aligned}
\mathbf{h}_m = \mathbf{f}(\hat{\mathbf{p}}_F, \hat{\mathbf{p}}_E) &= \mathbf{B}_{FE} \hat{\mathbf{x}}_{FE} \\
\mathbf{H}_m = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{p}_F}\right|_{\hat{\mathbf{p}}_F, \hat{\mathbf{p}}_E} &= -\mathbf{B}_{FE} \mathbf{J}_{1\oplus}\{\mathbf{0}, \hat{\mathbf{x}}_{FE}\} \mathbf{B}_F^\top \\
\mathbf{G}_m = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{p}_E}\right|_{\hat{\mathbf{p}}_F, \hat{\mathbf{p}}_E} &= \mathbf{B}_{FE} \mathbf{J}_{2\oplus}\{\hat{\mathbf{x}}_{FE}, \mathbf{0}\} \mathbf{B}_E^\top
\end{aligned}
\tag{2.29}
$$

2.2.3.2 Pairing Features and Observations

During the pairing process in the previous section both features were assumed to be in the global coordinate system. During SLAM, however, new features are observed from the vehicle reference frame V which itself is included in the state vector and is uncertain. This situation is visualized in figure 2.5.

For this case, the measurement equation relating two features $\mathcal{F}$ and $\mathcal{E}$ is

$$
\begin{aligned}
\mathbf{f}_m(\mathbf{p}, \mathbf{p}_E) &= \mathbf{B}_{FE} \mathbf{x}_{FE} \\
&= \mathbf{B}_{FE}(\ominus \mathbf{x}_{GF} \oplus \mathbf{x}_{GV} \oplus \mathbf{x}_{VE}) \\
&= \mathbf{B}_{FE}(\ominus \mathbf{B}_F^\top \mathbf{p}_F \oplus \hat{\mathbf{x}}_{FE} \oplus \mathbf{J}_{EV} \mathbf{d}_V \oplus \mathbf{B}_E^\top \mathbf{p}_E) \\
&= \mathbf{0}
\end{aligned}
\tag{2.30}
$$

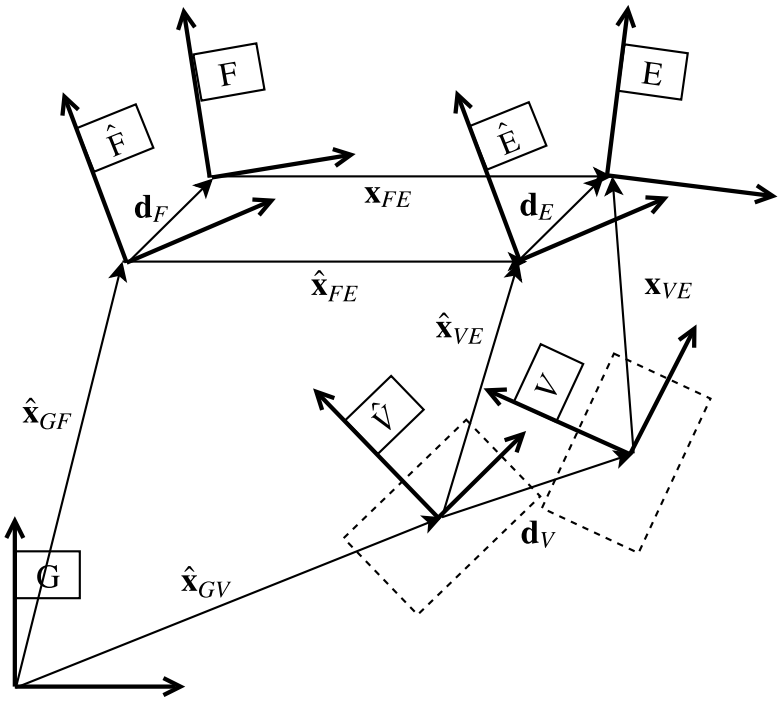


Figure 2.5: *Pairing a state model feature with a feature observed from the vehicle reference frame is done using the relative location vector $\boldsymbol{x}_{FE}$.*

where $\mathbf{p}$ include perturbation vectors of the entire SLAM state vector

$$\mathbf{p} = \begin{bmatrix} \mathbf{p}_V \\ \mathbf{p}_{F_1} \\ \vdots \\ \mathbf{p}_{F_n} \end{bmatrix} \tag{2.31}$$

and $\mathbf{J}_{ER}$ is the Jacobian of the transformation $\mathbf{x}_{EV}$ for propagating vehice uncertainty to feature $\mathcal{E}$, explained further in Appendix A.3.3. Coefficients derived with respect to the state vector result in nonzero values for vehicle position and feature $\mathcal{F}$ which is included in the pairing

$$\begin{aligned} \mathbf{h}_m &= \mathbf{B}_{FE}\hat{\mathbf{x}}_{FE} \\ \mathbf{H}_m &= \begin{pmatrix} \mathbf{H}_\mathcal{V} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{H}_\mathcal{F} & \mathbf{0} & \dots & \mathbf{0} \end{pmatrix} \\ \mathbf{H}_\mathcal{V} &= \mathbf{B}_{FE}\mathbf{J}_{2\oplus}\{\hat{\mathbf{x}}_{FE}, \mathbf{0}\}\mathbf{J}_{EV} \\ \mathbf{H}_\mathcal{F} &= -\mathbf{B}_{FE}\mathbf{J}_{1\oplus}\{\mathbf{0}, \hat{\mathbf{x}}_{FE}\}\mathbf{B}_F^\top \\ \mathbf{G}_\mathcal{E} &= \mathbf{B}_{FE}\mathbf{J}_{2\oplus}\{\hat{\mathbf{x}}_{FE}, \mathbf{0}\}\mathbf{B}_E^\top \end{aligned} \tag{2.32}$$

2.2.3.3 Mahalanobis Distance

The Mahalanobis distance is a distance between an observations $\mathbf{x} = [x_1, x_2, \dots, x_n]^\top$ and a distribution [8]. If the distribution has mean $\boldsymbol{\mu} = [\mu_1, \mu_2, \dots, \mu_n]^\top$ and co-

variance $\mathbf{S}$ then the Mahalanobis distance is defined as

$$D = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^\top \mathbf{S}^{-1} (\mathbf{x} - \boldsymbol{\mu})} \tag{2.33}$$

Two landmarks can be considered equal if the squared Mahalanobis distance is below a threshold obtained from the chi-square distribution

$$D^2 \leq \chi^2_{r,\alpha} \tag{2.34}$$

for a certain significance level α and $r = rank(\mathbf{x})$ degrees of freedom.

2.2.4 Estimating Uncertain Locations

Estimating state model (2.6) through time update (2.12) and measurement update (2.13) requires estimation techniques. This section introduces two types of estimation techniques: EKF and EIF. An EKF integrates one measurement at the time and updates the state model after each integration. The EIF, however, integrates a batch of measurements simultaneously and updates the state model after each batch.

2.2.4.1 Extended Kalman Filter

When using a Kalman Filter (KF) the map defined in (2.6) is modelled by Gaussian variables, presented in section 2.2.2.1, using the mean $\hat{\boldsymbol{x}}_t$ and covariance $\mathbf{C}_t$ of the state vector at time t

$$\hat{\boldsymbol{x}}_t = \begin{bmatrix} \hat{\mathcal{V}} \\ \hat{\mathcal{M}} \end{bmatrix}_t \qquad \mathbf{C}_t = \begin{bmatrix} \mathbf{C}_{\mathcal{V}|\mathcal{V}} & \mathbf{C}_{\mathcal{V}|\mathcal{M}} \\ \mathbf{C}^T_{\mathcal{M}|\mathcal{V}} & \mathbf{C}_{\mathcal{M}|\mathcal{M}} \end{bmatrix}_t \tag{2.35}$$

The goal of a KF is to keep $\hat{\boldsymbol{x}}$ and $\mathbf{C}$ updated at all time steps. According to Durrant-Whyte et al. [2] a KF can be applied by making the following definitions. First, the conditional mean is defined as

$$\hat{\boldsymbol{x}}_{t|l} = E[\boldsymbol{x}_t \mid \mathcal{Z}_{0:l}, \mathcal{U}_{0:t+1}, \mathcal{V}_0](t \geq l) \tag{2.36}$$

For future reference this notation is simplified to $\hat{\boldsymbol{x}}_t$ when $l = t$. The mean at time $t + 1$ is therefore

$$\hat{\boldsymbol{x}}_{t+1} = \begin{bmatrix} \hat{\mathcal{V}}_{t+1} \\ \hat{\mathcal{M}}_{t+1} \end{bmatrix} = E\left[\begin{bmatrix} \mathcal{V}_{t+1} \\ \mathcal{M}_{t+1} \end{bmatrix} \mid \mathcal{Z}_{0:t+1}, \mathcal{U}_{0:t+1}, \mathcal{V}_0 \right] \tag{2.37}$$

and the covariance matrix

$$\begin{aligned}\mathbf{C}_{t+1} &= \begin{bmatrix} \mathbf{C}_{\mathcal{V}|\mathcal{V}} & \mathbf{C}_{\mathcal{V}|\mathcal{M}} \\ \mathbf{C}_{\mathcal{V}|\mathcal{M}}^T & \mathbf{C}_{\mathcal{M}|\mathcal{M}} \end{bmatrix}_{t+1} \\ &= E\left[\begin{pmatrix} \mathcal{V}_{t+1} - \hat{\mathcal{V}}_{t+1} \\ \mathcal{M}_{t+1} - \hat{\mathcal{M}}_{t+1} \end{pmatrix} \begin{pmatrix} \mathcal{V}_{t+1} - \hat{\mathcal{V}}_{t+1} \\ \mathcal{M}_{t+1} - \hat{\mathcal{M}}_{t+1} \end{pmatrix}^T \mid \mathcal{Z}_{0:t+1}, \mathcal{U}_{0:t+1}, \mathcal{V}_0 \right]\end{aligned} \tag{2.38}$$

Vehicle motion model (2.8) can be expressed as

$$P(\mathcal{V}_{t+1} \mid \mathcal{V}_t, \mathbf{u}_{t+1}) \Leftrightarrow \mathcal{V}_{t+1} = \mathbf{f}(\mathcal{V}_t, \boldsymbol{u}_{t+1}) + \mathbf{w} \tag{2.39}$$

where $\mathbf{f}$ is a non-linear function of vehicle kinematics and $\mathbf{w}$ additional Gaussian noise with zero mean and covariance $\mathbf{Q}$.

Measurement model (2.9) for the mth measurement can be expressed as

$$P(\boldsymbol{z}_m \mid \mathcal{V}_{t+1}, \mathcal{M}_t) \Leftrightarrow \boldsymbol{z}_m = \mathbf{h}_m(\mathcal{V}_{t+1}, \mathcal{M}_t) + \mathbf{v}_m \tag{2.40}$$

where $\mathbf{h}_m$ is a non-linear function that returns position of a landmark in the vehicle's reference frame and $\mathbf{v}_m$ additional Gaussian noise with zero mean and covariance $\mathbf{R}_m$.

Since $\mathbf{f}$ and $\mathbf{h}_m$ are non-linear an EKF is used, which means that function (2.39) and (2.40) are linearized around the current estimate in each time step.

The first step of the recursive process introduced in section 2.2.1 is the time update. To apply $\mathbf{f}$ on covariance matrix (2.38) the Jacobian is computed at the estimate $\hat{\mathcal{V}}_t$. The time update is therefore

$$\begin{aligned}\hat{\boldsymbol{x}}_{t+1|t} &= \begin{bmatrix} \mathbf{f}(\hat{\mathcal{V}}_t, \mathbf{u}_{t+1}) \\ \hat{\mathcal{M}}_t \end{bmatrix} = \begin{bmatrix} \hat{\mathcal{V}}_{t+1} \\ \hat{\mathcal{M}}_t \end{bmatrix} \\ \mathbf{C}_{t+1|t} &= \mathbf{F}\, \mathbf{C}_{t|t} \mathbf{F}^T + \mathbf{Q}\end{aligned} \tag{2.41}$$

where $\mathbf{F} = \nabla \mathbf{f}$. Time updates are not applied for landmarks since they are assumed stationary.

Assuming correct landmark extraction and association, the second step is the measurement update. First a prediction of a landmark location at time $t+1$ is made with measurement model (2.40) from the estimated vehicle position $\hat{\mathcal{V}}_{t+1}$

$$\hat{\boldsymbol{z}}_m = \mathbf{h}_m(\hat{\mathcal{V}}_{t+1}, \hat{\mathcal{M}}_t) \tag{2.42}$$

Then a real observation $\boldsymbol{z}_m$ of a landmark at time $t+1$ is made according measurement model (2.40) from the true vehicle position $\mathcal{V}_{t+1}$

$$\boldsymbol{z}_m = \mathbf{h}_m(\mathcal{V}_{t+1}, \mathcal{M}_t) + \mathbf{v}_m \tag{2.43}$$

Now the following matrices can be formed

$$\begin{aligned} \boldsymbol{v}_m &= \boldsymbol{z}_m - \hat{\boldsymbol{z}}_m \\ \mathbf{S}_m &= \mathbf{H}_m \mathbf{C}_{t+1|t} \mathbf{H}_m^T + \mathbf{R}_m \\ \mathbf{K}_m &= \mathbf{C}_{t+1|t} \mathbf{H}_m^T \mathbf{S}_m^{-1} \end{aligned} \tag{2.44}$$

where $\mathbf{H}_m = \nabla \mathbf{h}_m$ is the Jacobian of $\mathbf{h}_m$ at the estimate $\hat{\mathcal{V}}_{t+1}$ and $\hat{\mathcal{M}}_t$, $\boldsymbol{v}_m$ the innovation (difference between the observed value at time $t+1$ and the prediction based on prior information), $\mathbf{S}_m$ the innovation covariance matrix and $\mathbf{K}$ the optimal Kalman gain. Kalman gain is a measure of how much to trust the observed landmarks.

Equations (2.41) and (2.44) are used to form mean and covariance for state vector $\boldsymbol{x}_{t+1}$

$$\hat{\boldsymbol{x}}_{t+1} = \hat{\boldsymbol{x}}_{t+1|t} + \mathbf{K}_m \boldsymbol{v}_m \tag{2.45}$$

$$\mathbf{C}_{t+1} = (\mathbf{I} - \mathbf{K}_m \mathbf{H}_m) \mathbf{C}_{t+1|t} \tag{2.46}$$

and the recursive step is complete.

2.2.4.2 Extended Information Filter

The EKF introduced in section 2.2.4.1 has an equivalent form: the information form [15]. An EIF is also called an inverse covariance filter since the information form of covariance matrix $\mathbf{C}$ and state vector $\hat{\mathbf{x}}$ is replaced with an information matrix and information vector

$$\begin{aligned} \hat{\boldsymbol{y}}_t &= \mathbf{C}_t^{-1} \hat{\boldsymbol{x}}_t \\ \mathbf{Y}_t &= \mathbf{C}_t^{-1} \end{aligned} \tag{2.47}$$

which consist of the inverse covariance matrix. The information form of predicted covariance and state vector is in the same manner expressed as

$$\begin{aligned} \hat{\boldsymbol{y}}_{t+1|t} &= \mathbf{C}^{-1} \hat{\boldsymbol{x}}_{t+1|t} \\ \mathbf{Y}_{t+1|t} &= \mathbf{C}_{t+1|t}^{-1} \end{aligned} \tag{2.48}$$

The measurement vector and covariance in information form are defined as

$$\begin{aligned} \mathbf{I}_m &= \mathbf{H}_m^\top \mathbf{R}_m^{-1} \mathbf{H}_m \\ \mathbf{i}_m &= \mathbf{H}_m^\top \mathbf{R}_m^{-1} \boldsymbol{z}_m \end{aligned} \tag{2.49}$$

and the measurement update of N measurements

$$\begin{aligned}\hat{\boldsymbol{y}}_{t+1} &= \hat{\boldsymbol{y}}_{t+1|t} + \sum_{m=1}^{N} \mathbf{i}_m \\ \mathbf{Y}_{t+1} &= \mathbf{Y}_{t+1|t} + \sum_{m=1}^{N} \mathbf{I}_m\end{aligned} \tag{2.50}$$

Equation (2.50) implies that EIFs support integration of multiple measurements in one time step. Castellanos et al. [14] connect features within a landmark through measurement equations and reestablish correlations using an EIF. An endpoint $\mathcal{E}$ with reference frame E and the segment $\mathcal{S}$ with reference frame S from which the endpoint was being derived from, are related by considering $\mathcal{S}$ as a measurement m. The state vector to be updated includes both the endpoint and segments perturbation vectors

$$\mathbf{x} = \begin{bmatrix}\mathbf{p}_E \\ \mathbf{p}_S\end{bmatrix} \tag{2.51}$$

and the measurement equation relating them is

$$\begin{aligned}\mathbf{f}_m(\mathbf{x}, \mathbf{p}_S) &= \mathbf{B}_{SE}\mathbf{x}_{SE} \\ &= \mathbf{B}_{SE}(\ominus \mathbf{B}_S^\top \mathbf{p}_S \oplus \hat{\mathbf{x}}_{SE} \oplus \mathbf{B}_E^\top \mathbf{p}_E) \\ &= \mathbf{0}\end{aligned} \tag{2.52}$$

where

$$\mathbf{B}_{SE} = \begin{pmatrix}0 & 1 & 0 \\ 0 & 0 & 1\end{pmatrix} \tag{2.53}$$

Linearizing equation (2.52) gives

$$\mathbf{f}_m(\mathbf{x}, \mathbf{p}_S) \approx \mathbf{h}_m + \mathbf{H}_m(\mathbf{x} - \hat{\mathbf{x}}) + \mathbf{G}_m(\mathbf{p}_S - \hat{\mathbf{p}}_S) = \mathbf{0} \tag{2.54}$$

where the coefficients are calculated according to equation (2.29) to

$$\begin{aligned}\mathbf{h}_m &= \mathbf{f}_m(\hat{\mathbf{x}}, \hat{\mathbf{p}}_S) \ = \mathbf{B}_{SE}\hat{\mathbf{x}}_{SE} \\ \mathbf{H}_m &= \begin{pmatrix}\mathbf{H}_m^E & \mathbf{H}_m^S\end{pmatrix} \\ \mathbf{H}_m^E &= \ \ \mathbf{B}_{SE}\mathbf{J}_{2\oplus}\{\hat{\mathbf{x}}_{SE}, \mathbf{0}\}\mathbf{B}_E^\top \\ \mathbf{H}_m^S &= -\mathbf{B}_{SE}\mathbf{J}_{1\oplus}\{\mathbf{0}, \hat{\mathbf{x}}_{SE}\}\mathbf{B}_S^\top \\ \mathbf{G}_m &= -\mathbf{B}_{SE}\mathbf{J}_{1\oplus}\{\mathbf{0}, \hat{\mathbf{x}}_{SE}\}\mathbf{B}_S^\top\end{aligned} \tag{2.55}$$

Note that location vector $\hat{\mathbf{x}}_{SE}$ specify the endpoint location from the segments reference frame while the segment is considered a measurement. In the general derivation

of coefficients in equation (2.28) this vector is inverted, explaining the difference in sign and Jacobian order from the coefficients in (2.55). Also note that $\mathbf{H}_m^S$ and $\mathbf{G}_m$ are identical since the perturbation vector $\mathbf{p}_S$ is considered as a measurement and is also included in the state vector.

Corners are related to segments in the same fashion, using the state vector containing perturbation vectors of the segment and corner to be related

$$\mathbf{x} = \begin{bmatrix} \mathbf{p}_C \\ \mathbf{p}_S \end{bmatrix} \tag{2.56}$$

However, since the reference frame of a corner lies along the bisector of two consecutive segments, the difference in orientation should not equal zero. The measurement equation relating corners and segments is therefore

$$\begin{aligned} \mathbf{g}_m(\mathbf{x}, \mathbf{p}_S) &= \mathbf{B}_{SC}\mathbf{x}_{SC} \\ &= \mathbf{B}_{SC}(\ominus \mathbf{B}_S^\top \mathbf{p}_S \oplus \hat{\mathbf{x}}_{SC} \oplus \mathbf{B}_C^\top \mathbf{p}_C) \\ &= \lambda \end{aligned} \tag{2.57}$$

where λ is the desired angle between segment $\mathcal{S}$ and corner $\mathcal{C}$ and

$$\mathbf{B}_{SC} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.58}$$

This can be written

$$\mathbf{f}_m(\mathbf{x}, \mathbf{p}_S) = \mathbf{g}_m(\mathbf{x}, \mathbf{p}_S) - \lambda = \mathbf{0} \tag{2.59}$$

and linearization gives

$$\mathbf{f}_m(\mathbf{x}, \mathbf{p}_S) \approx \mathbf{h}_m + \mathbf{H}_m(\mathbf{x} - \hat{\mathbf{x}}) + \mathbf{G}_m(\mathbf{p}_S - \hat{\mathbf{p}}_S) = \mathbf{0} \tag{2.60}$$

where the coefficients are calculated using equation (2.29) to

$$\begin{aligned} \mathbf{h}_m &= \mathbf{f}_m(\hat{\mathbf{x}}, \hat{\mathbf{p}}_S) \ = \mathbf{B}_{SC}\hat{\mathbf{x}}_{SC} - \lambda \\ \mathbf{H}_m &= \begin{pmatrix} \mathbf{H}_m^C & \mathbf{H}_m^S \end{pmatrix} \\ \mathbf{H}_m^C &= \ \ \mathbf{B}_{SC}\mathbf{J}_{2\oplus}\{\hat{\mathbf{x}}_{SC}, \mathbf{0}\}\mathbf{B}_C^\top \\ \mathbf{H}_m^S &= -\mathbf{B}_{SE}\mathbf{J}_{1\oplus}\{\mathbf{0}, \hat{\mathbf{x}}_{SC}\}\mathbf{B}_S^\top \\ \mathbf{G}_m &= -\mathbf{B}_{SC}\mathbf{J}_{1\oplus}\{\mathbf{0}, \hat{\mathbf{x}}_{SC}\}\mathbf{B}_S^\top \end{aligned} \tag{2.61}$$

Coefficients relating features within a landmark are used in an EIF to update the entire landmark to reestablish correlations. Consider an L-shaped landmark consisting of five features

$$\mathcal{L} = \begin{pmatrix} \mathcal{E}_1 & \mathcal{S}_1 & \mathcal{C} & \mathcal{S}_2 & \mathcal{E}_2 \end{pmatrix}. \tag{2.62}$$

Features within $\mathcal{L}$ can be related using measurement equations (2.55) and (2.61). Segment $\mathcal{S}_1$ is related to endpoint $\mathcal{E}_1$ and corner $\mathcal{C}$, and segment $\mathcal{S}_2$ is related to corner $\mathcal{C}$ and endpoint $\mathcal{E}_2$. This forms two matrices, $\mathbf{H}_1$ and $\mathbf{H}_2$, containing relation coefficients

$$\begin{aligned}\mathbf{H}_1 &= \begin{pmatrix} \mathbf{H}_1^{E_1} & \mathbf{H}_1^{S_1} & 0 & 0 & 0 \\ 0 & \mathbf{H}_2^{S_1} & \mathbf{H}_2^{C} & 0 & 0 \end{pmatrix} \\ \mathbf{H}_2 &= \begin{pmatrix} 0 & 0 & \mathbf{H}_3^{C} & \mathbf{H}_3^{S_2} & 0 \\ 0 & 0 & 0 & \mathbf{H}_4^{S_2} & \mathbf{H}_4^{E_2} \end{pmatrix}\end{aligned} \tag{2.63}$$

where $\mathbf{H}_m^F$ is the coefficient for feature $\mathcal{F}$ in measurement equation m. The coefficients $\mathbf{H}_1$ and $\mathbf{H}_2$ include measurement equations for relating the state vector

$$\mathbf{x}^{\mathcal{L}} = \begin{bmatrix} \mathbf{p}_{E_1} & \mathbf{p}_{S_1} & \mathbf{p}_C & \mathbf{p}_{S_2} & \mathbf{p}_{E_2} \end{bmatrix} \tag{2.64}$$

The correlation update is performed after each reobservation of a feature to reestablish correlations. The reobserved feature should not be modified during the correlation update, therefore a measurement equation relating the observed feature with itself needs to be added. Relating two identical features using equation

$$\begin{aligned}\mathbf{f}_m(\mathbf{p}_S, \mathbf{p}_S) &= \mathbf{B}_S \mathbf{x}_{SS} \\ &= \mathbf{B}_S(\ominus \mathbf{B}_S^\top \mathbf{p}_S \oplus \hat{\mathbf{x}}_{SS} \oplus \mathbf{B}_S^\top \mathbf{p}_S) \\ &= \mathbf{0}\end{aligned} \tag{2.65}$$

result in coefficients that equal the identity

$$\begin{aligned}\mathbf{h}_m &= \mathbf{0} \\ \mathbf{H}_m^{SS} &= -\mathbf{I}_2 \\ \mathbf{G}_m &= \mathbf{I}_2\end{aligned} \tag{2.66}$$

Adding $\mathbf{H}_m^{SS}$ concludes the measurement equations

$$\begin{aligned}\mathbf{H}_1 &= \begin{pmatrix} \mathbf{H}_1^{E_1} & \mathbf{H}_1^{S_1} & 0 & 0 & 0 \\ 0 & \mathbf{H}_2^{S_1} & \mathbf{H}_2^{C} & 0 & 0 \\ 0 & \mathbf{H}_3^{S_1 S_1} & 0 & 0 & 0 \end{pmatrix} \\ \mathbf{H}_2 &= \begin{pmatrix} 0 & 0 & \mathbf{H}_4^{C} & \mathbf{H}_4^{S_2} & 0 \\ 0 & 0 & 0 & \mathbf{H}_5^{S_2} & \mathbf{H}_5^{E_2} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}\end{aligned} \tag{2.67}$$

needed for performing an EIF measurement update according to equation (2.50)

$$\begin{aligned} \hat{\boldsymbol{y}}_{t+1} &= \hat{\boldsymbol{y}}_{t+1|t} + \sum_{m=1}^{2} \mathbf{H}_m^\top \mathbf{R}_m^{-1} \boldsymbol{z}_m \\ \mathbf{Y}_{t+1} &= \mathbf{Y}_{t+1|t} + \sum_{m=1}^{2} \mathbf{H}_m^\top \mathbf{R}_m^{-1} \mathbf{H}_m \end{aligned} \tag{2.68}$$

where $\mathbf{z}_m$ contains all linearized measurement equations as defined in equation (2.23) and $\mathbf{R}_m$ is the measurement covariance as defined in equation (2.24). Since the state variables are considered as measurements, the state vector and covariance can be written [14]

$$\begin{aligned} \hat{\boldsymbol{x}}_{t+1}^{\mathcal{L}} &= \mathbf{C}_{t+1}^{\mathcal{L}} \sum_{m=1}^{2} \mathbf{H}_m^\top \mathbf{R}_m^{-1} \boldsymbol{z}_m \\ \mathbf{C}_{t+1}^{\mathcal{L}} &= \left(\sum_{m=1}^{2} \mathbf{H}_m^\top \mathbf{R}_m^{-1} \mathbf{H}_m \right)^{-1} \end{aligned} \tag{2.69}$$

which reestablishes correlations between features in landmark $\mathcal{L}$.

3

Landmark Extraction

Landmarks are features within sensor data that are used as building blocks for maps. A good landmark should be both easy to find and easy to reobserve [11]. Different landmarks are suited for different mapping environments, and the main objects in parking area scenarios are vehicles. Based on the assumption that the contour of a vehicle is rectangular, lines and corners can be used as landmarks. This chapter presents a method for extracting lines and corners from radar data. The method is inspired by the incremental line fitting algorithm, which achieved good results on lidar data in a comparison study by Nguyen et al. [7]. However, the incremental algorithm makes use of the ordering of lidar detections which radar detections do not have. Therefore, this thesis proposes an alternative algorithm that consists of two clustering methods and orthogonal least square line fitting. The clustering stages are visualized in figure 3.1.

3.1 Single-linkage Clustering

The goals of the first clustering method are to group detections that belong to the same landmark and create an ordering of them. A single-linkage clustering based on Euclidean distances between detections is chosen to achieve those goals. The single-linkage clustering is known to produce long and skinny clusters [12], which is desired when the purpose is finding lines.

As described in 2.1.1, a single-linkage clustering can be obtained by creating a MST and cutting edges longer than a threshold. Two methods for choosing a threshold are tried. The first is to pick a constant threshold based on heuristic knowledge of how closely cars are usually parked. The second method tries to find a more dynamic threshold based on properties of the MST and formulates the threshold as

$$d_{\text{edge}} < \mu + \sigma \tag{3.1}$$

where d_{edge} is the edge length, μ and σ are the mean and standard deviation of the edge lengths in the MST.

The first three pictures in figure 3.1 illustrate how radar detections are processed into a MST, which is turned into clusters by cutting long edges according to the general threshold.

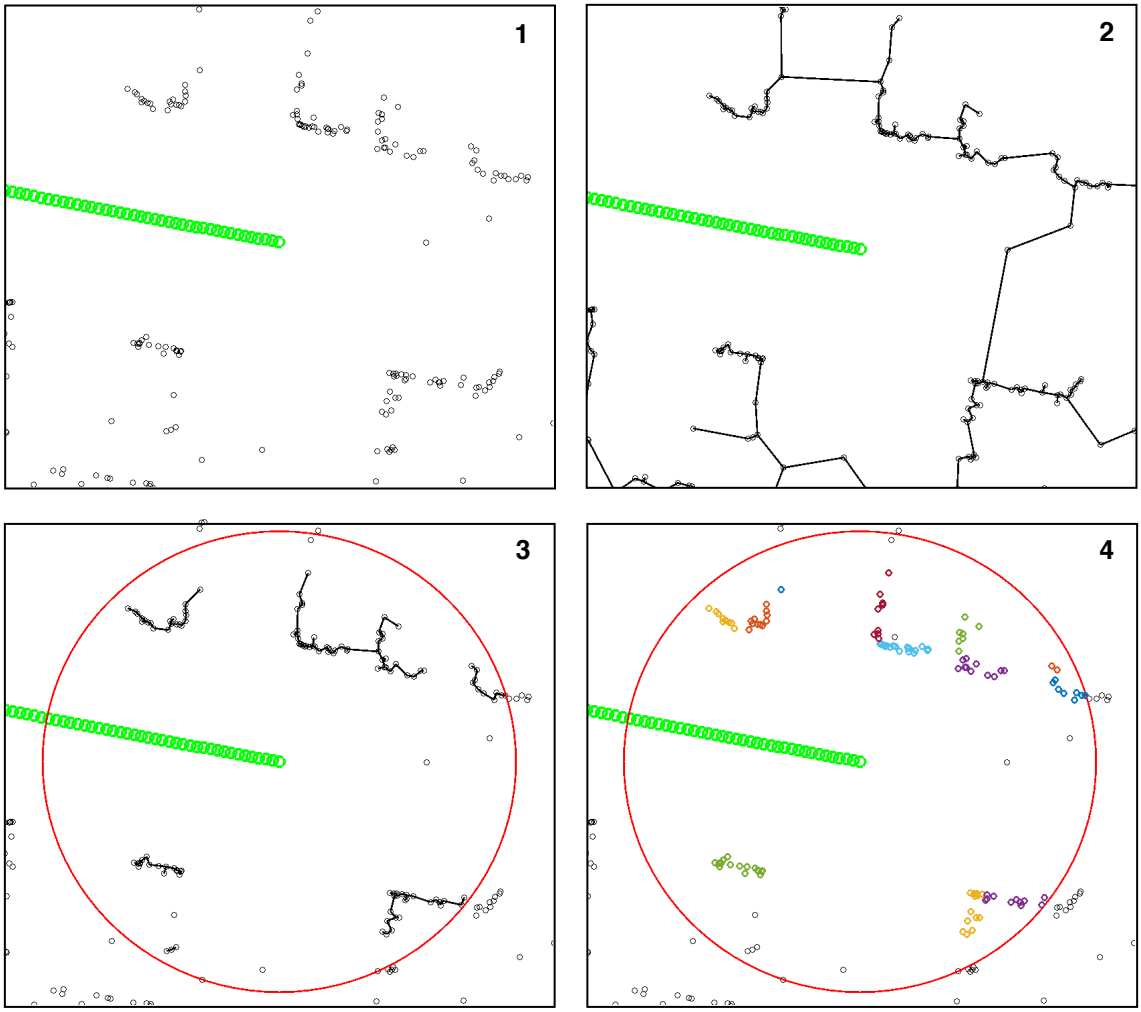


Figure 3.1: *Stages of the landmark extraction process. First, a MST is created based on Euclidean distances between observations. Secondly, long edges are cut by a threshold that depends on mean and standard deviation of all edge lengths, forming clusters. Finally, each cluster is segmented into line clusters using the line segmentation algorithm visualized in figure 3.2.*

3.2 Line Segmentation

Each cluster produced by the MST is segmented into lines. The line segmentation algorithm greedily decides if a point belongs to the current line and recursively starts a new line if it does not. This process is visualized in figure 3.2 and explained in algorithm 1.

The result of the clustering depends on the order in which points are traversed. The graph diameter is used to decide starting point, then a Breadth First Search (BFS) from the starting point decides graph traversal order. In each iteration a point is either added to the current line cluster or not. To be added it must satisfy two conditions: the distance to the closest point and longest eigenvector of the cluster must be below certain thresholds. If conditions are not satisfied a new segmentation

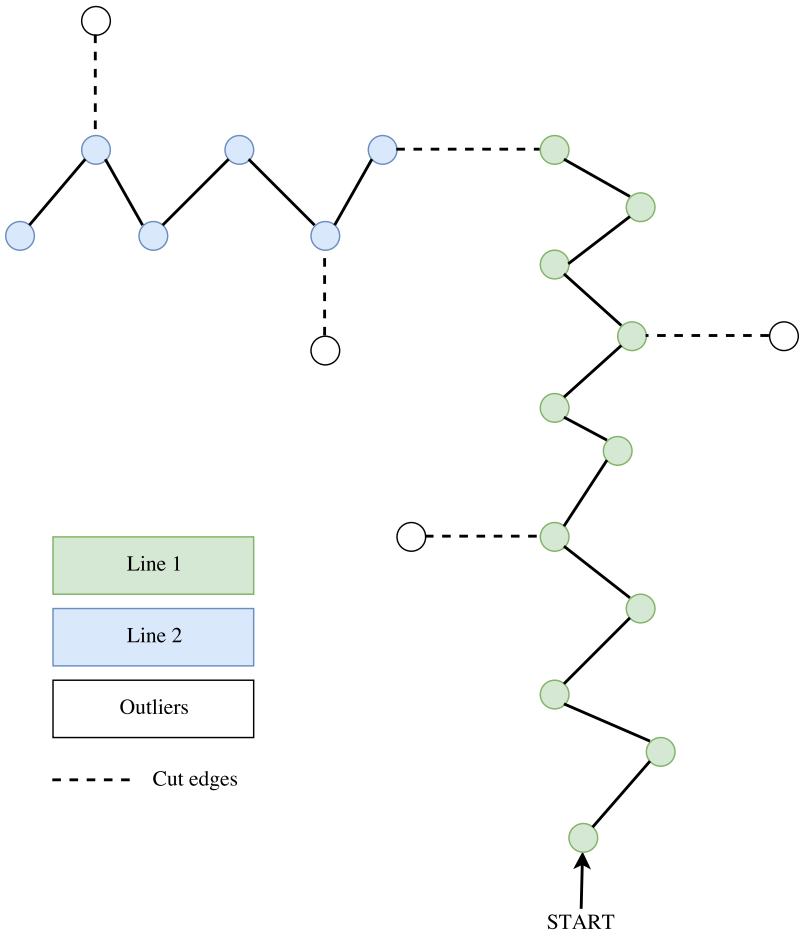


Figure 3.2: *Line clustering algorithm that greedily decides if the next point belongs to the current line or not, and recursively starts a new line if it does not*

is started from the current point in the subtree rooted at this point. Similarly to the single-linkage clustering both constant and dynamic thresholds are tried.

The dynamic threshold for distance to the closest point is based on the normal distribution of the shortest edges in the MST. Long edges are first filtered out from the edge set by partitioning the weights into bins and removing bins for large weights. The threshold is therefore

$$d_{\text{cluster}} < \mu + q\sigma \tag{3.2}$$

where d_{cluster} is the closest point distance, μ and σ are mean and standard deviation of bins containing the shortest edges in the MST and q specifies which quantile to use.

The dynamic threshold for the distance to the longest eigenvector is

$$d_{\text{line}} < \frac{c}{N} + \lambda \tag{3.3}$$

where d_{line} is the orthogonal distance to the longest eigenvector, N the size of the line cluster, λ the smallest eigenvalue of the point distribution, and c is a constant.

Algorithm 1: Line Segmentation

```
Function lineSegments = lineSegmentation(startPoint, MST)
    lineSegment = [startPoint];
    foreach point in BFS(startPoint, MST) do
        d1 = distanceToSegment(point, lineSegment);
        d2 = distanceToLongestEigenvector(point, lineSegment);
        if satisfyCondition(d1, d2) then
            lineSegment.add(point);
        else
            newMST = subtree(point, MST);
            MST = removeSubtree(MST, newMST);
            newSegments = lineSegmentation(point, newMST);
            lineSegments.add(newSegments);
        end
    end
    lineSegments.add(lineSegment);
end
```

When a new line is initialized, the decision of the next point is dominated by (3.2) since (3.3) is large if λ and N are small. However, the more points added to the line cluster the more (3.3) will influence the decision. When the cluster is large enough, (3.3) will mainly depend on λ.

In each iterative step the following must be calculated

- The next point according to a BFS.
- The distance to the closest point in the cluster.
- An update of covariance, eigenvectors and eigenvalues of the cluster after a new point is added.

The BFS is performed in $\mathcal{O}(n+m)$ time where n is the number of points and m the number of edges. Since the graphs are MST's we have that $m = n - 1$. Since some edges are cut during the iterative process, it is most time efficient to calculate one layer of the BFS at the time. The distance to the closest point in the cluster is directly provided from the MST, as it is the length of the edge from the previously visited point. The covariance, eigenvectors and eigenvalues are updated in constant time as explained in appendix A.1. Therefore, the total time complexity of the line segmentation algorithm is $\mathcal{O}(n)$.

3.3 Orthogonal Least Square Line Fitting

The resulting clusters are assumed to represent lines without outliers. Therefore, lines are fitted using the method described in section 2.1.3. If the angle θ between the longest eigenvectors of two adjacent line clusters fulfills the relation $|90 - \theta| < \delta$ they are considered orthogonal. In this case two orthogonal lines are fitted to the

clusters and a corner and two lines are extracted as landmarks.

3.4 Preprocess Detections

The line segmentation method described in section 3.2 works best on clusters that are line or L-shaped. However radar detections do not only exist on surfaces of parked cars, but also within parked cars due to noise. Detections within cars cause clusters to have other shapes than lines and L:s, which can cause the algorithm to extract wrong lines. To improve performance of the algorithm two methods are suggested for making clusters more line or L-shaped.

3.4.1 Graph Diameter Filtering

The first idea for removing outliers is to make use of the diameter path of a cluster's MST. In the left picture of figure 3.3 it can be observed that detections are more dense along the surface of the car and only a few outlier detections exist within the car. In this case the diameter path of the MST will only contain detections from the surface and can be used to create an L-shaped cluster as can be seen in the right picture of figure 3.3.

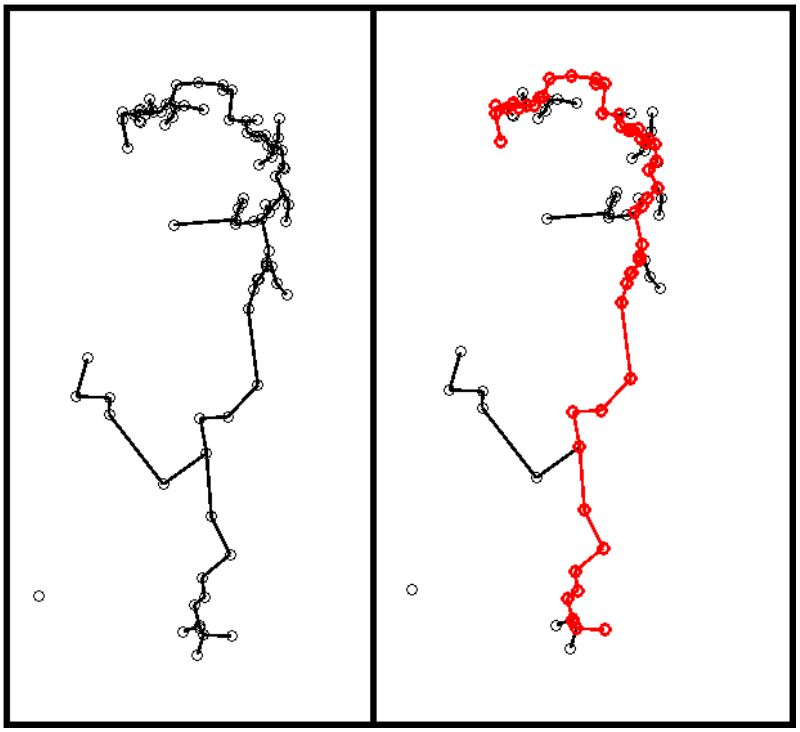


Figure 3.3: *An illustration of how the graph diameter can be used to remove detections within a car. The left pictures shows the MST of a car cluster with some outlier detections within the car. The right picture shows the same cluster with the diameter path highlighted in red.*

3.4.2 Free Space Filtering

The second idea for removing outliers is to remove detections that from the cars perspective are blocked by other detections. By removing blocked detections, the detections on the surface of a car will be kept and those within a car will be removed. In contrast to the previous idea using the diameter this approach filters the detections before the MST is created. The algorithm is described in algorithm 2 and in figure 3.4 can three stages of the filtering be seen. The left picture shows unfiltered detections, the middle picture illustrates how detections are associated to a line, and the last picture shows filtered detections.

Algorithm 2: Free Space Filtering

1. Create lines originating from the car with dθ angular difference between each consecutive line
2. Associate each point to its closest line
3. Project each point to its closest line
4. Sort the projected points along each line
5. For each line keep the projected point closest to the car and all points within distance d from that point

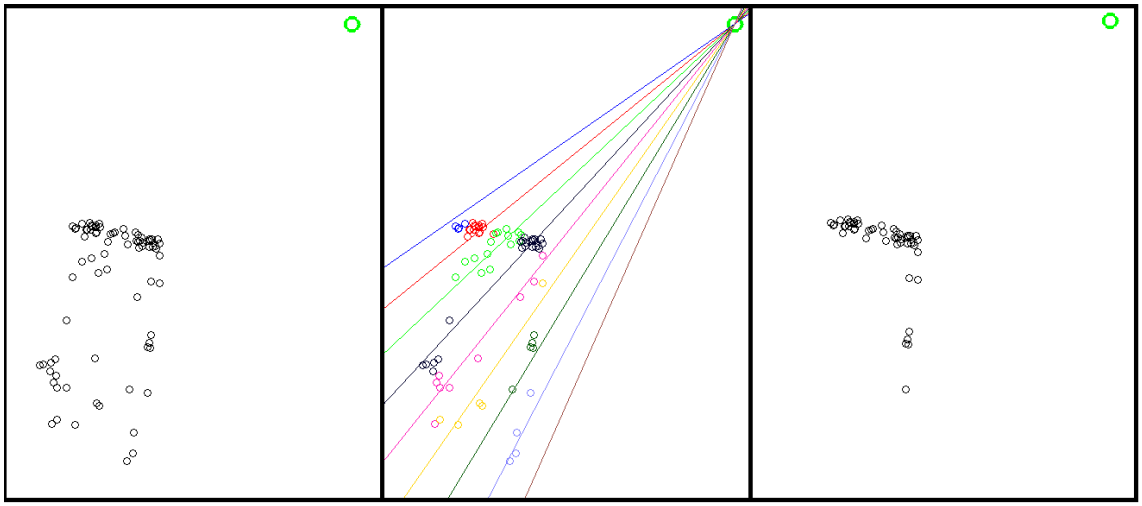


Figure 3.4: *An illustration of the free space algorithm. The left picture shows position of the vehicle (green dot) and unfiltered detections (black dots). The middle picture visualizes lines from the free space algorithm and which detections are associated to which line. Finally the right picture shows filtered detections.*

4

Simultaneous Localization and Mapping

Solving SLAM is a recursive process, shown in figure 4.1, executed at each discrete time step. The process is divided into several steps: time update, landmark extraction, data association and measurement update. The process starts with the time update, which means making an estimation of the next vehicle position based on odometry data according to the vehicle motion model. Then a prediction is made of where landmark locations should be according to the observation model. The prediction is based on the estimation of the new vehicle position from the time update. After reobserving the environment, landmarks are extracted and associated with landmarks from the previous time step. From the predicted and observed landmark locations, estimated vehicle position and landmark locations are updated, which is called the measurement update. This chapter explains the SLAM process for both non-correlated and correlated feature representations.

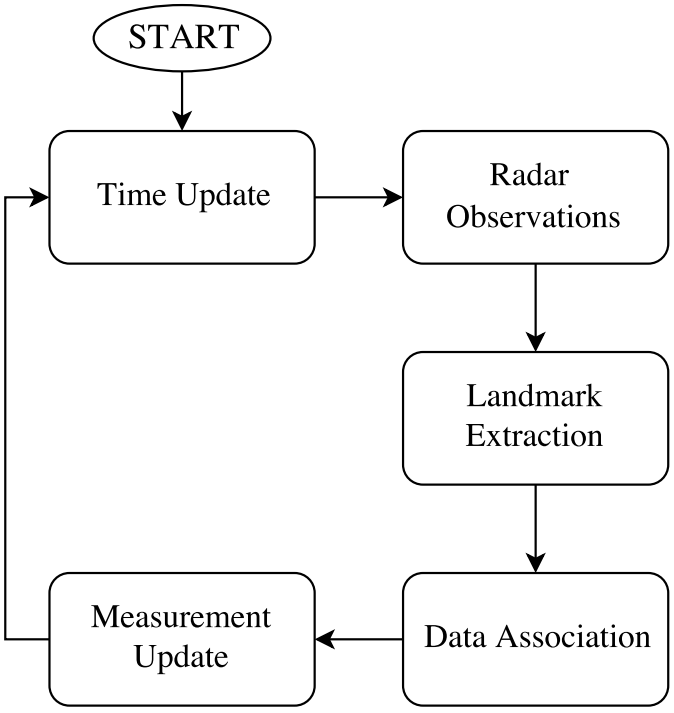


Figure 4.1: *Recursive process for solving SLAM.*

4.1 Non-correlated Feature Representation

This section describes how the general EKF method described in section 2.2.4.1 is applied when using non-correlated features. Non-correlated means that each line is treated as a separate line and it is not taken into account that lines from different sides of a vehicle are dependent.

4.1.1 State Model

Parameters specifying vehicle and landmark positions are stored in a state model. The positions are uncertain and are therefore represented by probability distributions, e.g. Gaussian distributions with means and covariance matrices. The position of each object is represented by an x, y and θ value. The vehicle parameters represent middle position and heading, while the parameters for lines represent middle point of the line and angle between the global x-axis and the line. This is visualized in figure 4.2.

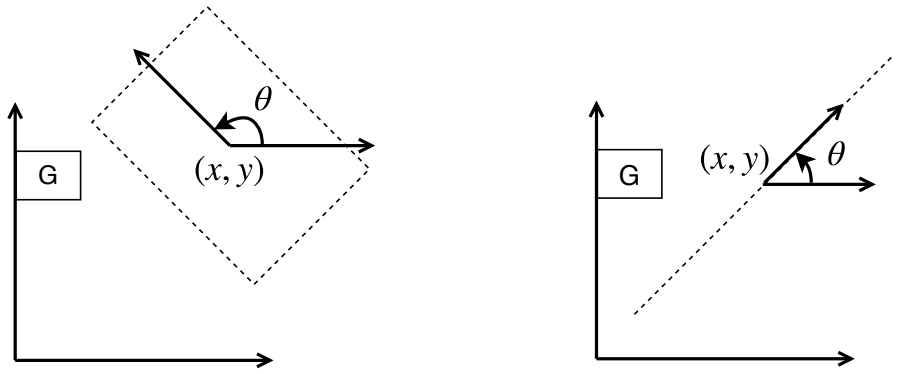


Figure 4.2: *Vehicle (left) and line (right) parameters represented in the global coordinate system.*

Best estimates of state vector $\boldsymbol{x}$ are stored in $\hat{\boldsymbol{x}}$, starting with estimated vehicle position $\hat{\mathcal{V}}$ followed by estimated landmark positions $\hat{\mathcal{L}}_1, \ldots, \hat{\mathcal{L}}_n$ in a global coordinate system $\mathcal{G}$,

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \hat{\mathcal{V}} \\ \hat{\mathcal{L}}_1 \\ \vdots \\ \hat{\mathcal{L}}_n \end{bmatrix}_{\mathcal{G}} \tag{4.1}$$

where

$$\hat{\mathcal{V}} = \begin{bmatrix} x_{\mathcal{G}\mathcal{V}} \\ y_{\mathcal{G}\mathcal{V}} \\ \theta_{\mathcal{G}\mathcal{V}} \end{bmatrix}, \qquad \hat{\mathcal{L}}_i = \begin{bmatrix} x_{\mathcal{G}\mathcal{L}_i} \\ y_{\mathcal{G}\mathcal{L}_i} \\ \theta_{\mathcal{G}\mathcal{L}_i} \end{bmatrix} \tag{4.2}$$

and n is the number of landmarks. The covariance matrix of state vector $\boldsymbol{x}$ includes covariance of vehicle and landmark positions in the diagonal entries, and their cross-covariance in the off-diagonal entries

$$\mathbf{C} = \begin{bmatrix} \Sigma_{\mathcal{V}\mathcal{V}} & \Sigma_{\mathcal{V}\mathcal{L}_1} & \dots & \Sigma_{\mathcal{V}\mathcal{L}_n} \\ \Sigma_{\mathcal{L}_1\mathcal{V}} & \Sigma_{\mathcal{L}_1\mathcal{L}_1} & \dots & \Sigma_{\mathcal{L}_1\mathcal{L}_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{\mathcal{L}_n\mathcal{V}} & \Sigma_{\mathcal{L}_n\mathcal{L}_1} & \dots & \Sigma_{\mathcal{L}_n\mathcal{L}_n} \end{bmatrix} \tag{4.3}$$

where

$$\Sigma_{ij} = \begin{bmatrix} \sigma_{x_i}\sigma_{x_j} & \sigma_{x_i}\sigma_{y_j} & \sigma_{x_i}\sigma_{\theta_j} \\ \sigma_{y_i}\sigma_{x_j} & \sigma_{y_i}\sigma_{y_j} & \sigma_{y_i}\sigma_{\theta_j} \\ \sigma_{\theta_i}\sigma_{x_j} & \sigma_{\theta_i}\sigma_{y_j} & \sigma_{\theta_i}\sigma_{\theta_j} \end{bmatrix} \tag{4.4}$$

Since all landmarks are lines in this representation an additional vector $\boldsymbol{\ell}$ is needed for line lengths, as suggested by Garulli et al. [5],

$$\boldsymbol{\ell} = \begin{bmatrix} l_1 \\ \vdots \\ l_n \end{bmatrix} \tag{4.5}$$

where l_i is the length of landmark $\mathcal{L}_i$.

The state model is used during implementation of the map building process described in the following sections.

4.1.2 Time Update

Uncertain locations of vehicle and landmarks are stored in the state model described in section 4.1.1. The state model is updated in every time step. First, only the vehicle position is updated and therefore entries in the state model that include landmark positions remain the same. The updated vehicle position is calculated from the previous state using vehicle motion model

$$\mathbf{f}(x_t, y_t, \theta_t) = \begin{cases} x_{t+1} = x_t + v_t \cos(\theta_t + \theta_s)\ \mathrm{d}t \\ y_{t+1} = y_t + v_t \sin(\theta_t + \theta_s)\ \mathrm{d}t \\ \theta_{t+1} = \theta_t + \dot{\theta}_t\ \mathrm{d}t \end{cases} \tag{4.6}$$

where v_t is velocity of the vehicle at time t, θ_s is slip angle (angle between heading and velocity vector) and $\dot{\theta}_t = \frac{\mathrm{d}\theta_t}{\mathrm{d}t}$ yaw rate. The updated $\hat{\boldsymbol{x}}$ is therefore

$$\hat{\boldsymbol{x}}_{t+1|t} = \begin{bmatrix} \mathbf{f}(x_t, y_t, \theta_t) \\ \hat{\mathcal{L}}_1 \\ \vdots \\ \hat{\mathcal{L}}_n \end{bmatrix} \tag{4.7}$$

The covariance matrix is updated by propagating errors from the previous state and adding odometry errors. The Jacobian matrix is used to propagate uncertainties for non-linear functions (for further reading regarding uncertainty propagation see appendix A.2).

The Jacobian matrix $\mathbf{F}_V = \nabla \mathbf{f}$ of the vehicle motion model is

$$\mathbf{F}_V = \begin{bmatrix} 1 & 0 & -v_t \sin(\theta_t + \theta_s) \mathrm{d}t \\ 0 & 1 & v_t \cos(\theta_t + \theta_s) \mathrm{d}t \\ 0 & 0 & 1 + \ddot{\theta}\, \mathrm{d}t \end{bmatrix} \tag{4.8}$$

where the angular acceleration $\ddot{\theta}$ is assumed to be zero. The state vector update in equation (4.7) only updates estimated vehicle position. When updating the covariance matrix, however, cross-covariances of vehicle and landmarks are also updated. Therefore Jacobian $\mathbf{F}$ with respect to the entire state vector is

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_V & \mathbf{0}_{3\times n} \\ \mathbf{0}_{n\times 3} & \mathbf{I}_{n\times n} \end{bmatrix} \tag{4.9}$$

The covariance matrix is therefore updated using propagation of uncertainty for the non-linear case (see appendix A.2), and adding uncertainties from using odometry data for velocity, slip angle and yaw rate

$$\mathbf{C}_{t+1|t} = \mathbf{F}\mathbf{C}_{t|t}\mathbf{F}^\top + \mathbf{G}\mathbf{C}_{\text{odometry}}\mathbf{G}^\top \tag{4.10}$$

where

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_{3\times 3} \\ \mathbf{0}_{n\times 3} \end{bmatrix}, \tag{4.11}$$

This finalizes the time update. The next step is the measurement update in which landmarks are reobserved and used to update the entire state model.

4.1.3 Measurement Update

The entire state model is updated during the measurement update. This step iterates through all newly extracted features and associates them to landmarks within the current state model according to the method described in section 4.1.5. Since landmarks in the state model are in the global coordinate system they are transformed into the vehicle reference frame using measurement model

$$\mathcal{L}_\mathcal{V} = \mathbf{h}_m(\mathcal{V}_\mathcal{G}, \mathcal{L}_\mathcal{G}) = \begin{bmatrix} \cos(\theta_{\mathcal{G}\mathcal{V}}) & -\sin(\theta_{\mathcal{G}\mathcal{V}}) & 0 \\ \sin(\theta_{\mathcal{G}\mathcal{V}}) & \cos(\theta_{\mathcal{G}\mathcal{V}}) & 0 \\ 0 & 0 & 1 \end{bmatrix}^\top \left(\begin{bmatrix} x_{\mathcal{G}\mathcal{L}} \\ y_{\mathcal{G}\mathcal{L}} \\ \theta_{\mathcal{G}\mathcal{L}} \end{bmatrix} - \begin{bmatrix} x_{\mathcal{G}\mathcal{V}} \\ y_{\mathcal{G}\mathcal{V}} \\ \theta_{\mathcal{G}\mathcal{V}} \end{bmatrix} \right) \tag{4.12}$$

To propagate errors during this transformation, the Jacobian matrix $\mathbf{H}_m = \nabla \mathbf{h}_m$ is calculated with respect to the state vector $\hat{\boldsymbol{x}}$. Since the only objects involved

during one iteration is the vehicle position and the current landmark, the partial derivatives with respect to all other landmarks are zero. Therefore $\mathbf{H}_m$ is divided into two Jacobian matrices $\mathbf{H}_\mathcal{V}$ and $\mathbf{H}_\mathcal{L}$ with respect to the vehicle and landmark parameters, respectively. The Jacobians are

$$\mathbf{H}_\mathcal{V} = \begin{bmatrix} -\cos(\theta_{\mathcal{GV}}) & -\sin(\theta_{\mathcal{GV}}) & -(x_{\mathcal{GL}} - x_{\mathcal{GV}})\sin(\theta_{\mathcal{GV}}) + (y_{\mathcal{GL}} - y_{\mathcal{GV}})\cos(\theta_{\mathcal{GV}}) \\ \sin(\theta_{\mathcal{GV}}) & -\cos(\theta_{\mathcal{GV}}) & -(x_{\mathcal{GL}} - x_{\mathcal{GV}})\cos(\theta_{\mathcal{GV}}) - (y_{\mathcal{GL}} - y_{\mathcal{GV}})\sin(\theta_{\mathcal{GV}}) \\ 0 & 0 & -1 \end{bmatrix} \tag{4.13}$$

$$\mathbf{H}_\mathcal{L} = \begin{bmatrix} \cos(\theta_{\mathcal{GV}}) & \sin(\theta_{\mathcal{GV}}) & 0 \\ -\sin(\theta_{\mathcal{GV}}) & \cos(\theta_{\mathcal{GV}}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.14}$$

The matrix $\mathbf{H}_m$ is formed by inserting $\mathbf{H}_\mathcal{V}$ and $\mathbf{H}_\mathcal{L}$ at their proper indices

$$\mathbf{H}_m = \begin{bmatrix} \mathbf{H}_\mathcal{V} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{H}_\mathcal{L} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix} \tag{4.15}$$

Assuming correct landmark extraction and association, the state model is updated based on the difference between the previous and reobserved landmark position. This difference is called the innovation

$$\boldsymbol{v}_m = \boldsymbol{z}_m - \hat{\boldsymbol{z}}_m \tag{4.16}$$

where $\boldsymbol{z}_m$ is the observation and $\hat{\boldsymbol{z}}_m = \mathbf{h}(\hat{\boldsymbol{x}})$. The covariance of the innovation is

$$\mathbf{S}_m = \mathbf{H}_m \mathbf{C}_{t+1|t} \mathbf{H}_m^\top + \mathbf{R}_m \tag{4.17}$$

which is propagated uncertainty of $\boldsymbol{z}_m$ through the measurement model and $\mathbf{R}_m$ is uncertainty of the new observation based on uncertainty in the radar. The measurement update is performed by calculating the Kalman gain

$$\mathbf{K}_m = \mathbf{C}_{t+1|t} \mathbf{H}_m^\top \mathbf{S}_m^{-1} \tag{4.18}$$

and updating the state model according to the EKF equations

$$\hat{\boldsymbol{x}}_{t+1} = \hat{\boldsymbol{x}}_{t+1|t} + \mathbf{K}_m \boldsymbol{v}_m \tag{4.19}$$

$$\mathbf{C}_{t+1} = (\mathbf{I} - \mathbf{K}_m \mathbf{H}_m) \mathbf{C}_{t+1|t} \tag{4.20}$$

The length of the updated line is calculated in three steps. First endpoints of the observed line $\hat{\mathbf{z}}_m$ and endpoint of the previous line $\mathcal{L}_t$ are projected onto the updated line $\mathcal{L}_{t+1}$. Then the longest segment between the projected endpoints is found. Finally the new length is chosen to be as long as possible while still being contained within the segment.

4.1.4 Adding New Features to State Model

New features that were not associated to any of the current landmarks within the state model are considered new landmarks. New landmarks are added to the state model by the process described in this section. Given a feature and the current vehicle state

$$\hat{\mathcal{L}}_{\mathcal{V}} = \begin{bmatrix} x_{\mathcal{V}\mathcal{L}} \\ y_{\mathcal{V}\mathcal{L}} \\ \theta_{\mathcal{V}\mathcal{L}} \end{bmatrix}, \qquad \hat{\mathcal{V}} = \begin{bmatrix} x_{\mathcal{G}\mathcal{V}} \\ y_{\mathcal{G}\mathcal{V}} \\ \theta_{\mathcal{G}\mathcal{V}} \end{bmatrix} \tag{4.21}$$

the first step is to transform the feature from the vehicle reference frame to the world reference frame. That is accomplished by applying $\mathbf{h}_m^{-1}$, which is the inverse of the measurement model defined in equation 4.12.

$$\hat{\mathcal{L}}_{\mathcal{G}} = \mathbf{h}_m^{-1}(\hat{\mathcal{L}}_{\mathcal{V}}, \hat{\mathcal{V}}_{\mathcal{G}}) = \begin{bmatrix} x_{\mathcal{G}\mathcal{L}} \\ y_{\mathcal{G}\mathcal{L}} \\ \theta_{\mathcal{G}\mathcal{L}} \end{bmatrix} = \begin{bmatrix} \cos(\theta_{\mathcal{G}\mathcal{V}}) & -\sin(\theta_{\mathcal{G}\mathcal{V}}) & 0 \\ \sin(\theta_{\mathcal{G}\mathcal{V}}) & \cos(\theta_{\mathcal{G}\mathcal{V}}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{\mathcal{V}\mathcal{L}} \\ y_{\mathcal{V}\mathcal{L}} \\ \theta_{\mathcal{V}\mathcal{L}} \end{bmatrix} + \begin{bmatrix} x_{\mathcal{G}\mathcal{V}} \\ y_{\mathcal{G}\mathcal{V}} \\ \theta_{\mathcal{G}\mathcal{V}} \end{bmatrix} \tag{4.22}$$

The next step is to append the transformed feature to the current state model $\hat{\boldsymbol{x}}_{t+1}$ and its length l to the length vector $\boldsymbol{\ell}$.

$$\hat{\boldsymbol{x}}_{t+1} = \begin{bmatrix} \hat{\boldsymbol{x}}_{t+1} \\ \hat{\mathcal{L}}_{\mathcal{G}} \end{bmatrix}, \qquad \boldsymbol{\ell} = \begin{bmatrix} \boldsymbol{\ell} \\ l \end{bmatrix} \tag{4.23}$$

Finally the covariance $\mathbf{C}_{t+1}$ is augment to $\mathbf{C}'_{t+1}$ as proposed by Collier [11] using Jacobians $\mathbf{J}_{\mathcal{V}}$ and $\mathbf{J}_{\mathcal{L}}$ of $\mathbf{h}_m^{-1}$ with respect to $\hat{\mathcal{V}}$ and $\hat{\mathcal{L}}_{\mathcal{V}}$, respectively .

$$\mathbf{J}_{\mathcal{V}} = \begin{bmatrix} 1 & 0 & -x_{\mathcal{V}\mathcal{L}}\sin(\theta_{\mathcal{G}\mathcal{V}}) - y_{\mathcal{V}\mathcal{L}}\cos(\theta_{\mathcal{G}\mathcal{V}}) \\ 0 & 1 & x_{\mathcal{V}\mathcal{L}}\cos(\theta_{\mathcal{G}\mathcal{V}}) - y_{\mathcal{V}\mathcal{L}}\sin(\theta_{\mathcal{G}\mathcal{V}}) \\ 0 & 0 & 1 \end{bmatrix} \tag{4.24}$$

$$\mathbf{J}_{\mathcal{L}} = \begin{bmatrix} \cos(\theta_{\mathcal{G}\mathcal{V}}) & -\sin(\theta_{\mathcal{G}\mathcal{V}}) & 0 \\ \sin(\theta_{\mathcal{G}\mathcal{V}}) & \cos(\theta_{\mathcal{G}\mathcal{V}}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.25}$$

$$\mathbf{C}_{t+1} = \begin{pmatrix} \mathbf{C}_{\mathcal{V}} & \mathbf{C}_{\mathcal{V}\mathcal{M}} \\ \mathbf{C}_{\mathcal{V}\mathcal{M}}^{\top} & \mathbf{C}_{\mathcal{M}} \end{pmatrix} \tag{4.26}$$

$$\mathbf{C}'_{t+1} = \begin{pmatrix} \mathbf{C}_{\mathcal{V}} & \mathbf{C}_{\mathcal{V}\mathcal{M}} & \mathbf{C}_{\mathcal{V}}^{\top}\mathbf{J}_{\mathcal{V}}^{\top} \\ \mathbf{C}_{\mathcal{V}\mathcal{M}}^{\top} & \mathbf{C}_{\mathcal{M}} & \mathbf{C}_{\mathcal{V}\mathcal{M}}^{\top}\mathbf{J}_{\mathcal{V}}^{\top} \\ \mathbf{J}_{\mathcal{V}}\mathbf{C}_{\mathcal{V}} & \mathbf{J}_{\mathcal{V}}\mathbf{C}_{\mathcal{V}\mathcal{M}} & \mathbf{J}_{\mathcal{V}}\mathbf{C}_{\mathcal{V}}\mathbf{J}_{\mathcal{V}}^{\top} + \mathbf{J}_{\mathcal{L}}\mathbf{R}_m\mathbf{J}_{\mathcal{L}}^{\top} \end{pmatrix} \tag{4.27}$$

After the addition of new landmarks is complete landmarks that have been observed too few times during a given time period are removed. Thereby the number of spurious landmarks within the state model is reduced.

This completes the state model update, and the process continues recursively to the next time frame.

4.1.5 Data Association

When new landmarks have been extracted they need to be associated with landmarks viewed from previous observations, in order to update the state model during the measurement update phase. This association process has to deal with several difficulties

- All previous landmarks are not reobserved at every time step
- Spurious landmarks can exist among both previous and new landmarks
- Wrongly associating a new landmark to a previous one increases errors within the map

A data association algorithm consists of two components [8], a compatibility test that determines if two landmarks can be associated, and a criterion for deciding which of the compatible matchings is the best matching. A common approach within SLAM is to use a gated nearest neighbor algorithm which is adopted for this SLAM scenario. The gated nearest neighbor approach passes each possible association-pair through a number of validation gates that decide if the matchings are compatible. Among the compatible matchings the one with closest landmarks is chosen. If no compatible matchings exist the landmark is considered new and is added to the state model.

For the case when landmarks are lines, three validation gates similar to the ones used by Garulli et al. [5] are used.

- Angle gate: An association-pair passes the angle gate if the angle between the direction vectors of two lines is less than a threshold.
- Midpoint gate: An association-pair passes the midpoint gate if the orthogonal distance from the midpoint of the new line to the previous line is below a threshold
- Overlap gate: An association-pair passes the overlap gate if the portion of the new line that overlaps the previous line is greater than a threshold. Or it passes if the shortest distance between any of the lines endpoints is below a threshold.

The best matching is chosen to be the one with smallest Mahalanobis distance, as suggested by Neira et al. [8]. The Mahalanobis distance is computed as follows

$$D^2 = \boldsymbol{v}_i \mathbf{S}_i^{-1} \boldsymbol{v}_i^\top \tag{4.28}$$

where $\boldsymbol{v}_i$ is the innovation and $\mathbf{S}_i$ the innovation covariance, which are defined in equation 4.16 respectively 4.17.

4.2 Correlated Feature Representation

Using the SPmodel introduced in section 2.2.2.2 makes it possible to include correlations between features. This is desirable in the parking area scenario since sides of vehicles depend on each other. Reobserving one side of a vehicle should propagate

to the other sides, updating the estimated location of them as well. Including correlations makes it possible to utilize information obtained from different angles for updating the vehicle position. Therefore, the estimated positions of the cars should be more certain than using a model that neglects correlations as presented in section 4.1.

4.2.1 State Model

The state model in the previous section contains positions and angles of lines in a global coordinate system. The state model used in this section, however, contains perturbation vectors of different types of features: endpoints, segments and corners. This representation by Castellanos et al. [14] is called a SPmodel and is introduced in section 2.2.2.2.

$$\hat{\mathbf{p}} = \begin{bmatrix} \hat{\mathbf{d}}_{\mathcal{V}} \\ \hat{\mathbf{p}}_{\mathcal{F}_1} \\ \vdots \\ \hat{\mathbf{p}}_{\mathcal{F}_n} \end{bmatrix} \tag{4.29}$$

where $\mathcal{F}_1, \ldots, \mathcal{F}_n$ are features such as endpoints, segments and corners. The state covariance is

$$\mathbf{C} = \begin{bmatrix} \Sigma_{\mathcal{V}\mathcal{V}} & \Sigma_{\mathcal{V}\mathcal{F}_1} & \cdots & \Sigma_{\mathcal{V}\mathcal{F}_n} \\ \Sigma_{\mathcal{F}_1\mathcal{V}} & \Sigma_{\mathcal{F}_1\mathcal{F}_1} & \cdots & \Sigma_{\mathcal{F}_1\mathcal{F}_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{\mathcal{F}_n\mathcal{V}} & \Sigma_{\mathcal{F}_n\mathcal{F}_1} & \cdots & \Sigma_{\mathcal{F}_n\mathcal{F}_n} \end{bmatrix} \tag{4.30}$$

$$\Sigma_{ij} = E[(\mathbf{p}_i - \hat{\mathbf{p}}_i)(\mathbf{p}_j - \hat{\mathbf{p}}_j)^\top] \tag{4.31}$$

This state model is used during the implementation of the map building process described in the following sections

4.2.2 Time Update

As mentioned in previous sections, only the vehicle position is updated during the time update. Therefore, the time update for this representation is similar to the process described in section 4.1.2. The vehicle position in the next time step is estimated using the vehicle motion model, which can be seen as a relative location vector [14]

$$\hat{\mathbf{x}}_{GV_{t+1|t}} = \hat{\mathbf{x}}_{GV_{t|t}} \oplus \mathbf{x}_{V_t V_{t+1}} \tag{4.32}$$

where G is the global coordinate system and V is the reference frame attached to vehicle $\mathcal{V}$.

During that process, the vehicle position in the next time step is predicted using odometry data, and the covariance is propagated to the next state by calculating the Jacobian of the vehicle motion model (equation (4.8)). Propagation of uncertainty can be further read about in Appendix A.2.

When using this model, however, the covariance of the vehicle is not represented in its location variables. Recall from section 2.2.2.2 that uncertain locations of geometric objects are represented by estimated location vectors and normally distributed error vectors, called differential location vectors. Therefore, the vehicle position, which is defined with its local reference frame V, in a global reference G frame is

$$\mathbf{x}_{GV} = \hat{\mathbf{x}}_{GV} \oplus \mathbf{d}_V \tag{4.33}$$

The covariance is therefore the uncertainty of the differential location vector $\mathbf{d}_V$ of the vehicle

$$\mathbf{C}_V = E[(\mathbf{d}_V - \hat{\mathbf{d}}_V)(\mathbf{d}_V - \hat{\mathbf{d}}_V)^\top] \tag{4.34}$$

Using this model, the covariance of $\mathbf{d}_V$ is propagated to the next state using the Jacobian $\mathbf{J}_{V_{t+1},V_t}$ of the transformation $\mathbf{x}_{V_t,V_{t+1}}$, see Appendix A.3.3

$$\mathbf{d}_{V_{t+1|t}} = \mathbf{J}_{V_{t+1},V_t}\mathbf{d}_{V_{t|t}} \tag{4.35}$$

Odometry errors $\mathbf{d}_{\mathcal{V}_k,\mathcal{V}_{k+1}}$ also need to be added to conclude the error vector for the next state

$$\mathbf{d}_{V_{t+1|t}} = \mathbf{J}_{V_{t+1},V_t}\mathbf{d}_{V_{t|t}} + \mathbf{d}_{V_t,V_{t+1}} \tag{4.36}$$

From this its possible to form matrices for the perturbation vector of the entire state model

$$\mathbf{p}_{t+1} = \mathbf{F}\mathbf{p}_t + \mathbf{G}\mathbf{d}_{V_t,V_{t+1}} \tag{4.37}$$

where

$$\begin{aligned} \mathbf{F} &= \begin{pmatrix} \mathbf{J}_{V_{t+1},V_t} & \mathbf{O}_{3\times n} \\ \mathbf{O}_{n\times 3} & \mathbf{I}_{n\times n} \end{pmatrix} \\ \mathbf{G} &= \begin{pmatrix} \mathbf{I}_{3\times 1} \\ \mathbf{0}_{n\times 1} \end{pmatrix} \end{aligned} \tag{4.38}$$

Using the matrices in (4.38) Kalman equations can be formed

$$\begin{aligned} \hat{\mathbf{p}}_{t+1|t} &= \mathbf{F}\hat{\mathbf{p}}_t \\ \mathbf{C}_{t+1|t} &= \mathbf{F}\mathbf{C}_t\mathbf{F} + \mathbf{G}\mathbf{C}_{V_t,V_{t+1}}\mathbf{G} \end{aligned} \tag{4.39}$$

concluding the time update

4.2.3 Measurement Update

The first step of the measurement update step is similar to the measurement update prodecure explained in section 4.1.3, however it requires additional steps. After the Kalman measurement update, correlations are reestablished using an EIF as explained in section 2.2.4.2. The estimated positions $\hat{\mathbf{x}}$ are also centered, i.e. moved to the mean of the perturbation vector, creating zero mean perturbations of all features.

4.2.3.1 Measurement Integration

EKF equations and the linearized measurement equation (2.23) gives the measurement update [14]

$$\begin{aligned} \hat{\mathbf{p}}_{t+1} &= \hat{\mathbf{p}}_{t+1|t} + \mathbf{K}_m(\mathbf{z}_m - \mathbf{H}_m\hat{\mathbf{p}}_{t+1|t}) \\ &= \hat{\mathbf{p}}_{t+1|t} - \mathbf{K}_m\mathbf{h}_m \\ \mathbf{C}_{t+1} &= (\mathbf{I} - \mathbf{K}_m\mathbf{H}_m)\mathbf{C}_{t+1|t} \end{aligned} \tag{4.40}$$

where

$$\begin{aligned} \mathbf{K}_m &= \mathbf{C}_{t+1|t}\mathbf{H}_m^\top\mathbf{S}_m^{-1} \\ \mathbf{S}_m &= \mathbf{H}_m\mathbf{C}_{t+1|t}\mathbf{H}_m^\top + \mathbf{R}_m \end{aligned} \tag{4.41}$$

are Kalman gain and innovation covariance. Note that the innovation $\boldsymbol{v}_m$ as defined in equation (2.44) here equals the negative measurement equation of paired features $\mathbf{h}_m$, which can be understood from studying the coefficients of the linearized measurement equation (2.24).

4.2.3.2 Correlations Update

After each measurement integration, all other features within the same landmark are updated to reestablish correlations. This is done using an EIF as explained in section 2.2.4.2. Consider a state vector $\hat{\mathbf{p}}^{\mathcal{L}}$ including the perturbation vectors of features for a L-shaped landmark

$$\mathcal{L} = \begin{Bmatrix} \mathcal{E}_1 & \mathcal{S}_1 & \mathcal{C} & \mathcal{S}_2 & \mathcal{E}_2 \end{Bmatrix} \tag{4.42}$$

Also consider that a measurement of segment $\mathcal{S}_1$ was integrated using the Kalman measurement update explained in the previous section. The entire state vector is updated to maintain correlations using equation (2.67) and (2.69)

$$\begin{aligned}\hat{\mathbf{p}}_{t+1}^{\mathcal{L}} &= \mathbf{C}_{t+1}^{\mathcal{L}} \sum_{m=1}^{M} \mathbf{H}_m \mathbf{R}_m^{-1} \boldsymbol{z}_m \\ \mathbf{C}_{t+1}^{\mathcal{L}} &= \left(\sum_{m=1}^{M} \mathbf{H}_m \mathbf{R}_m^{-1} \mathbf{H}_m^{\top} \right)^{-1}\end{aligned} \tag{4.43}$$

where $\boldsymbol{z}_m$, $m \in \{1, ..., M\}$ are linearized measurement equations used to relate features as explained in section 2.2.4.2. Recall that landmark features are considered measurements in this estimation process, hence no new observations are used.

4.2.3.3 Center Estimated Positions

Since the state vector in the SPmodel does not contain the estimated positions of features, but rather error vectors of the estimated positions, the estimated positions need to be centered after each measurement update [14]. Centering the state vector means transforming each estimated location vector $\hat{\boldsymbol{x}}$ with the mean of its perturbation vector $\hat{\mathbf{p}}$

$$\hat{\boldsymbol{x}}'_{t+1} = \begin{bmatrix} \hat{\mathbf{x}}_{GV} \oplus \hat{\mathbf{d}}_V \\ \hat{\mathbf{x}}_{GF} \oplus \mathbf{B}_F^{\top} \hat{\mathbf{d}}_F \end{bmatrix} \tag{4.44}$$

and propagating the uncertainty to the new reference frames

$$\mathbf{C}'_{t+1} = \mathbf{Q} \mathbf{C}_{t+1} \mathbf{Q}^{\top} \tag{4.45}$$

where

$$\mathbf{Q} = \begin{pmatrix} \mathbf{B}_V \mathbf{J}_{2\oplus}^{-1}\{\mathbf{d}_V, \mathbf{0}\} \mathbf{B}_V^{\top} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_F \mathbf{J}_{2\oplus}^{-1}\{\mathbf{B}_F^{\top}, \mathbf{p}_F, \mathbf{0}\} \mathbf{B}_F^{\top} \end{pmatrix} \tag{4.46}$$

The mean of the perturbation vectors also need to be set to zero

$$\hat{\mathbf{p}}'_{t+1} = \mathbf{0} \tag{4.47}$$

4.2.4 Adding New Features to State Model

Features that was not used as measurements are considered new features and need to be added to the state vector. Since features are observed from the vehicles reference frame the initial estimated position of a new feature is [14]

$$\hat{\mathbf{x}}_{GF} = \hat{\mathbf{x}}_{GV} \oplus \hat{\mathbf{x}}_{VF} \tag{4.48}$$

and the uncertainty of the feature depend on the uncertainty of the vehicle, hence the covariance matrix is appended as

$$\mathbf{C}'_{t+1} = \begin{pmatrix} \mathbf{C}_{\mathcal{V}} & \mathbf{C}_{\mathcal{V}\mathcal{M}} & \mathbf{C}_{\mathcal{V}}\mathbf{J}_{FV}^{\top}\mathbf{B}_{F}^{\top} \\ \mathbf{C}_{\mathcal{V}\mathcal{M}}^{\top} & \mathbf{C}_{\mathcal{M}} & \mathbf{C}_{\mathcal{V}\mathcal{M}}\mathbf{J}_{FV}^{\top}\mathbf{B}_{F}^{\top} \\ \mathbf{B}_{F}\mathbf{J}_{FV}\mathbf{C}_{\mathcal{V}} & \mathbf{B}_{F}\mathbf{J}_{FV}\mathbf{C}_{\mathcal{V}\mathcal{M}} & \mathbf{B}_{F}\mathbf{J}_{FV}\mathbf{C}_{\mathcal{V}}\mathbf{J}_{FV}^{\top}\mathbf{B}_{F}^{\top} + \mathbf{C}_{\mathcal{F}} \end{pmatrix} \tag{4.49}$$

4.2.5 Data Association

Similar to section 4.1.5, new landmarks need to be associated with landmarks in the state model to be able to perform the measurement update phase. In this model landmarks consist of multiple feature, and therefore compatibility of all features improves the association [14]. Compatibility of individual features is done by the pairing process described in section 2.2.3.2. A joint compatibility check of all features is performed by stacking coefficients, obtained from equation 2.32, of all features within a landmark. The measurement equations at the linearization point are therefore

$$\mathbf{h}_m = \begin{bmatrix} \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_{N_f} \end{bmatrix} \tag{4.50}$$

where N is the number of features within the landmark. Similarly, the coefficients for the state vector form

$$\mathbf{H}_m = \begin{pmatrix} \mathbf{H}_{\mathcal{V}_1} & \mathbf{H}_{\mathcal{F}_1} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ \mathbf{H}_{\mathcal{V}_{N_f}} & \mathbf{0} & \dots & \mathbf{H}_{\mathcal{F}_{N_f}} & \dots & \mathbf{0} \end{pmatrix} \tag{4.51}$$

and coefficients for the measurements

$$\mathbf{G}_m = \begin{pmatrix} \mathbf{G}_{\mathcal{E}_1} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{G}_{\mathcal{E}_{N_f}} \end{pmatrix} \tag{4.52}$$

Landmarks are considered compatible if the $\chi^2_{r,\alpha}$ test described in section 2.2.3.3 is passed.

5

Test and Verification

This chapter describes how performance of the landmark extraction and map building algorithms, introduced in chapter 3 and 4, is evaluated against lidar data. First the test vehicle and sensor setup are described, followed by a presentation of test scenarios. Finally the evaluation of landmark extraction and map building is explained.

5.1 Test Setup

To test the developed algorithms on real world data, a car like the one in figure 5.1 is used. The car is equipped with one radar sensor in front and two on each side, as well as a lidar sensor on the roof. Each radar sensor provides detections $d_{\text{polar}} = \begin{bmatrix} \theta & r \end{bmatrix}$ of surrounding objects, where θ is angle and r range to an object in polar coordinates from the sensor. Due to sensor characteristics a detection contains noise in both angle and range, which is assumed to be Gaussian with covariance matrix $\mathbf{C}_{\text{polar}}$. To facilitate the processing of detections they are transformed to a Cartesian coordinate system centered in the front bumper of the car. The transformation is described in A.4, which produces detections represented by $d_{\text{cart}} = \begin{bmatrix} x & y \end{bmatrix}$ and covariance $\mathbf{C}_{\text{cart}}$.

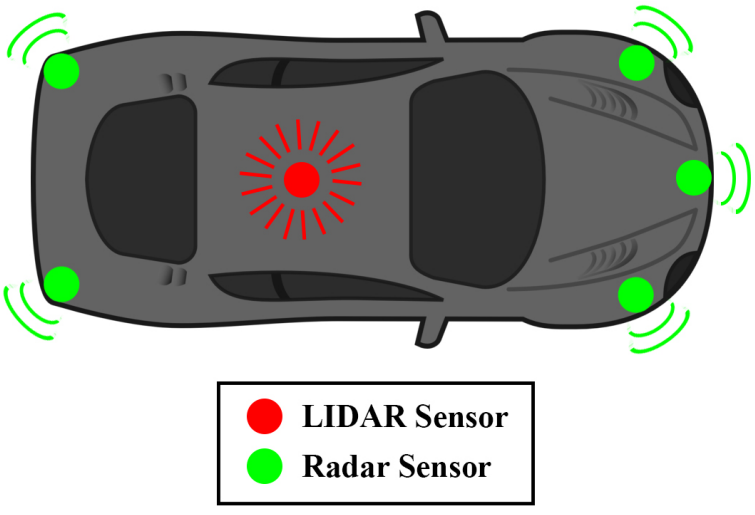


Figure 5.1: *An illustration of the car used to gather data and its sensor setup.*

The lidar sensor, unlike radar sensors, rotates and provides detections of the environment in three dimensions. To make lidar data comparable to radar data from a given time frame it needs to be filtered and transformed. First the most recent full lidar scan is filtered by height to only contain detections from sides of cars. Then the remaining detections are projected on the two dimensional plane. An example of transformed detections can be seen in figure 5.2, which also demonstrates the difference in the level of details between radar and lidar. In the figure it can also be seen that there exist some lidar detections on the hood of cars, therefore the free space algorithm described in algorithm 2 is necessary for lidar data.

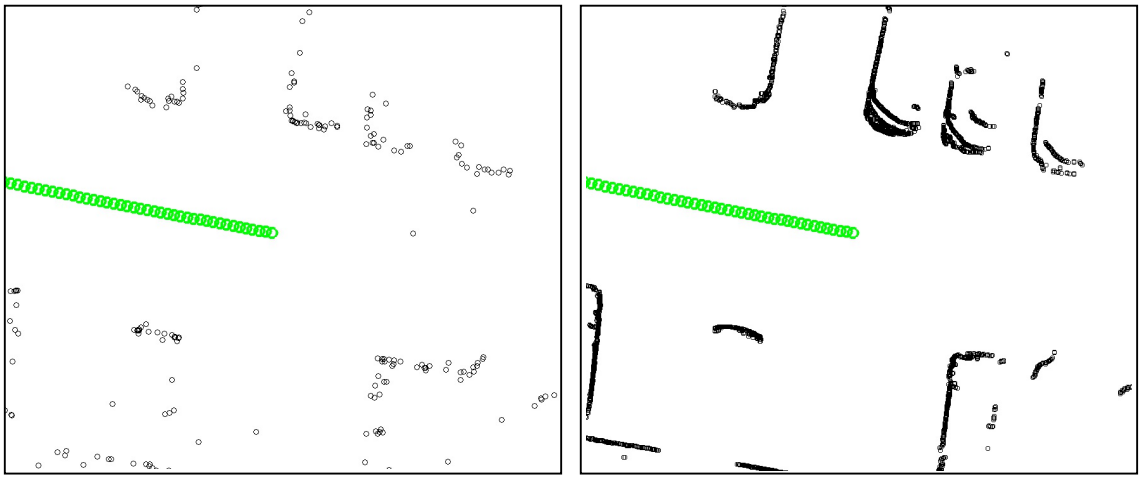


Figure 5.2: *Radar (left) and lidar (right) observations gathered during a time interval of 250 milliseconds. The green dots indicate the vehicle trajectory.*

5.2 Test Scenarios

Data for testing was gathered from three partially filled parking blocks that matched the delimitations stated in section 1.4. All blocks had nearly flat ground, contained only stationary cars and were free from other objects e.g. rails. The layout of the blocks can be seen in figure 5.3, 5.4 and 5.5, where the first is larger and more sparse than the other two. Data was gathered during one lap around each block.

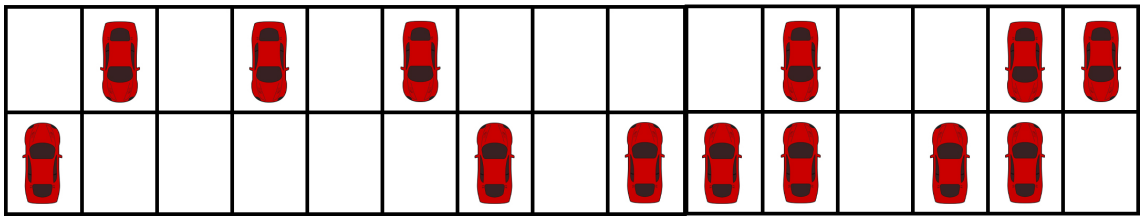


Figure 5.3: *Parking block at Liseberg's parking area and its configuration of cars*

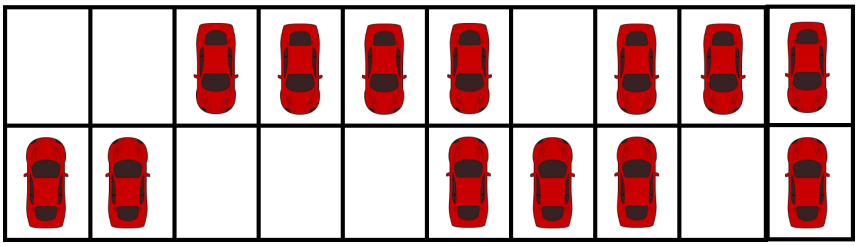


Figure 5.4: *Parking block at the office parking area and its configuration of cars*

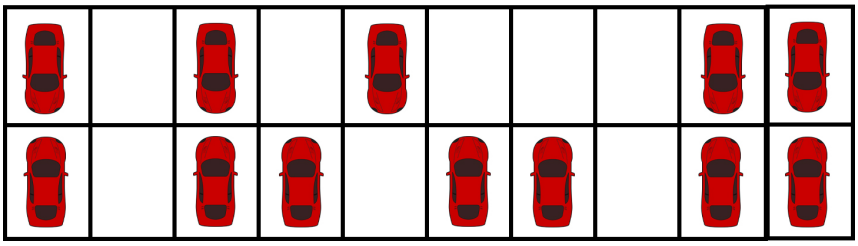


Figure 5.5: *Additional parking block at the office parking area and its configuration of cars*

5.3 Landmark Extraction Evaluation

To evaluate performance of the landmark extraction algorithm, lines extracted from radar data are compared to lines extracted from lidar data. The comparison is made for lines extracted at each time frame in all three previously mentioned parking blocks. The same algorithm is used to extract lidar lines, but parameters are tweaked to increase performance.

For each time frame radar and lidar lines are associated using the method described in 4.1.5. This produces four quantities

- N_{LR} : the number of lidar lines that have a matching radar line
- N_{RL} : the number of radar lines that have a matching lidar line
- N_L : total number of lidar lines
- N_R : total number of radar lines

which in turn are used to compute three statistical performance measures similar to the ones used by Nguyen et al. [7] in their line extraction comparison.

$$\text{True positive rate} = \frac{N_{LR}}{N_L} \tag{5.1}$$

$$\text{Precision} = \frac{N_{RL}}{N_R} \tag{5.2}$$

$$\text{False discovery rate} = 1 - \text{Precision} \tag{5.3}$$

In addition to these measures it is also interesting to investigate how similar radar lines are lidar lines. The following similarity measures are used

- Angle: angle between direction vectors of two lines
- Midpoint: orthogonal distance from the midpoint of the radar line to the lidar line
- Overlap: portion of the radar line that overlaps the lidar line (if the radar line is a segment of the lidar line it is 100% overlap).
- Length: difference in length

Figure 5.6 shows an example of line extraction performed using radar and lidar data. The evaluation method associates the horizontal and vertical lines with each other. The quantities N_{LR}, N_{RL}, N_L and N_R are increased by two, once for each line that has a match. Furthermore, the similarity measures listed above are computed for both pairs.

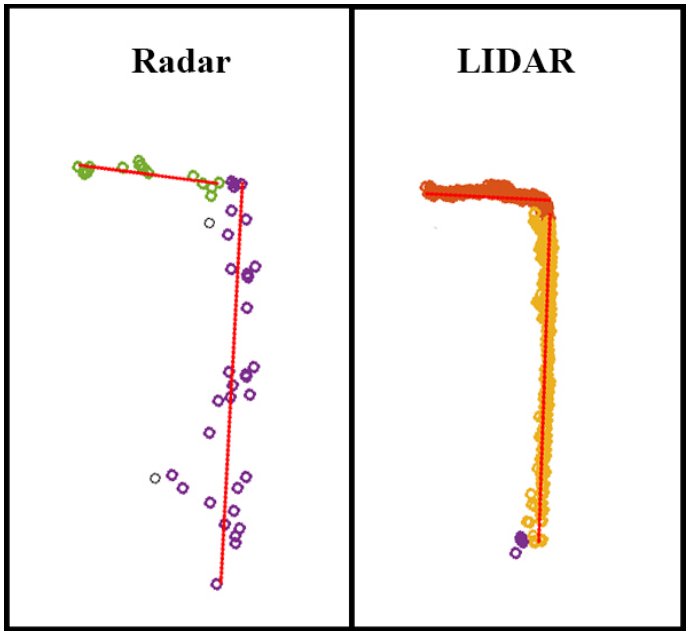


Figure 5.6: *A comparison of line extraction performed using radar and lidar data of the same car*

5.4 Map Evaluation

The map building algorithms described in 4.1 and 4.2 are evaluated similarly to the line extraction, except that only lines in the final maps are considered. First radar and lidar maps are built using the same algorithm. They are then compared line by line with the method described in 5.3. In addition to statistics, a picture with maps on top of each other is provided. A picture makes it possible to observe which lines are good, bad or missing. It is also easier to evaluate which available parking spaces that can be detected.

6
Results

This chapter presents results of developed algorithms using the methods described in section 5. It is divided into landmark extraction and map results. Both sections contain results from different variations of the algorithms to deduce how parameters affect the result.

6.1 Landmark Extraction Accuracy

Landmark extraction using radar data is compared in each time frame against lidar data. One time frame is shown in figure 6.1, where lines have been extracted within the radius (red circle) of the current vehicle position (rightmost green dot). All time frame comparisons from three test scenarios in section 5.2 are combined into a final result presented in one table and four graphs. Results from four different variations of the algorithm are presented.

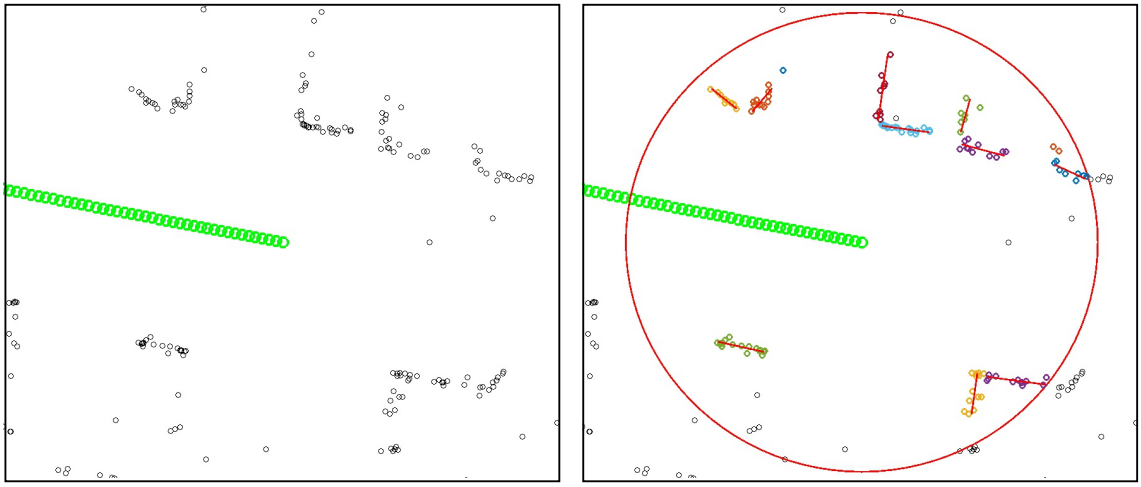


Figure 6.1: *An example of the line extraction algorithm. The left shows radar observations gathered during a time interval of 250 milliseconds. The right shows result of the line extraction algorithm applied on radar observations.*

The first test is to compare constant against dynamic parameters for the single-linkage clustering and line segmentation methods described in 3.1 and 3.2. The comparison is presented in table 6.1 and figure 6.2. In the table it can be seen that both versions extract a similar amount of lines, while the dynamic version achieves

higher true positive rate and slightly lower precision. The figure shows that the dynamic version produces more lines with low angular error, which is also reflected in the table by a lower mean angular error. However, the constant version achieves better overlap and length precision as can be seen in the figure where it has more lines with high overlap and small length error. The midpoint precision is similar for both versions with a small edge for the constant version.

Table 6.1: *A comparison of the line extraction result with constant respectively dynamic parameters.*

	Constant parameters	Dynamic parameters
Nbr. radar lines	27323	27203
True positive rate (%)	54.3	57.9
Precision (%)	72.6	71.6
False discovery rate (%)	27.4	28.4
Mean angular error (deg)	11.0	10.1
Mean midpoint error (m)	0.20	0.21
Mean overlap rate (%)	78.8	73.4
Mean length error (m)	1.31	1.46

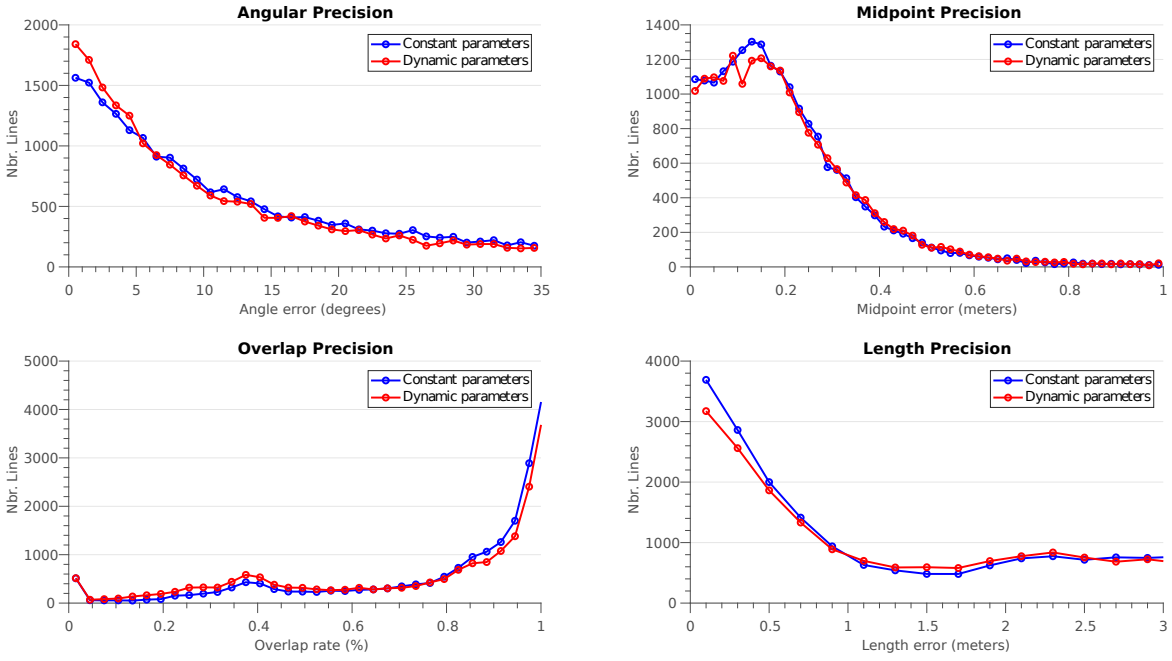


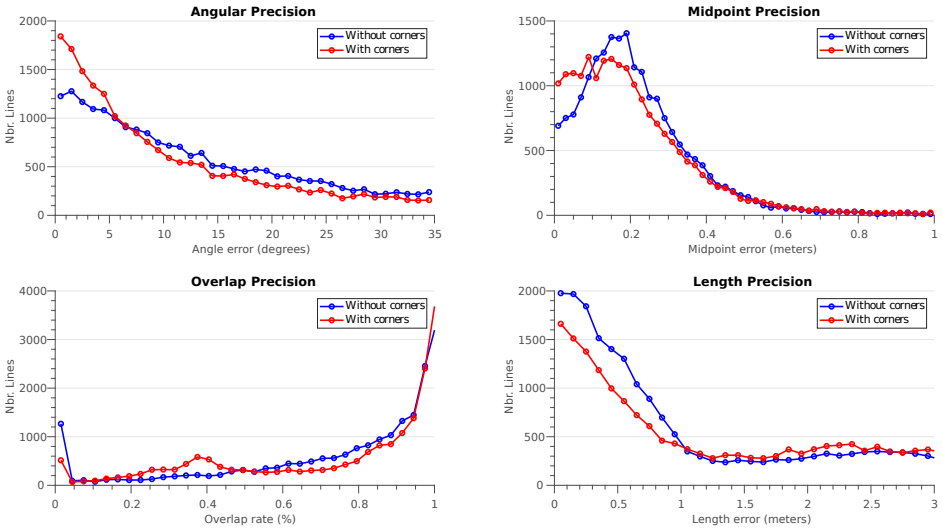
Figure 6.2: *A comparison of the line extraction result with constant respectively dynamic parameters.*

The second test is to compare the algorithm with and without the extraction of orthogonal lines (corners) described in section 3.3. Results are shown in table 6.2 and figure 6.3. In the table it can be seen that the version with corners extracts

fewer lines, but achieves higher true positive rate and precision. In addition it also has a better angular precision, as can be seen in the figure where it has more lines with angular errors below 5 degrees. However, the overlap and midpoint precision is similar according to mean values in the table, though the figure shows that the corner version has more lines with midpoint error below 0.1 meters. In contrast to the angular precision the version without corners achieves better length precision, as can clearly be observed in the figure where the blue graph is above the red for small length errors.

Table 6.2: *A comparison of line extraction results with and without the orthogonal line (corners) assumption.*

	Only lines	With corners
Nbr. radar lines	30390	27203
True positive rate (%)	52.8	57.9
Precision (%)	66.3	71.6
False discovery rate (%)	33.7	28.4
Mean angular error (deg)	12.0	10.1
Mean midpoint error (m)	0.22	0.21
Mean overlap rate (%)	74.0	73.4
Mean length error (m)	1.03	1.46



,

Figure 6.3: *A comparison of line extraction results with and without the orthogonal line (corners) assumption.*

The third test is to compare the two preprocessing methods (graph diameter and free space filtering) described in section 3.4.1 and 3.4.2. Results are shown in table 6.3 and figure 6.4. In the table it can be seen that the diameter version extracts almost twice as many lines, has a higher true positive rate, but worse precision. The figure shows two similarly shaped angular precision curves, while the tables indicates

a lower mean error for the free space version. A similar observation can be made for overlap precision, where the graphs are similar but mean overlap rate is higher for the free space version. The free space version also achieves better midpoint precision. The table indicates a lower mean error and the figure shows that the two graphs show similar trends, but the peak of the blue graph has an offset towards smaller error. However, the diameter version has much better length precision as indicated in the figure where the red curve is clearly above the blue curve for errors below 0.5 meters.

Table 6.3: *A comparison of line extraction result with the free space filter versus the graph diameter as preprocessing method.*

	With diameter	With free space
Nbr. radar lines	45641	27203
True positive rate (%)	69.5	57.9
Precision (%)	53.8	71.6
False discovery rate (%)	46.2	28.4
Mean angular error (deg)	11.0	10.1
Mean midpoint error (m)	0.25	0.21
Mean overlap rate (%)	69.2	73.4
Mean length error (m)	1.40	1.46

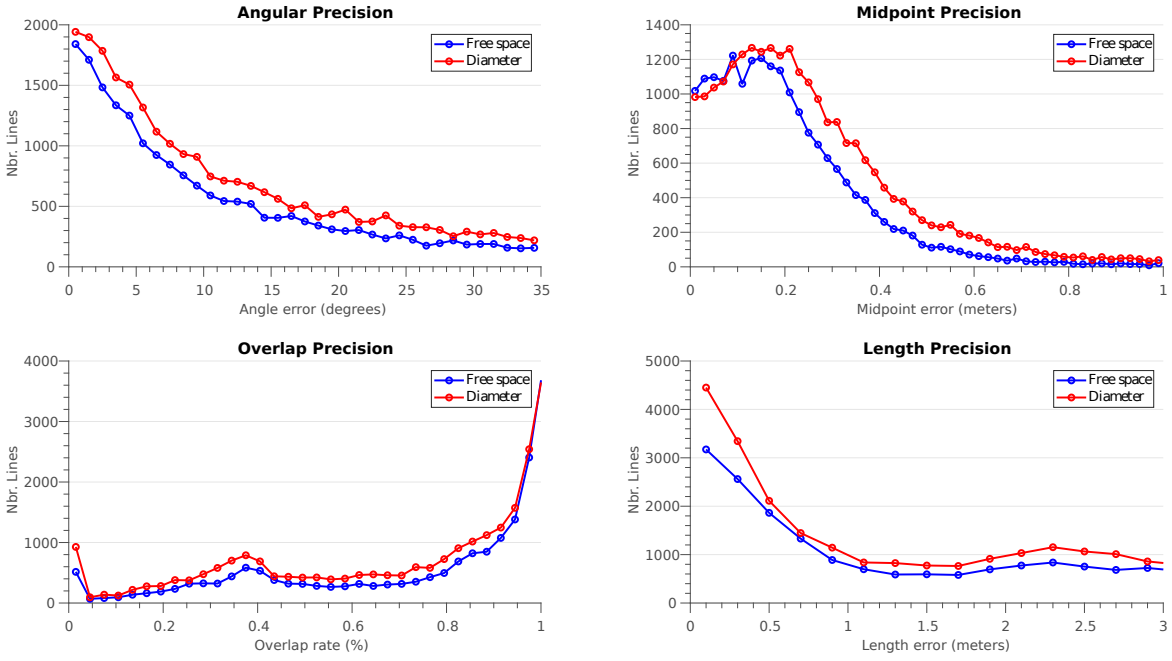


Figure 6.4: *A comparison of line extraction result with the free space filter versus the graph diameter as preprocessing method.*

The final landmark extraction test is to compare how different time intervals for gathering observations affects the result. The time intervals are measured in frames,

where one frame is approximately 50 milliseconds. Results for 6, 8 and 10 frames are shown in table 6.4 and figure 6.5. In the table it can be seen that the number of extracted lines and the true positive rate increases with the number of frames, while the precision decreases. The increase and decrease seems to be larger between 6 and 8 frames compared to between 8 and 10 frames. In the figure it can be observed that the angular precision curves are very similar with almost no deviation from each other. Similarly the midpoint curves are almost identical, except for a higher peak at 0.15 meters for 8 and 10 frames which gives them slightly better midpoint precision. The overlap and length precision increases with the number of frames, as can be seen in the graphs where data points for overlap greater than 0.8 and length error below 0.5 are higher for more frames.

Table 6.4: *A comparison of line extraction result when detections are gathered from different number of frames.*

	6 frames	8 frames	10 frames
Nbr. radar lines	27203	28761	29418
True positive rate (%)	57.9	58.7	59.0
Precision (%)	71.6	70.5	70.3
False discovery rate (%)	28.4	29.5	29.7
Mean angular error (deg)	10.1	10.4	10.4
Mean midpoint error (m)	0.21	0.20	0.20
Mean overlap rate (%)	73.4	75.4	76.7
Mean length error (m)	1.46	1.39	1.36

6.2 Map Accuracy

Map accuracy is evaluated by comparing a map built from radar detections with the same map built from lidar detections line by line. Comparisons from all three test scenarios are combined into one table and four graphs. This section is divided into non-correlated and correlated feature representation with map results from the algorithm described in 4.1 and 4.2.

6.2.1 Non-correlated Feature Representation

The algorithm using a non-correlated feature representation in section 4.1 is tested with different preprocessing methods and different time intervals for collecting observations.

Results from the comparison of preprocessing methods (section 3.4.1 graph diameter and section 3.4.2 free space filtering) are shown in table 6.5 and figure 6.6. In the table it can be seen that the map from the free space version contains 26 lines more than the lidar map, while the diameter version contains 156 more. However, the diameter version achieves better true positive rate but worse precision. The

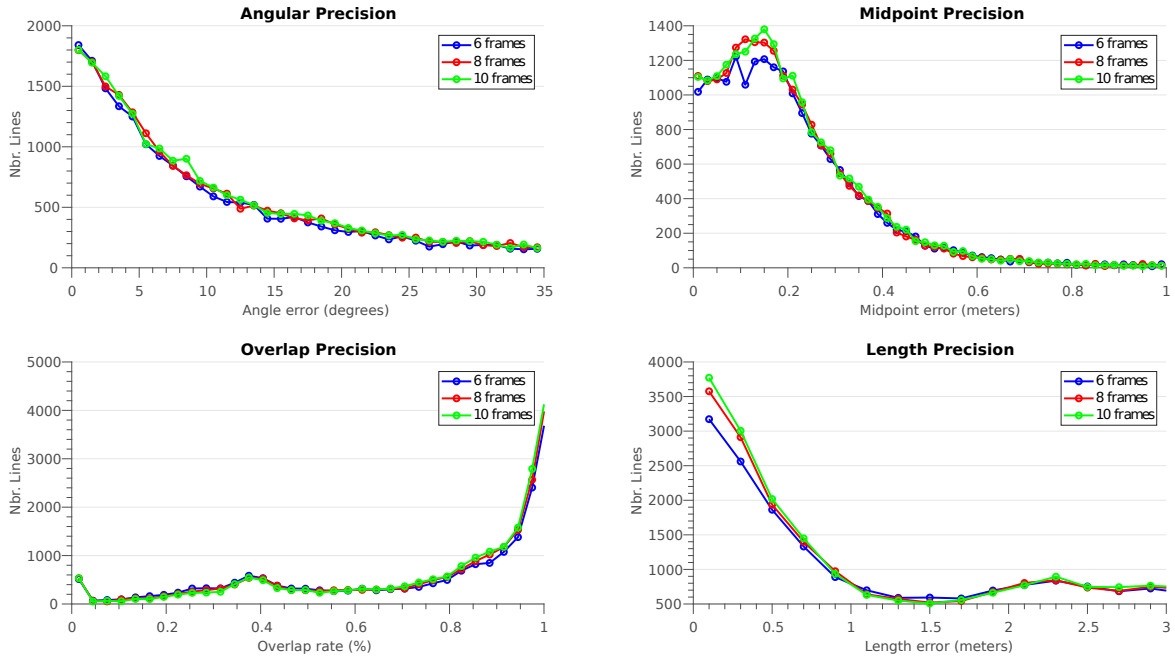


Figure 6.5: *A comparison of line extraction result when detections are gathered from different number of frames.*

table also shows that the free space version achieves better mean values on all four precision metrics. This agrees with the figure where it can be seen for angle error 20-30 degrees, midpoint error 0.4-0.8 and overlap below 0.5 that there are significantly more lines from the diameter version. Although, the free space version extracts less lines it can be seen in the table that it has equally many high overlap and low midpoint error lines as the diameter version.

To see how the map results differentiates from the line extraction results, a comparison between table 6.5 and table 6.3 can be made. Both versions has better true positive rate in the map evaluation, while only the diameter version increases its precision. Only the free space version increases its angular, midpoint and overlap precision in the map evaluation, while the diameter version has similar values on both. However, both versions has worse length precision.

Results from the comparison with different time intervals for collecting observations are shown in table 6.6 and figure 6.7. In the table it can be observed that the 8 frame version has fewest lines and highest true positive rate, while the precision is similar for all. It can also be seen that 8 and 10 frames achieves slightly better mean angle. In addition the angular precision graph shows that 8 frames has most lines with smallest error, and 6 frames has more lines between 15 and 20 degrees error. In contrast to the angle precision the mean midpoint error is slightly better for 6 and 8 frames, while 10 frames has more lines with minimum error. In the table it can be observed that the overlap precision seems to decrease when the number of frames increase, and the overlap precision graph shows more lines with high overlap for the version with 6 frames. However, the mean length error is similar for 6 and

Table 6.5: *A comparison of lines in the final maps built using the graph diameter versus the free space filter as preprocessing method*

	With diameter	With free space
Nbr. radar lines	350	220
Nbr. lidar lines	194	194
True positive rate (%)	86.1	78.4
Precision (%)	61.7	70.9
False discovery rate (%)	38.3	29.1
Mean angular error (deg)	11.0	9.6
Mean midpoint error (m)	0.25	0.19
Mean overlap rate (%)	68.1	79.6
Mean length error (m)	1.79	1.64

10 frames but slightly worse for 8 frames, while the length precision graph shows similar trends for all versions.

The difference between map results and landmark extraction results can be observed by comparing table 6.6 with table 6.4. The map results show better true positive rate and similar precision. The mean values indicate slightly better angle for 8 and 10 frames in the map, as well as better overlap. However, the midpoint precision was similar but the length precision was worse in the map results.

Table 6.6: *A comparison of lines in the final maps built using line extraction with different number of radar frames.*

	6 frames	8 frames	10 frames
Nbr. radar lines	227	220	229
Nbr. lidar lines	194	194	194
True positive rate (%)	76.3	78.4	76.3
Precision (%)	70.9	70.9	71.1
False discovery rate (%)	29.1	29.1	28.9
Mean angular error (deg)	10.4	9.6	9.6
Mean midpoint error (m)	0.19	0.19	0.20
Mean overlap rate (%)	81.3	79.6	78.0
Mean length error (m)	1.59	1.64	1.60

In addition to the statistical results previously presented, a plot of maps built of one parking area is visualized in figure 6.8. Plots of two other parking areas are provided in Appendix A.2 and A.3. The radar maps were built using orthogonal line fitting, free space filtering and time intervals of 8 frames. In figure 6.8 it can be seen that almost all radar lines were near a car and did not occlude any free space. Another observation is that there were few radar lines in between closely parked cars, while the lidar map often had lines there. Another difference between radar and lidar can be seen in the right most part of figure 6.8 where there were long radar lines over several closely parked car, while the lidar maps had one line per car front. In

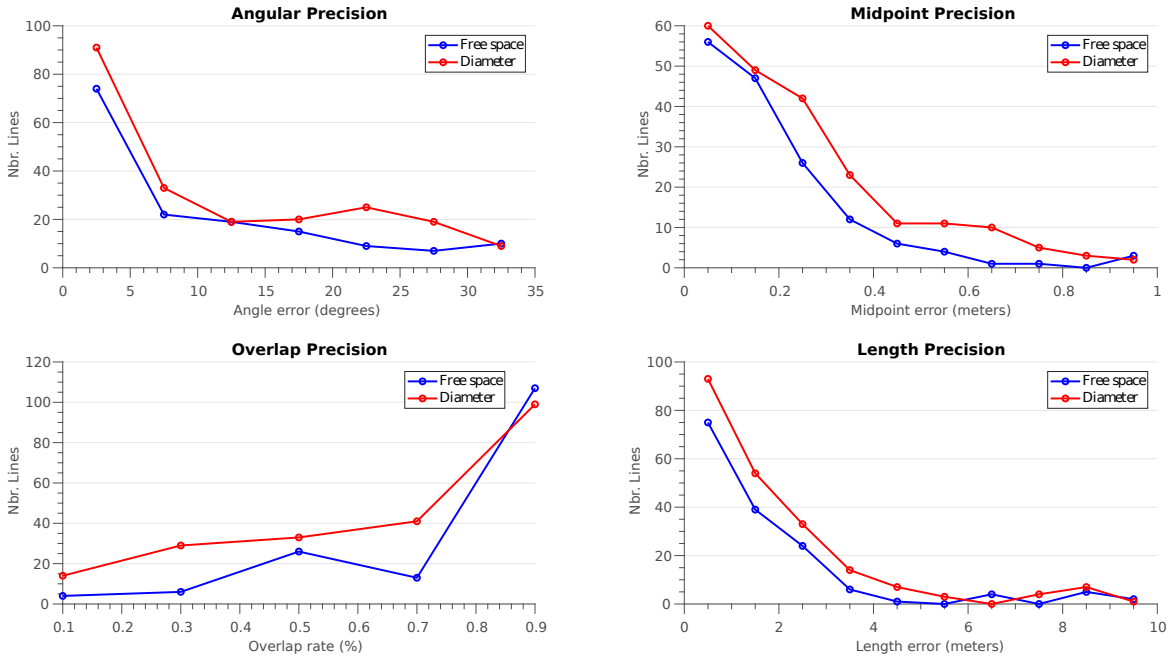


Figure 6.6: *A comparison of lines in the final maps built using the graph diameter versus the free space filter as preprocessing method*

contrast to the too long lines, lines representing car sides on each side of one empty parking space are often too short, as can be seen in the middle of figure 6.8.

6.2.2 Correlated Feature Representation

Applying the SPmodel introduced in section 2.2.2.2 make it possible to form landmarks of multiple features such as lines, corners and endpoints. Whenever one feature in a landmark is reobserved, all other features is updated to maintain correlations. The left and right figures in 6.9 show maps built using the SPmodel with radar and lidar sensors, respectively. The Kalman measurement update, explained in section 4.2.3.1, is not included in this result. The measurement integration instead places the estimated position between the old position and the reobservation at every measurement update and an extra constraint is added to the data association for angle difference. With a strict angular data association it is possible to observe which landmarks that are observed most often since uncertain landmarks with large angular differences does not get associated.

The radar map built from using the SPmodel can be compared to the map built from using the standard model in section 4.1 in figure 6.10. Regarding lines as individual landmarks makes it harder to filter out noisy lines, since the difference in number of reobservations is smaller. Therefore, there are multiple lines representing the same vehicle side in the map.

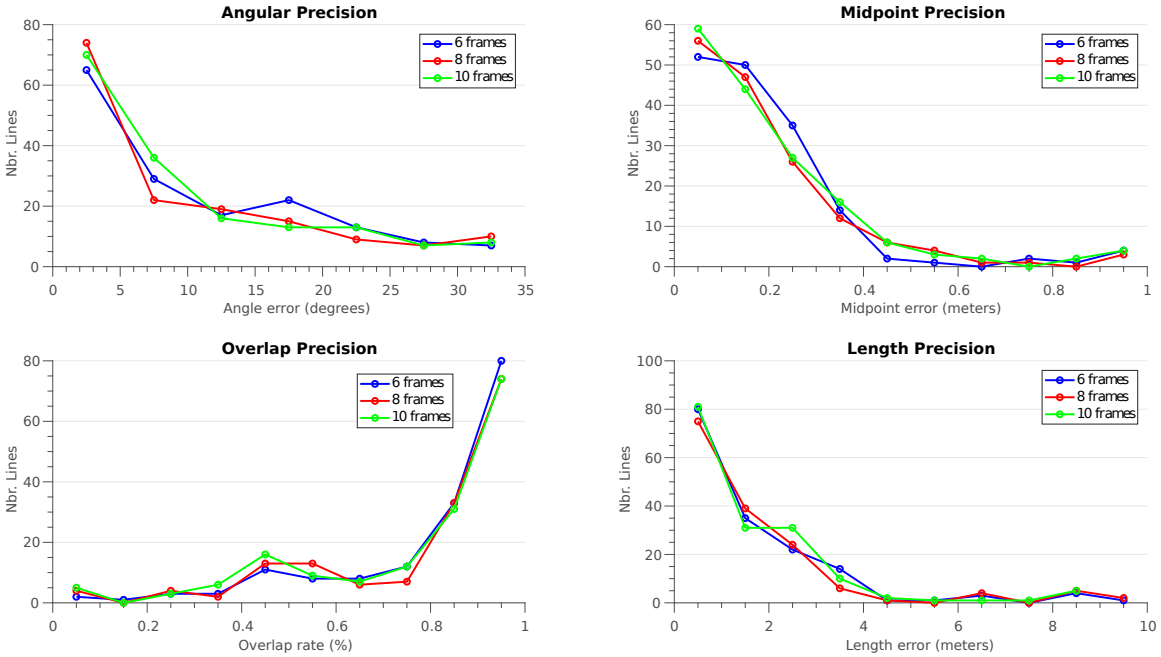


Figure 6.7: *A comparison of lines in the final maps built using line extraction with different number of radar frames.*

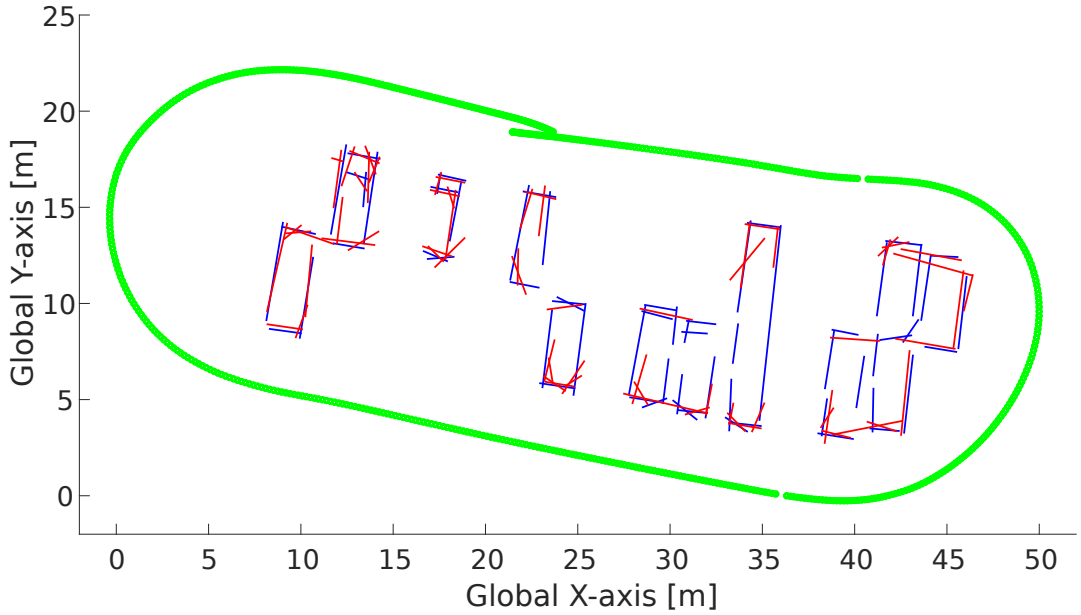


Figure 6.8: *A lidar map (blue) and a radar map (red) of the parking block at Liseberg. The vehicle trajectory is represented by the green line.*

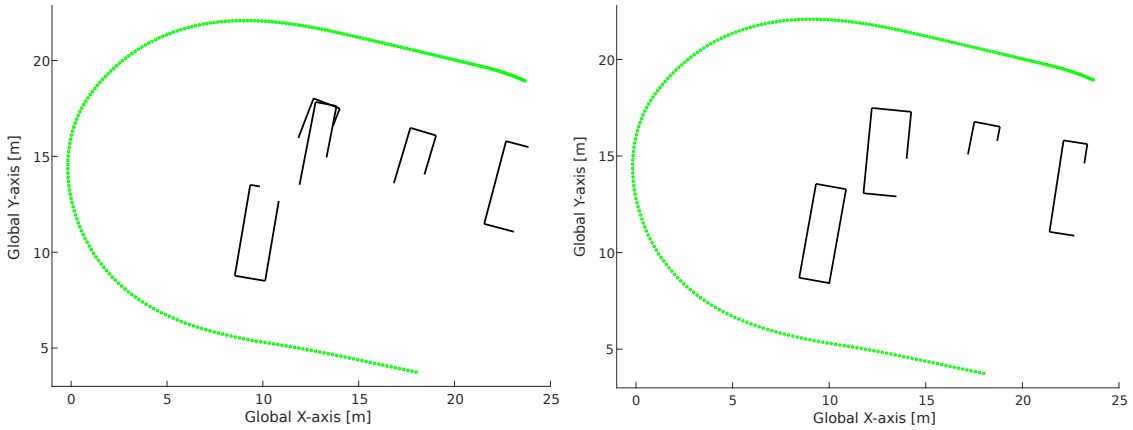


Figure 6.9: *Maps built using the SPmodel and different sensor types; radar (left) and lidar (right). The green lines show how the vehicle moved during map building.*

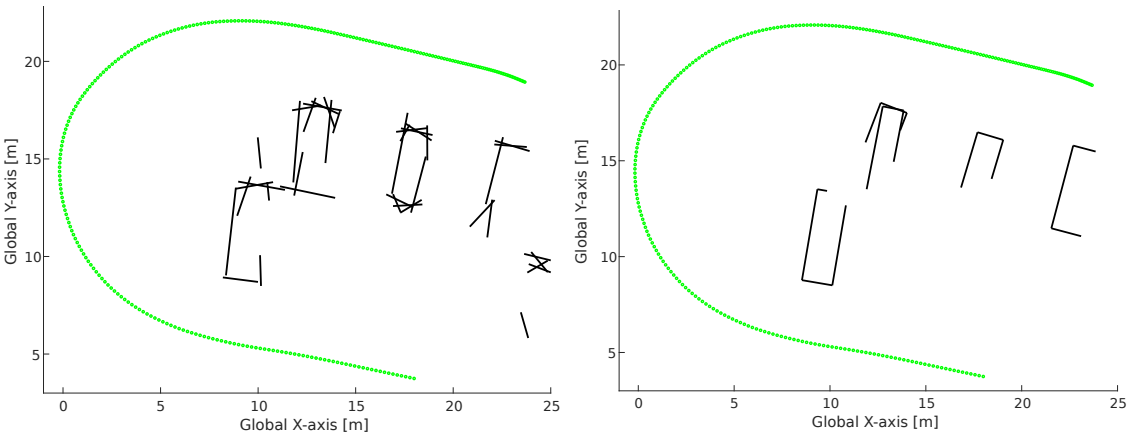


Figure 6.10: *Maps visualizing the difference in result when taking correlations into consideration (right) and not (left).*

7
Discussion

The aim of this project is to build maps of parking areas using radar sensor data and a feature-based SLAM approach. A map is considered accurate enough if it is possible to find empty parking spaces within it. To evaluate the developed algorithms lidar data is used as reference.

Our approach consists of a landmark extraction algorithm and two different map building algorithms. The landmark extraction algorithm finds lines and corners within radar data using two clustering methods together with orthogonal least square line fitting. In addition two different preprocessings of detections using graph diameter and free space filtering are proposed to extract better lines. The first map building method treats each line individually and updates the map using a standard EKF approach. The second approach makes use of the fact that lines of the same car are correlated and uses more advanced landmarks.

In the following sections key results of the algorithms are presented and discussed together with suggestions for future work.

7.1 Landmark Extraction

Using a landmark-based approach for SLAM in the parking are scenario turned out to be a challenge when using radar data. Noisy detections made it difficult to extract lines using classic line fitting techniques. For example, a popular technique for detecting lines when using a lidar sensor is the Incremental Line Fitting Algorithm. It works well with a lidar in indoor environments, where clear lines and corners of walls are present. Cars, however, do not have clear corners since the fronts are often round and rarely rectangular. Trying to fit lines to a point cluster forming half a circle results in high uncertainty in line angles and lengths.

The assumption that cars are rectangular works well when viewing a car slightly from the side. Having detections from the side and front of the car helps stabilize the corner position, which can explain the result in table 6.2 where the corner version performs better than the only lines version. In the parking area scenario, however, sides of cars are mainly detectable when there are in fact available parking spaces. Since the objective for this parking area mapping is to build a map suitable for detecting available parking spaces, this assumption was plausible.

The issue of having multiple noisy line clusters in one dataset made it clear that clustering techniques were necessary. Otherwise the best line fit could span across

two cars, resulting in false landmarks. The first clustering method, a Single-linkage clustering, is intended to cluster between cars. This works well when the distance between cars is large, i.e. when there are available parking spaces between them. The clustering does not work well for cars that are parked too closely to each other. This will result in one point cluster for multiple cars. This can be an explanation to the existence of lines with large errors in the graphs in section 6.1.

The second clustering method is necessary to fit orthogonal lines and form corners. The idea of this algorithm is to build line clusters and start a new cluster whenever the next point is considered to deviate from the line. Dealing with circular car fronts, this algorithm sometimes cuts the cluster in the middle of the front, resulting in a corner that is pointing straight ahead. In this situation the extracted lines have an angular error of 45 degrees. These corners will not be associated to any lidar line due to the large difference in angle, which can be a contribution to the high false discovery rates seen in section 6.1.

To reduce landmark extraction noise, graph diameter and free space filtering were introduced as preprocessing methods. As can be seen in table 6.3 free space filtering mostly gives better result. A reason might be that the diameter path not always follows the sides of a car, depending on the amount of noisy detections within the car. However the graph diameter filtering gave better true positive rate, which indicates that the free space filtering sometimes removes too much information.

7.2 Simultaneous Localization and Mapping

The map building process becomes difficult when landmarks have a high uncertainty. A strict data association adds many false landmarks to the state model, which quickly grows large. This increases the time complexity for data association, which is quadratic in the number of landmarks. On the other hand, a generous data association will associate good lines with bad lines, resulting in false locations for landmarks in the map.

The idea of using the SPmodel was to maintain correlations between lines and also more easily filter out false landmarks. By connecting lines and building large landmarks, the number of reobservations becomes the sum of reobservations of all lines that are included. The measurement integration during building of the maps in figure 6.9 places the estimated position between the old position and the reobservation at every measurement update. Using this measurement integration, along with a strict data association, shows which landmarks are seen most often. Therefore, the result in 6.9 shows potential in using the SPmodel for mapping parking areas since it is possible to form vehicle-shaped landmarks from landmarks extracted using both radar and lidar data. Integrating the EKF measurement integration as suggested in section 4.2.3.1 could produce better estimates in feature locations. Because of the large uncertainty when using the line-based representation, presented in section 4.1, vehicle localization becomes difficult. Using other features such as corners could however allow a better correction in vehicle position.

One drawback in using the SPmodel arises due to occlusions. Occlusions often

arise in parking area scenarios since vehicles shadow other vehicles. Consider the situation when a vehicle is observed, then shadowed from view by other cars, and then observed again at later time steps. This will result in two formed landmarks for the same vehicle, resulting in fewer reobservations per landmark. Therefore small parts of vehicles risk being filtered away as noise if the number of reobservations are low. One example of this is visible in figure 6.9. The back end of the middle car of the top row has been filtered away in both maps. The back is visible through the available parking spaces, but was not associated to the landmark representing the front of the same vehicle. This drawback is not present when using the line-base representation in section 4.1. This is clear when studying the maps in figure 6.10 which show the same parking area when using line-based representation and the SPmodel. The line-based map cannot form rectangular shapes of vehicles, but all vehicle sides are visible.

Due to the rectangular car assumption, false landmarks are reobserved as often as true lines. Therefore, the line-based representation fails to filter out false landmarks. The SPmodel, however, succeeds in building rectangular-shaped landmarks which increases the number of reobservations of true landmarks. False landmarks can therefore be filtered out. It is however a risk that it also filters out true vehicle sides if occlusions are not handled.

7.3 Future Work

The results in chapter 6 show the potential in using a landmark-based approach to SLAM when finding available parking spaces using radar data. There are several possibilities to improve results by increasing stability, time efficiency and reduce uncertainty.

7.3.1 Landmark Extraction

Landmark extraction is a fundamental part in landmark-based approaches to SLAM. Therefore, improvements of landmark extraction algorithms have a direct affect on map quality. Similarly, data used as input to landmark extraction is fundamental for landmark quality, hence an important stage in the map building process is data preprocessing to reduce noise. Two methods are introduced in this thesis: graph diameter filtering for removing outliers, and freespace filtering for removing false detections inside vehicles. Other methods that could be applied to reduce false detections are to utilize the characteristics of radar sensors. For example, radar sensors additionally return range rate of detections, making it possible to detect and filter dynamic objects such as detections of pedestrians and moving vehicles.

The algorithms for extracting landmarks that are presented in chapter 3 could also be improved to increase map quality. For example, the assumption that each resulting cluster from the single-linkage clustering in section 3.1 forms parts of one vehicle is not true when vehicles are parked too closely. This problem is visible in figure 3.1 in the bottom-left image where two vehicles are connected and forms one cluster. This

problem could be resolved by adding an extra clustering algorithm that utilizes the graph diameter visualized in figure 3.3 to detect and cut at the intersection in the MST graph. Another suggestion for improving landmark extraction is to consider different types of landmarks than rectangles for vehicle shapes. Since it is common with oval car fronts, a spline-based landmark model could obtain more accurate vehicle shapes.

Improvements could also be added after landmark extraction. For example, a measure of what are good and bad lines could be beneficial for filtering bad lines, which could increase efficiency of data association.

7.3.2 Simultaneous Localization and Mapping

Pursuing the suggestions in the previous section could increase map quality. Improvements could however also be added to the map building processes in chapter 4. The positions of landmarks are currently estimated based on uncertainty in sensor and odometry. The estimation could also consider uncertainty due to landmark extraction. Flaws in landmark extraction could be detected and result in a high uncertainty for bad landmarks. Landmarks far away should also have a higher uncertainty than landmarks close to the vehicle.

Another problem discussed in section 7.2 is the effect of occlusions when applying the SPmodel. Taking occlusions into consideration to successfully associate vehicle parts to form full rectangles could make the maps visualized in figure 6.9 complete.

Furthermore, the implementing the EKF measurement integration in section 4.2.3.1 would provide estimates of feature locations in the SPmodel. This would simplify data association and allow vehicle localization based on corner extraction.

7.4 Ethical and Social Aspects

Allowing cars to park by themselves has many advantages. For example, people can save time and energy to pursue other matters, and the size of parking areas can be reduced to create space. However, there are ethical aspects to consider when leaping into the idea of self-driving cars.

For example, when including robots in every day tasks it is important to reflect on how the human factor is removed from the situation. Consider a scenario when there is one available space in a parking area and multiple vehicles wanting to park. One of the vehicles contains a person who needs the parking space more than the others, for example a pregnant woman or a disabled person. Removing the human factor from the decision would rob that person of getting the last parking space. The choice of the robot would cause damage to a situation typically solved through communication between drivers.

Another aspect to consider is the fact that allowing robots to perform tasks, normally performed by humans, will result in fewer job opportunities. In some countries it

is common to provide valet parking at hotels. Automating valet parking would therefore reduce the number of job possibilities in those counties.

Furthermore, accidents in parking areas can occur. If there is a crash between a self-driving car and a car operated by a human, the conflict could be hard to resolve. Therefore there are additional juridical aspects that need to be considered.

8

Conclusion

The objective of this master thesis was to investigate the possibility of locating available parking spaces using radar data and a landmark-based approach to SLAM. Because of the noisiness of radar data, the main challenge was to extract landmarks accurate enough to build a map.

Despite high uncertainty in landmarks due to noise, the resulting maps were accurate enough to detect available spaces. However, the estimated positions of parked vehicles should not be used for precise navigation, such as parking a vehicle. Furthermore, due to noisy data there were many false landmarks in the state model. Since the time complexity of data association is quadratic in the number of landmarks, a landmark-based solution using radar data is unfit for real-time application.

Although, the results from applying the SPmodel showed potential in building maps with a higher precision and less noise. Pursuing the suggestions listed in section 7.3 could result in estimations fit for navigation. It could also allow for a better vehicle correction since the position of a corner is more deterministic than a line segment.

Bibliography

[1] Waymo, *On the Road to Fully Self-Driving*, Waymo, Mountain View, CA, USA, 2017. Accessed on: Feb., 5, 2018. [Online]. Available: https://storage.googleapis.com/sdc-prod/v1/safety-report/waymo-safety-report-2017.pdf

[2] H. Durrant-Whyte and T. Bailey, *Simultaneous localization and mapping: part I*, in IEEE Robotics & Automation Magazine, vol. 13, no. 2, pp. 99-110, June 2006. doi: 10.1109/MRA.2006.1638022

[3] R. Dubé, M. Hahn, M. Schütz, J. Dickmann and D. Gingras, *Detection of parked vehicles from a radar based occupancy grid*, 2014 IEEE Intelligent Vehicles Symposium Proceedings, Dearborn, MI, 2014, pp. 1415-1420. doi: 10.1109/IVS.2014.6856568

[4] F. Schuster, C. G. Keller, M. Rapp, M. Haueis and C. Curio, *Landmark based radar SLAM using graph optimization*, 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, 2016, pp. 2559-2564. doi: 10.1109/ITSC.2016.7795967

[5] A. Garulli, A. Giannitrapani, A. Rossi and A. Vicino, *Mobile robot SLAM for line-based environment representation*, Proceedings of the 44th IEEE Conference on Decision and Control, 2005, pp. 2041-2046. doi: 10.1109/CDC.2005.1582461

[6] J. Lv, Y. Kobayashi, A. A. Ravankar and T. Emaru, *Straight Line Segments Extraction and EKF-SLAM in Indoor Environment*, Journal of Automation and Control Engineering, vol. 2, no. 3, Sept 2014

[7] V. Nguyen, A. Martinelli, N. Tomatis and R. Siegwart, *A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics*, 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005, pp. 1929-1934. doi: 10.1109/IROS.2005.1545234

[8] J. Neira and J. D. Tardos, *Data association in stochastic mapping using the joint compatibility test*, in IEEE Transactions on Robotics and Automation, vol. 17, no. 6, pp. 890-897, Dec 2001. doi: 10.1109/70.976019

[9] R. C. Prim, *Shortest connection networks and some generalizations*, in The Bell System Technical Journal, vol. 36, no. 6, pp. 1389-1401, Nov. 1957. doi: 10.1002/j.1538-7305.1957.tb01515.x

[10] G. Leach, *Improving Worst-Case Optimal Delaunay Triangulation Algorithms*, in 4th Canadian Conference on Computational Geometry, 1992

[11] J. Collier, *SLAM Techniques and Algorithms*, presented at 6th Canadian Conference on computer Robot Vision, 25-27 May, 2009. [Online]. Available: http://computerrobotvision.org/2009/tutorial_day/crv09SLAMTutorial.pdf

[12] X. Zhu, *Clustering*, class notes for CS769: Advanced Natural Language Processing, Department of Computer Science, University of Wisconsin-Madison, Madison, Wisconsin, USA, spring 2010. [Online]. Avaliable: http://pages.cs.wisc.edu/ jerryzhu/cs769/clustering.pdf

[13] P. Arbenz, class notes for *EINFÜHRUNG IN MATLAB*, Computer Science Department, Eidgenössische Technische Hochschule, Zürich, Switzerland, autumn 2008. [Online]. Avaliable: https://www.inf.ethz.ch/personal/arbenz/MatlabKurs/node87.html

[14] J. A. Castellanos, J. D. Tardos, *Mobile Robot Localization and Map Building: A Multisensor Fusion Approach*. Boston: Springer, 1999.

[15] C. Stachniss, *Robot Mapping: Extended Information Filter*, class notes for Autonome Intelligente Systeme, Albert Ludwigs Universität Freiburg, 2012-11-14. Accessed on: June., 4, 2018. [Online]. Available: http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam06-eif.pdf

[16] Trafikkontoret, *Trafik- och resandeutveckling*, Trafikkontoret, Gothenburg, Sweden, 2017. Accessed on: June, 14, 2018. [Online]. Available: http://goteborg.se/wps/wcm/connect/866c6249-4a74-4d92-8d3b-36c7b89ca03f/Trafik+och+resandeutveckling+2016+WEBB.pdf?MOD=AJPERES

A

Appendix 1

A.1 Incremental Update of Covariance, Eigenvectors and Eigenvalues

The covariance, eigenvectors and eigenvalues of point clusters are computed during line segmentation. Incrementally updating these values can reduce the time complexity and is therefore presented in this section.

The covariance of two variablex x and y is defined as

$$\begin{aligned} \mathrm{S}_{xy}^{(n)} &= \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x}^{(n)})(y_i - \bar{y}^{(n)}) \\ &= \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i y_i - \bar{y}^{(n)}\sum_{i=1}^{n} x_i - \bar{x}^{(n)}\sum_{i=1}^{n} y_i + n\bar{x}^{(n)}\bar{y}^{(n)}\right) \\ &= \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i y_i - \bar{y}^{(n)} n\bar{x}^{(n)} - \bar{x}^{(n)} n\bar{y}^{(n)} + n\bar{x}^{(n)}\bar{y}^{(n)}\right) \\ &= \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i y_i - n\bar{y}^{(n)}\bar{x}^{(n)}\right) \end{aligned} \tag{A.1}$$

where

$$\begin{aligned} \bar{x}^{(n)} &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ \bar{y}^{(n)} &= \frac{1}{n}\sum_{i=1}^{n} y_i \end{aligned} \tag{A.2}$$

is the sample mean.

The sample mean can be updated from the sample mean for n points as

$$\begin{aligned}\bar{x}^{(n+1)} &= \frac{1}{n+1}\sum_{i=1}^{n+1} x_i \\ &= \frac{1}{n+1}(x_1 + \ldots + x_{n+1}) \\ &= \frac{1}{n+1}(\frac{n}{n}(x_1 + \ldots + x_n) + x_{n+1}) \\ &= \frac{n}{n+1}\bar{x}^{(n)} + \frac{1}{n+1}x_{n+1}\end{aligned} \tag{A.3}$$

From equations (A.1) and (A.3), the covariance when adding a point $n+1$ is calculated as

$$\mathrm{S}_{xy}^{(n+1)} = \frac{1}{n}\left(\sum_{i=1}^{n} x_i y_i + x_{n+1}y_{n+1} - (n+1)\bar{y}^{(n+1)}\bar{x}^{(n+1)}\right) \tag{A.4}$$

The covariance matrix of a set of points (x, y) is

$$\mathbf{A} = \begin{bmatrix}\mathrm{S}_{xx} & \mathrm{S}_{yx} \\ \mathrm{S}_{xy} & \mathrm{S}_{yy}\end{bmatrix} \tag{A.5}$$

The eigenvalues can be calculated from the covariance matrix by performing an eigendecomposition. The vector $\mathbf{v}$ is an eigenvector of matrix $\mathbf{A}$ if it satisfies

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v} \tag{A.6}$$

where λ are the eigenvalues. The eigenvalues are obtained by solving the equation

$$\begin{aligned}\det(\mathbf{A} - \lambda\mathbf{I}) = 0 \\ \Leftrightarrow \begin{vmatrix}\mathrm{S}_{xx} - \lambda & \mathrm{S}_{yx} \\ \mathrm{S}_{xy} & \mathrm{S}_{yy} - \lambda\end{vmatrix} = 0 \\ \Leftrightarrow (\mathrm{S}_{xx} - \lambda)(\mathrm{S}_{yy} - \lambda) - \mathrm{S}_{yx}\mathrm{S}_{xy} = 0\end{aligned} \tag{A.7}$$

which has the solution

$$\lambda = \frac{1}{2}\left(\mathrm{S}_{xx} + \mathrm{S}_{yy} \pm \sqrt{(\mathrm{S}_{xx} + \mathrm{S}_{yy})^2 - 4(\mathrm{S}_{xx}\mathrm{S}_{xx} - \mathrm{S}_{xy}\mathrm{S}_{yx})}\right) \tag{A.8}$$

The eigenvectors are obtained from solving equation (A.6) which can be written

$$\begin{bmatrix}\mathrm{S}_{xx} & \mathrm{S}_{yx} \\ \mathrm{S}_{xy} & \mathrm{S}_{yy}\end{bmatrix}\begin{bmatrix}v_1 \\ v_2\end{bmatrix} = \begin{bmatrix}\lambda_1 \\ \lambda_2\end{bmatrix}\begin{bmatrix}v_1 \\ v_2\end{bmatrix} \Leftrightarrow \begin{bmatrix}\mathrm{S}_{xx} - \lambda_1 & \mathrm{S}_{yx} \\ \mathrm{S}_{xy} & \mathrm{S}_{yy} - \lambda_2\end{bmatrix}\begin{bmatrix}v_1 \\ v_2\end{bmatrix} = \begin{bmatrix}0 \\ 0\end{bmatrix} \tag{A.9}$$

A.2 Propagation of Uncertainty

Propagation of uncertainty is how the uncertainties of variables propagate to function values during a mapping $\mathbf{f}(\mathbf{x})$.

When the functions $\mathbf{f} = f_1, \ldots, f_m$ are linear combinations of the variables $\mathbf{x} = x_1, \ldots, x_n$, they can be expressed in matrix form as

$$\mathbf{f} = \mathbf{A}\mathbf{x} \tag{A.10}$$

If the covariance matrix of $\mathbf{x}$ is Σ^x, then the covariance of $\mathbf{f}$ is

$$\Sigma^f = \mathbf{A}\Sigma^x\mathbf{A}^\top \tag{A.11}$$

When the functions $\mathbf{f}$ are non-linear combinations of the variables $\mathbf{x}$ the functions must be linearized to a first order Taylor expansion

$$f_k \approx f_k^0 + \sum_i^n \frac{\partial f_k}{\partial x_i} x_i \tag{A.12}$$

Therefore, the mapping can be approximated to a matrix form using the Jacobian matrix $\mathbf{J} = \nabla\mathbf{f}$

$$\mathbf{f} \approx f^0 + \mathbf{J}\mathbf{x} \tag{A.13}$$

Since f^0 is a constant it does not contribute to the error propagation, which is therefore

$$\Sigma^f = \mathbf{J}\Sigma^x\mathbf{J}^\top \tag{A.14}$$

for the non-linear case.

A.3 Transformations of Reference Frames

The transformation of a reference frame can be represented in two ways [14]: location vectors and homogenous matrices. A location vector consist of two Cartesian coordinates and one angle.

A.3.1 Compositions and Inversions

Two transformations, with locaton vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ is called a *composition* and is denoted

$$\mathbf{x}_3 = \mathbf{x}_1 \oplus \mathbf{x}_2. \tag{A.15}$$

while the inverse of one transformation $\mathbf{x}_1$ is denoted

$$\mathbf{x}_{(-1)} = \ominus \mathbf{x}_1. \tag{A.16}$$

A composition and inversion is visualized in figure A.1.

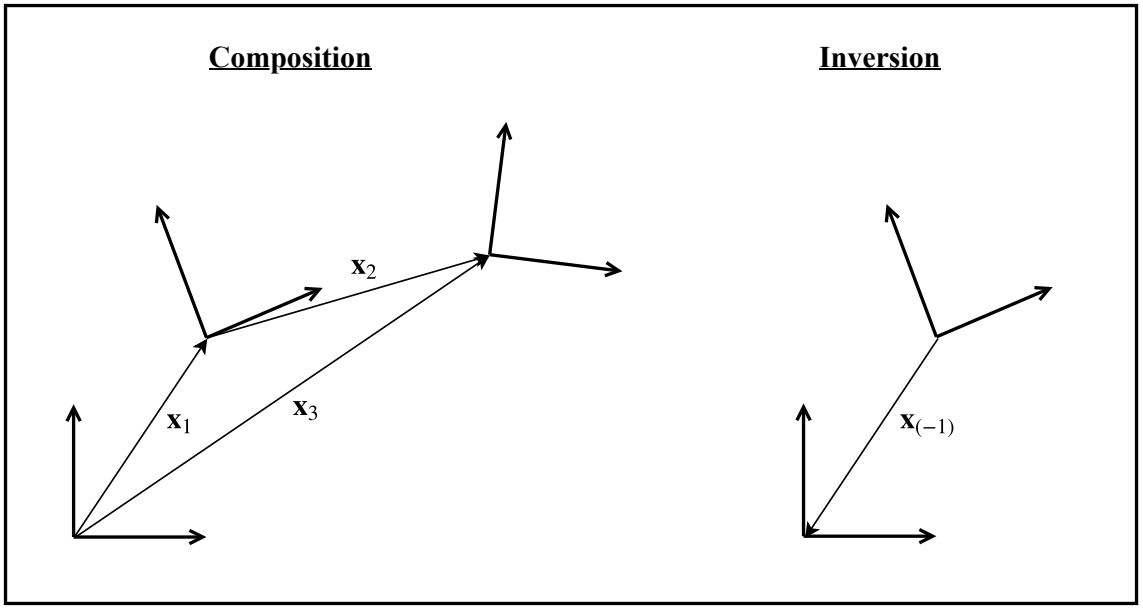


Figure A.1: *Visualization of composition and inversion of reference frame transformations.*

A homogenous matrix is defined as

$$\mathbf{H} = \begin{pmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} n_x & o_x & p_x \\ n_y & o_y & p_y \\ 0 & 0 & 1 \end{pmatrix} \tag{A.17}$$

where $\mathbf{R}$ is a rotation matrix, $\mathbf{p}$ is a translation vector and $\mathbf{n}$, $\mathbf{o}$ are the columns of $\mathbf{R}$. Compositions and inversions, in equations (A.15) and (A.16), can be expressed using Homogenous matrices. The product of two homogenous matrices is the equivalent to a composition

$$\mathbf{H}_3 = \mathbf{H}_1\mathbf{H}_2 = \begin{pmatrix} \mathbf{R}_1\mathbf{R}_2 & \mathbf{p}_1 + \mathbf{R}_1\mathbf{p}_2 \\ \mathbf{0} & 1 \end{pmatrix} \tag{A.18}$$

and the inversion of a homogenous matrix is equivalent to the inversion of a location vector

$$\mathbf{H}^{-1} = \begin{pmatrix} \mathbf{R}^\top & -\mathbf{R}^\top\mathbf{p} \\ \mathbf{0} & 1 \end{pmatrix} \tag{A.19}$$

The location vector of the transformation expressed by a homogenous matrix is obtained through

$$\mathbf{x} = Loc(\mathbf{H}) = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} p_x \\ p_y \\ \text{atan2}(n_y, n_x) \end{pmatrix} \tag{A.20}$$

and the homogenous matrix from a location vector through

$$\mathbf{H} = Hom(\mathbf{x}) = \begin{pmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{pmatrix} \tag{A.21}$$

A.3.2 Jacobians of Reference Frame Transformations

Jacobians of transformations are needed when propagating uncertainty between reference frames, see Appendix A.2. Therefore, Jacobians over different types of transformations are listed in this section.

A.3.3 Jacobian of a Transformation

The relative location of two reference frames A and B is specified by the location vector $\mathbf{x}_{AB}$. A differential change in location vector A, taking it to A' , is specified with the composition

$$\mathbf{x}_{A'} = \mathbf{x}_A \oplus \mathbf{d}_A. \tag{A.22}$$

The differential change of A can be propagated to B according to [**?**]

$$\begin{aligned} \mathbf{d}_A &= \mathbf{J}\{\mathbf{x}_{AB}\}\mathbf{d}_B \\ \mathbf{d}_B &= \mathbf{J}\{\mathbf{x}_{BA}\}\mathbf{d}_A = \mathbf{J}\{\mathbf{x}_{AB}\}^{-1}\mathbf{d}_A \end{aligned} \tag{A.23}$$

where

$$\begin{aligned} \mathbf{J}_{AB} = \mathbf{J}\{\mathbf{x}_{AB}\} &= \begin{pmatrix} \cos\theta & -\sin\theta & y_{AB} \\ \sin\theta & \cos\theta & -x_{AB} \\ 0 & 0 & 1 \end{pmatrix} \\ \mathbf{J}_{BA} = \mathbf{J}\{\mathbf{x}_{BA}\} = \mathbf{J}^{-1}\{\mathbf{x}_{AB}\} &= \begin{pmatrix} \cos\theta & \sin\theta & x_{AB}\sin\theta - y_{AB}\cos\theta \\ -\sin\theta & \cos\theta & x_{AB}\cos\theta + y_{AB}\sin\theta \\ 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{A.24}$$

A.3.4 Jacobians of Compositions

The Jacobians of compositions and inversions, introduced in appendix A.3.1, are provided by [14]. The Jacobian of a composition with respect to the first and second operand, respectively, is

$$\begin{aligned}\mathbf{J}_{1\oplus}\{\mathbf{x}_1,\mathbf{x}_2\} &= \begin{pmatrix}1 & 0 & -x_2\sin\theta_1 - y_2\cos\theta_1\\ 0 & 1 & x_2\cos\theta_1 - y_2\sin\theta_1\\ 0 & 0 & 1\end{pmatrix}\\ \mathbf{J}_{2\oplus}\{\mathbf{x}_1,\mathbf{x}_2\} &= \begin{pmatrix}\cos\theta_1 & -\sin\theta_1 & 0\\ \sin\theta_1 & \cos\theta_1 & 0\\ 0 & 0 & 1\end{pmatrix}\end{aligned} \tag{A.25}$$

where $\mathbf{x}_1 = (x_1, y_1, \theta_1)$ and $\mathbf{x}_2 = (x_2, y_2, \theta_2)$. These Jacobians are simplified when the operand, to which it is being derived, equals zero

$$\begin{aligned}\mathbf{J}_{1\oplus}\{\mathbf{0},\mathbf{x}\} &= \begin{pmatrix}1 & 0 & -y\\ 0 & 1 & x\\ 0 & 0 & 1\end{pmatrix}\\ \mathbf{J}_{2\oplus}\{\mathbf{0},\mathbf{x}\} &= \begin{pmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{pmatrix}\end{aligned} \tag{A.26}$$

A.3.5 Jacobian of Inversion

The Jacobian of an inversion is

$$\mathbf{J}_{\ominus}\{\mathbf{x}\} = \begin{pmatrix}-\cos\theta & -\sin\theta & x\sin\theta - y\cos\theta\\ \sin\theta & -\cos\theta & x\cos\theta + y\sin\theta\\ 0 & 0 & 1\end{pmatrix} \tag{A.27}$$

A.4 Sensor Observation Transformation

Given a detection $[\theta, d]^\top$ with covariance C^{polar} from a sensor with mounting position x_m, y_m and orientation α_m. The detection can be transformed to Cartesian coordinates in a coordinate system centered in the front bumper of the car by the following steps:

Transform to Cartesian coordinates

$$\begin{bmatrix}x^{\text{sensor}}\\ y^{\text{sensor}}\end{bmatrix} = \begin{bmatrix}d\sin(\theta)\\ d\cos(\theta)\end{bmatrix} \tag{A.28}$$

Compute the Jacobian of the transformation and update the covariance according to A.2

$$\mathbf{J} = \begin{bmatrix} d\cos(\theta), & \sin(\theta) \\ -d\sin(\theta), & \cos(\theta) \end{bmatrix} \tag{A.29}$$

$$\mathbf{C}^{\text{cartesian}} = \mathbf{J}\mathbf{C}^{\text{polar}}\mathbf{J}^{\top} \tag{A.30}$$

Change coordinate system using the sensor mounting position.

$$\mathbf{R} = \begin{bmatrix} \cos(-\alpha_m), & -\sin(-\alpha_m) \\ \sin(-\alpha_m), & \cos(-\alpha_m) \end{bmatrix} \tag{A.31}$$

$$\mathbf{t} = \begin{bmatrix} x_m \\ y_m \end{bmatrix} \tag{A.32}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{R} \begin{bmatrix} x^{\text{sensor}} \\ y^{\text{sensor}} \end{bmatrix} + \mathbf{t} \tag{A.33}$$

$$\mathbf{C} = \mathbf{R}\mathbf{C}^{\text{cartesian}}\mathbf{R}^{\top} \tag{A.34}$$

The transformed detection is represented by $[x, y]^{\top}$ with covariance $\mathbf{C}$.

A.5 Additional Map Results

The two additional radar, lidar overlay maps of the two office parking blocks are presented in figure A.2 and A.3.

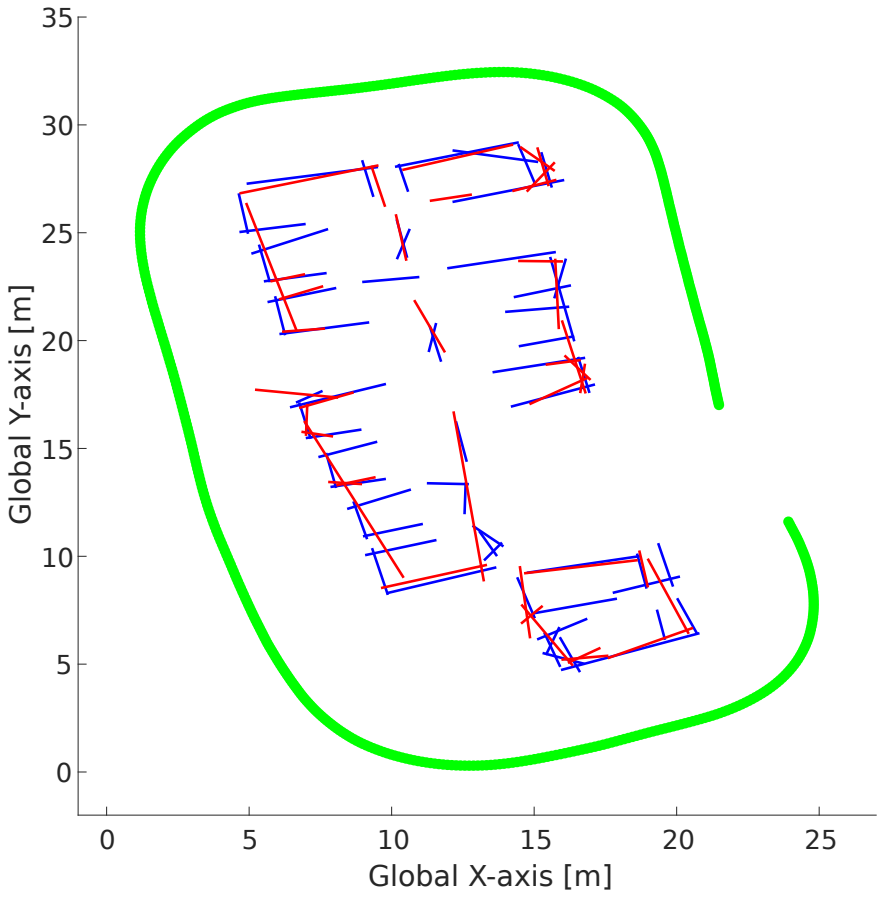


Figure A.2: *A lidar map (blue) and a radar map (red) of the first parking block at the office. The vehicle trajectory is represented by the green line.*

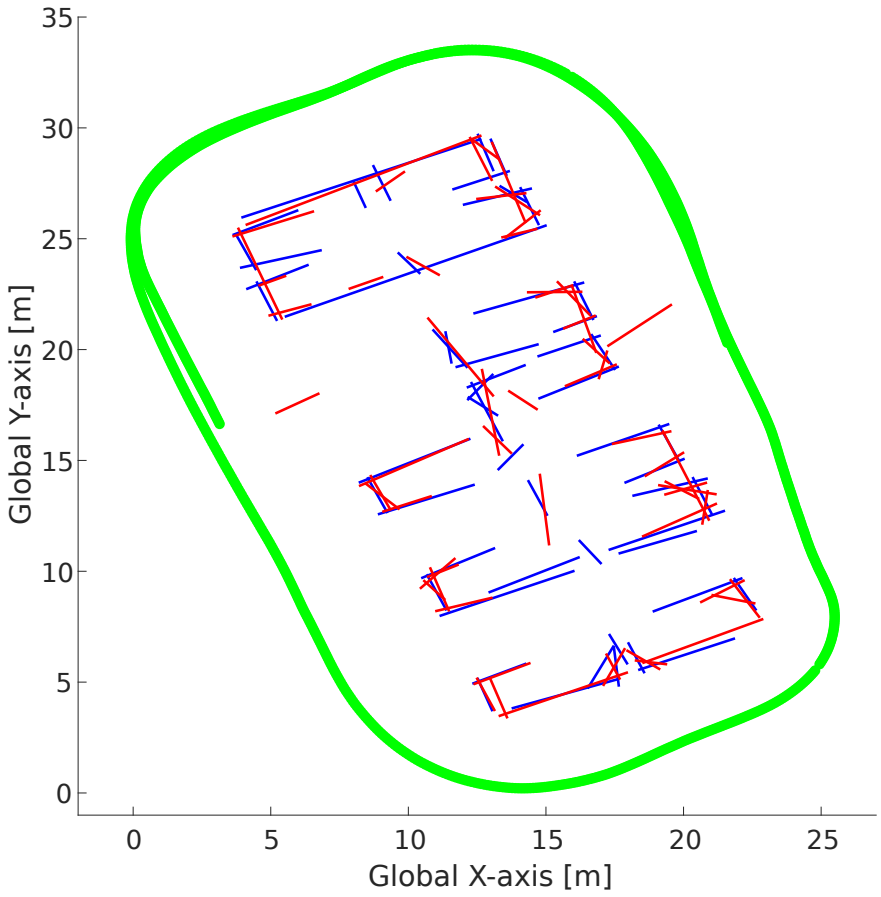


Figure A.3: *A lidar map (blue) and a radar map (red) of the second parking block at the office. The vehicle trajectory is represented by the green line.*