Hochschule Konstanz
Department of Electrical Engineering
and Information Technology

Tracking of Spline Modeled Extended Objects Using Random Finite Sets

Master Thesis

at

Institute of System Dynamics,
HTWG Konstanz

Study Program: Electrical Systems (EIM)

Submitted by:	Tim Baur, B. Eng.
Enrolment number:	297172
Supervisors:	Prof. Dr. Johannes Reuter
	Stefan Wirtensohn, M. Eng.
Processing period:	October 2018 – April 2019
Submission date:	03.04.2019

Declaration on oath

I hereby declare that I have written this work independently without using inadmissible outside help and that I have not used any other sources and aids than those indicated.

Date, Location	Signature

Acknowledgments

First, I would like to thank my supervisor Prof. Dr. Johannes Reuter
for the opportunity to write my master thesis in a field of research
that is such an important area for topical and future autonomous systems.
I would like to thank him for his guidance, suggestions and the trust he put in me.

I also want to thank my second supervisor Stefan Wirtensohn
for his support and the many suggestions that helped me a lot.
His stimulations on many problems I had during my studies
made a major contribution on the success of this thesis.

Finally, I would like to thank Julian Böhler
my fellow student during my master studies.
As he wrote his master thesis in parallel with mine,
we shared an office and had many helpful discussions.

Abstract

In the field of autonomously driving vehicles the environment perception containing dynamic objects like other road users is essential. Especially, detecting other vehicles in the road traffic using sensor data is of utmost importance. As the sensor data and the applied system model for the objects of interest are noise corrupted, a filter algorithm must be used to track moving objects. Using LIDAR sensors one object gives rise to more than one measurement per time step and is therefore called extended object. This allows to jointly estimate the objects, position, as well as its orientation, extension and shape. Estimating an arbitrary shaped object comes with a higher computational effort than estimating the shape of an object that can be approximated using a basic geometrical shape like an ellipse or a rectangle. In the case of a vehicle, assuming a rectangular shape is an accurate assumption.

A recently developed approach models the contour of a vehicle as periodic B-spline function [1]. This representation is an easy to use tool, as the contour can be specified by some basis points in Cartesian coordinates. Also rotating, scaling and moving the contour is easy to handle using a spline contour. This contour model can be used to develop a measurement model for extended objects, that can be integrated into a tracking filter. Another approach modeling the shape of a vehicle is the so-called bounding box that represents the shape as rectangle.

In this thesis the basics of single, multi and extended object tracking, as well as the basics of B-spline functions are addressed. Afterwards, the spline measurement model is established in detail and integrated into an extended Kalman filter to track a single extended object. An implementation of the resulting algorithm is compared with the rectangular shape estimator. The implementation of the rectangular shape estimator is provided. The comparison is done using long-term considerations with Monte Carlo simulations and by analyzing the results of a single run. Therefore, both algorithms are applied to the same measurements. The measurements are generated using an artificial LIDAR sensor in a simulation environment.

In a real-world tracking scenario detecting several extended objects and measurements that do not originate from a real object, named clutter measurements, is possible. Also, the sudden appearance and disappearance of an object is possible. A filter framework investigated in recent years that can handle tracking multiple objects in a cluttered environment is a random finite set based approach. The idea of random finite sets and its use in a tracking filter is recapped in this thesis. Afterwards, the spline measurement model is included in a multi extended object tracking framework. An implementation of the resulting filter is investigated in a long-term consideration using Monte Carlo simulations and by analyzing the results of a single run. The multi extended object filter is also applied to artificial LIDAR measurements generated in a simulation environment.

The results of comparing the spline based and rectangular based extended object trackers show a more stable performance of the spline extended object tracker. Also, some problems that have to be addressed in future works are discussed. The investigation of the resulting multi extended object tracker shows a successful integration of the spline measurement model in a multi extended object tracker. Also, with these results some problems remain, that have to be solved in future works.

Table of contents

I List of figures

II List of tables

III List of abbreviations

CPHD	Cardinalized probability hypothesis density
CTRV	Constant turn rate and velocity
CV	Constant velocity
EKF	Extended Kalman filter
FISST	Finite set statistics
GM	Gaussian mixture
IID	Independent identically distributed
JPDA	Joint probabilistic data association
LIDAR	Light detection and ranging
MHT	Multi hypothesis tracking
OSPA	Optimal subpattern assignment
PDF	Probability density function
PHD	Probability hypothesis density
RADAR	Radio detection and ranging
RFS	Random finite set
RMSE	Root mean square error
UKF	Unscented Kalman filter

IV General notation conventions

x, z	Scalar
$\boldsymbol{x}, \boldsymbol{z}$	Vector
X, Z	Matrix
$\boldsymbol{X}, \boldsymbol{Z}$	Set
$\boldsymbol{x}_{k\|k-1}, \boldsymbol{z}_{k\|k-1}$	Prediction at time k
$\boldsymbol{x}_{k\|k}, \boldsymbol{z}_{k\|k}$	Update at time k
$\mathcal{N}(\boldsymbol{x}, \hat{\boldsymbol{x}}, \Sigma)$	Multivariate normal distribution with mean $\hat{\boldsymbol{x}}$ and covariance matrix Σ
$x_k, \boldsymbol{x}_k, X_k, \boldsymbol{X}_k$	Quantity at time k
$\boldsymbol{X}^k, \boldsymbol{Z}^k$	Set of quantities up to time k
$\boldsymbol{x}_k^T, X_k^T$	Transposed quantity at time k
$\mathcal{F}(\boldsymbol{X})$	Set of all finite subsets of $\boldsymbol{X}$

V List of symbols

$\boldsymbol{x}_k$	Object state
$\boldsymbol{z}_k$	Measurement
$\boldsymbol{y}_k$	Measurement source
$f(\boldsymbol{x}_k)/F$	System state transition function/matrix
$h(\boldsymbol{x}_k)/H$	Measurement model function/matrix
$\boldsymbol{v}_k$	System noise
Q_k	System noise covariance matrix
$\boldsymbol{w}_k$	Measurement noise
R_k	Measurement noise covariance matrix
$\hat{\boldsymbol{x}}_k$	Estimated object state
P_k	Estimation covariance matrix
S_k	Innovation covariance matrix
K_k	Kalman gain
λ	Rate of Poisson distributed random number
$v_k(\boldsymbol{x})$	Multi object state intensity
ω_k	Weight of a GM component
$b_k(\boldsymbol{x})$	Birth RFS approximated as GM
$\kappa_k(\boldsymbol{x})$	Intensity of clutter RFS
J_k	Number of GM components
p_S	Probability of survival
p_D	Probability of detection
P_{MD}	Probability of multipath detection
T	Pruning threshold
U	Merging threshold
$J_{\max}$	Maximum number of GM components after merging and pruning
$\gamma(\boldsymbol{x})$	Rate of object generated measurements
p	Partition of a measurement set
$\boldsymbol{W}$	Cell of a partition
ω_p	Partition normalization factor
$d_{\boldsymbol{W}}$	Cell normalization factor
$\delta_{i,j}$	Kronecker delta
τ	B-spline walk parameter
$C(\tau)$	B-spline contour function
P	B-spline basis points
R_φ	Rotation matrix
S^C	Scaling matrix
$\boldsymbol{m}_k$	Object center
φ_k	Heading angle
$\hat{\boldsymbol{z}}_k$	Measurement in local coordinates
$\hat{\boldsymbol{y}}_k$	Measurement source in local coordinates
$k(\tau)$	Knot defining the active basis points
ε	Distance parameter for DBSCAN algorithm
σ	Standard deviation
r	Range of measurement
ϑ	Azimuth of measurement

1 Introduction

This chapter introduces the topics that are highlighted in this master thesis. Section 1.1 describes the motivation for working in this field of research. In section 1.2 the application for the presented algorithm is described. Section 1.3 is about to introduce the algorithm worked out and tested in this thesis, while the last section in this chapter summarizes the structure of the thesis.

1.1 Motivation

A brief research on autonomous driving gives an idea of the efforts being made to advance this topic. And this is only one field of research for the use of autonomous systems. In robotics in general, but also in medical technology or the investigation of biological processes, work is being done to automate processes. And these are just a few examples.

As the level of automation increases, the system environment needs to be captured more accurately. In the case of autonomous robots that are supposed to act freely in their environment, this means that fixed objects must be captured, and other moving objects must be tracked. The focus of this thesis is on tracking other moving objects.

The environment can be detected by various sensors. These include sensors that generate a point cloud, such as light detection and ranging (LIDAR) or radio detection and ranging (RADAR) sensors, but also camera systems. This thesis examines a 2D-LIDAR sensor as a measurement system.

When measuring moving objects with these systems, different problems can occur. First, the number of measured objects and the number of measurements each object generates is unknown. At the beginning of this research area, distant objects were measured and tracked. They generated only one measurement per time step and could therefore be modeled as point objects. But the further development of sensor technology has led to several measurements being generated per object and time step at a sufficiently short distance between the sensor and the measuring object. These objects are called extended objects.

Furthermore, it must be assumed that all measurements are afflicted with an error which must be compensated. The occurrence of incorrect measurements must also be considered. All these reasons lead to the necessity of a filter, which generates an estimate of the state of the individual objects from the measurement data at each time step. A solution to this problem is provided by the Bayesian filter theory.

1.2 Application description

In general, the solution to the multi extended object problem, presented in this thesis, can be applied to many applications. Here the tracking solution is applied to the automotive sector. The aim is to track extended objects with almost rectangular extension, with the possibility of detecting both small and large vehicles. Accordingly, the extension model must be scalable so that both a car and a truck can be detected.

For this application a 2D-LIDAR sensor is used. This sensor measures the distance and the angle of the detected measuring point, so the sensor measures in polar coordinates. Both the distance and the angle are not measured exactly.

Figure 1.1 shows an example of the examined application. The LIDAR sensor with a resolution of two degrees and a range of 30 meters is located in the origin. In this example, three objects

with different extensions are detected. The measurements are noise corrupted as mentioned above.

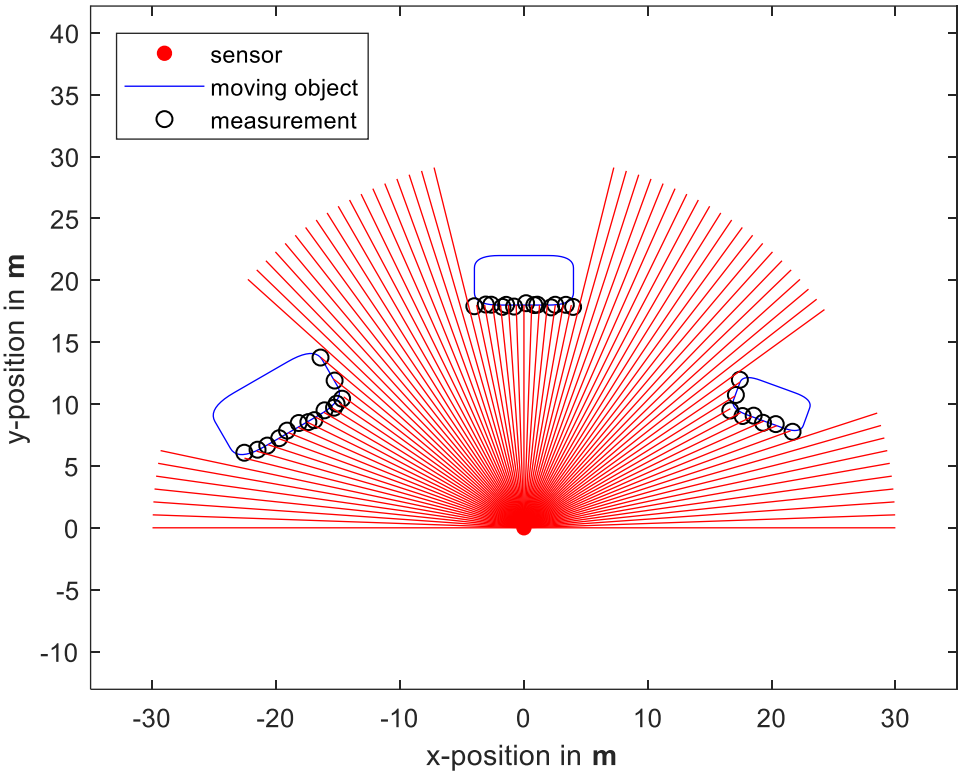

Figure 1.1: Example of the application presented in this thesis

With this sensor, either one or two sides of the object can be detected if it is assumed that there are no reflections of the laser beams. Otherwise, multipath detections at the sides of the objects away from the sensor would occur.

Situations not shown in this figure are false measurements and occlusions. False measurements, which are called clutter in the following, are randomly occurring measurements that cannot be assigned to a real object. In the case of occlusions, two objects are located one behind the other, so the sensor cannot capture the entire rear object.

1.3 Investigated approach

In the development of an approach for the multi extended object problem the measurement model of the extended object is a crucial part. In this thesis an approach is presented where the extension of the object is modeled as a quadratic periodic B-spline function. This makes it possible to display the contour as a rectangle with rounded edges as shown in Figure 1.1. This shape is ideal for a vehicle, but cannot be used for many other objects, therefore the contour function must be adapted. B-spline functions are easy to handle, because the shape can be modeled in Cartesian coordinates and the contour can easily be scaled in x and y Dimension.

To use the measurement model in a tracking scenario, it must be integrated into a Bayesian filter framework. In the case of a single object tracker the well-known extended Kalman filter is used. In the multi object case there are many scenarios that do not need to be considered in the single object case. First it is unknown how many objects of interest are situated in the surveillance area at every time step. It is always possible that a new object is born. Also, the

disappearance of an object is possible in every time step. Therefore, the assignment of the measurements to the given and new tracks is a very complicated part. Furthermore, it can never be assumed that a measurement actually belongs to a real object, since clutter measurements can occur as well. Various approaches exist to all these problems. The one presented in this thesis is a Gaussian mixture probability hypothesis density (GM-PHD) filter.

In order to evaluate the performance of the approach in the single object case, it is compared with an extension model that represents the shape as a rectangle. The two algorithms are compared mainly by Monte Carlo simulations.

1.4 Structure of the thesis

The remainder of this thesis is structured as follows. Chapter 2 gives a rough overview of the Bayesian filter theory. Therefore, a brief review of the derivation of the Bayes filter and its linear and nonlinear parametric solutions are given. Chapter 3 then continues with the problem of modeling extended objects. The ideas of different approaches, as well as the representation of the contour as a rectangle, are presented. Chapter 4 then discusses the multi object tracking problem. Here the idea of random finite sets is introduced and integrated in the Bayesian filter theory. The chapter ends with the presentation of a solution for the multi object tracking problem, the GM-PHD filter. The basics of splines are then explained in chapter 5. Also, the representation of the vehicle contour as a spline function is presented. In chapter 6 the Cartesian B-spline model for extended objects is introduced. Therefore, the derivation of the measurement model is performed. Finally, the integration of the measurement model in the extended Kalman and the GM-PHD filter framework is carried out in this chapter. The performance evaluation of the developed approach is presented in chapter 7. The thesis ends with the conclusions and ideas for future work in chapter 8.

2 Bayesian filter theory

In this chapter the basics of the Bayesian filter theory are discussed. The theory is based on the considerations for the estimation of states out of noisy measurement data. This estimation is provided by a Bayes filter, which is presented in section 2.1. Since this filter is of theoretical nature and cannot be implemented in this form, filter solutions for linear and nonlinear motion and measurement models, addressed in section 2.2, are presented in sections 2.3 and 2.4 respectively.

2.1 The Bayes filter

In general, the Bayes filter [2, pp. 22-25] [3, pp. 41-42] [4, pp. 13-15] is a framework for the state estimation out of noise corrupted measurement data. Assuming there is a measurement set $\boldsymbol{Z}^k = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \dots, \boldsymbol{z}_k\}$ up to time k. Based on these measurements, an estimation of the states of the object at the respective time is to be given. The set of states is denoted as $\boldsymbol{X}^k = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k\}$. It must be assumed that the measurements are noisy and therefore not accurate. Based on this assumption, the state estimation is to be given in the form of a probability distribution $p(\boldsymbol{X}^K|\boldsymbol{Z}^k)$ under the assumption of known measurement data. Since this distribution is very difficult to determine in reality, Bayes' theorem is applied in a first step. The probability of the kth set of states, given a measurement set, is then given as

$$p(\boldsymbol{X}^k|\boldsymbol{Z}^k) = \frac{p(\boldsymbol{Z}^k|\boldsymbol{X}^k) \cdot p(\boldsymbol{X}^k)}{p(\boldsymbol{Z}^k)}. \tag{2.1}$$

However, the information of the previous system states is not used adequately in this

equation of the states. The goal is a recursive presentation of the states. In order to obtain such a presentation, three basic assumptions for the further calculations are assumed below:

1. Principle of causality: It can be assumed that the measurements up to time $k-1$ are not influenced by the system state at time k.
2. Markov characteristic: The system state at time k is only dependent on the system state at time $k-1$. All previous system states can be ignored.
3. Measuring characteristic: The measurement at time k is only dependent on the system state at time k.

The application of these assumptions results in the presentation of a recursive Bayes rule and is then given as

$$p(\boldsymbol{X}^k|\boldsymbol{Z}^k) = \frac{p(\boldsymbol{z}_k|\boldsymbol{x}_k)}{p(\boldsymbol{z}_k|\boldsymbol{Z}^{k-1})} \cdot\ p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})\ \cdot p(\boldsymbol{X}^{k-1}|\boldsymbol{Z}^{k-1})\,. \tag{2.2}$$

According to this recursive Bayes rule, all previous system states are calculated in each iteration step, which represents an unnecessary computational effort. A representation in which the current state $\boldsymbol{x}_k$ is determined from the previous one $\boldsymbol{x}_{k-1}$ is desirable. This is achieved by considering the marginal distribution. The integration over all past states $\boldsymbol{x}_0, \dots, \boldsymbol{x}_{k-1}$ results in the algorithm of the Bayes filter, represented by the two following equations.

1. Prediction step or Chapman-Kolmogorov equation:

$$p(\boldsymbol{x}_k|\boldsymbol{Z}^{k-1}) = \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})\ \cdot p(\boldsymbol{x}_{k-1}|\boldsymbol{Z}^{k-1})\, d\boldsymbol{x}_{k-1}\,. \tag{2.3}$$

2. Update step:

$$p(\boldsymbol{x}_k|\boldsymbol{Z}^k) = \eta\ \cdot p(\boldsymbol{z}_k|\boldsymbol{x}_k)\ \cdot p(\boldsymbol{x}_k|\boldsymbol{Z}^{k-1})\,. \tag{2.4}$$

The normalization coefficient $\eta = \frac{1}{p(\boldsymbol{z}_k|\boldsymbol{Z}^{k-1})}$ is determined by the law of total probability by integrating over all system states that could have caused the given measurement and is then given as

$$p(\boldsymbol{z}_k|\boldsymbol{Z}^{k-1}) = \int p(\boldsymbol{z}_k|\boldsymbol{x}_k) \cdot\ p(\boldsymbol{x}_k|\boldsymbol{Z}^{k-1})\, d\boldsymbol{x}_k\,. \tag{2.5}$$

The probability density calculated in the prediction step (2.3) is called the prior density while the probability density of the update step (2.4) is called the posterior density. By applying these two steps, a statement about the system state under the consideration of the given measurement set can be made. The assumptions used for this derivation are very crucial in this context. If these assumptions do not apply to the application, a Bayes filter in this form cannot be used. However, the assumptions apply in most applications, which justifies their use. The Bayes filter can only be used in this form if the probability distributions used are known. Since these are not known in most cases, assumptions must be made to use the Bayes filter in reality. These assumptions are discussed in the following sections.

2.2 The motion and measurement model

In order to make a prediction of the change of state and the associated measurement, suitable models [5, pp. 91-119] [6, pp. 267-295] [7, pp. 199-257] must be applied for both processes. In general, both the motion and the measurement model can be represented as nonlinear functions. In the case of the motion model the transition depends on the previous system state $\boldsymbol{x}_{k-1}$, some system inputs $\boldsymbol{u}_{k-1}$ and a noise term $\boldsymbol{v}_{k-1}$, so the motion model is given as $\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{v}_{k-1})$. In the remainder of this thesis the system inputs will be neglected. The noise term $\boldsymbol{v}_{k-1}$ is assumed to be additive with zero mean, so the expectation value is $\mathbb{E}(\boldsymbol{v}_{k-1}) = 0$. The transition of the system states can then be represented as

$$x_k = f(\boldsymbol{x}_{k-1}) + \boldsymbol{v}_{k-1}. \tag{2.6}$$

Frequently used models are the Constant Velocity (CV) and Constant Turn Rate and Velocity (CTRV) [8] models.

In the case of the measurement model the calculation depends on the current system state $\boldsymbol{x}_k$ and a noise term $\boldsymbol{w}_k$ and is then given as $\boldsymbol{z}_k = h(\boldsymbol{x}_k, \boldsymbol{w}_k)$. As for the transition of the system state the noise term $\boldsymbol{w}_k$ is assumed to be additive with zero mean, so the expectation value is $\mathbb{E}(\boldsymbol{w}_k) = 0$ and the measurement model can then be represented as

$$\boldsymbol{z}_k = h(\boldsymbol{x}_k) + \boldsymbol{w}_k\,. \tag{2.7}$$

In general, the measurement model contains the conversion of the system states into the actual representation of the measurement states.

In the case of linear correlations in the motion and measurement model both can be simplified. The transition of the system is then given as

$$\boldsymbol{x}_k = F \cdot \boldsymbol{x}_{k-1} + \boldsymbol{v}_{k-1} \tag{2.8}$$

with transition matrix F, while the calculation of the measurements is given as

$$\boldsymbol{z}_k = H \cdot \boldsymbol{x}_k + \boldsymbol{w}_k \tag{2.9}$$

with measurement matrix H.

2.3 A solution of the Bayes filter – The Kalman filter

Since the prior and the posterior of (2.3) and (2.4) respectively are given by general probability densities, analytical densities must be assumed in reality. In the first step a system with linear motion and measurement model is considered, so the general equations (2.8) and (2.9) can be taken into account. When considering these equations, it becomes clear that the uncertainty in the prediction of the state and the measurement is in the noise term. These noise terms are assumed to be additive and normally distributed with zero mean. Given these assumptions the distribution of the system state with a given previous system state can be specified as

$$\begin{aligned} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) &= p(\boldsymbol{v}_{k-1}) = p(\boldsymbol{x}_k - F\boldsymbol{x}_{k-1}) \\ &= \mathcal{N}(\boldsymbol{x}_k - F\boldsymbol{x}_{k-1}, 0, Q_k) = \mathcal{N}(\boldsymbol{x}_k, F\boldsymbol{x}_{k-1}, Q_k)\,. \end{aligned} \tag{2.10}$$

Here the covariance matrix of the noise $\boldsymbol{v}_{k-1}$ is referred to as Q_k. The distribution of the measurement with a given system state can then be specified as

$$p(\boldsymbol{z}_k|\boldsymbol{x}_k) = p(\boldsymbol{w}_k) = p(\boldsymbol{z}_k - H\boldsymbol{x}_k) = \mathcal{N}(\boldsymbol{z}_k - H\boldsymbol{x}_k, 0, R_k) = \mathcal{N}(\boldsymbol{z}_k; H\boldsymbol{x}_k, R_k)\,. \quad (2.11)$$

Here the covariance matrix of the measurement noise $\boldsymbol{w}_{k-1}$ is referred to as R_k. In the recursive representation of the Bayes filter, the distribution of the state estimate from the last time step is finally needed. This is also assumed to be normal distributed and is given as

$$p(\boldsymbol{x}_{k-1}|\boldsymbol{Z}^{k-1}) = \mathcal{N}\left(\boldsymbol{x}_{k-1}; \hat{\boldsymbol{x}}_{k-1|k-1}, P_{k-1|k-1}\right) \quad (2.12)$$

with the expected value $\hat{\boldsymbol{x}}_{k-1|k-1}$ and the covariance matrix $P_{k-1|k-1}$. These three normal distributions can now be used in (2.3) – (2.5). The further simplifications are not presented here but can be performed using the Gaussian product theorem [9]. The result of these simplifications is the well-known Kalman filter [10] [2, pp. 25-30] [11, pp. 56-69] [5, pp. 34-48], whose algorithm can be divided into the following three steps.

1. Prediction of the system state with corresponding covariance:

$$\hat{\boldsymbol{x}}_{k|k-1} = F \cdot \hat{\boldsymbol{x}}_{k-1|k-1} \quad (2.13)$$

$$P_{k|k-1} = F \cdot P_{k-1|k-1} \cdot F^T + Q_k \quad (2.14)$$

2. Prediction of the measurement with corresponding Innovation covariance:

$$\hat{\boldsymbol{z}}_k = H \cdot \hat{\boldsymbol{x}}_{k|k-1} \quad (2.15)$$

$$S_k = H \cdot P_{k|k-1} \cdot H^T + R_k \quad (2.16)$$

3. Update of the system state and the corresponding covariance:

$$K_k = P_{k|k-1} \cdot H^T \cdot S_k^{-1} \quad (2.17)$$

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + K_k \cdot (\boldsymbol{z}_k - \hat{\boldsymbol{z}}_k) \quad (2.18)$$

$$P_{k|k} = P_{k|k-1} - K_k \cdot H \cdot P_{k|k-1} \quad (2.19)$$

The result of the filter is an estimation of the system state $\hat{\boldsymbol{x}}_{k|k}$ from a given measurement $\boldsymbol{z}_k$ with corresponding covariance $P_{k|k}$. However, not all measurement and motion models can be assumed to be linear. Furthermore, the assumption that all distributions can be assumed as Gaussians does not apply in all systems. Approaches to solve these problems are discussed in the following chapter.

2.4 Bayes filter solutions for nonlinear models

When using nonlinear models, the general equations (2.6) and (2.7) must be assumed as motion and measurement models. Once again, the noise terms are assumed to be additive and Gaussian with zero mean. In general, two approaches are known to determine the distribution of the state estimation in this case. The idea of the first approach is to linearize the models. Therefore, the Jacobi matrix of the motion model and the measurement model must be determined in each step. The filter that results from this approach is called the extended Kalman filter (EKF). The idea of the second approach is to approximate the distribution as normal distribution using the expectation value and the standard deviation, its parameters. Therefore, a sufficient number of so-called sigma-points must be drawn in each step, in order to be able to display the parameters of the Gaussian distribution. The number of sigma-points depends on the dimension of the normal distribution. The filter that results from this approach is called the unscented Kalman filter (UKF).

The nonlinear approach used in this thesis is an EKF. The UKF is not used in this thesis, thus it is not discussed further here. For the derivation and the UKF filter equations, the reader is referred to [12] [2, pp. 36-43]. The derivation of the EKF approach can also be found in the literature [11, pp. 106-116] [2, pp. 31-36] [5, pp. 48-54]. In the following only the filter equations are shown. Since the filter update of the EKF is identical to the update of the linear Kalman filter, only the prediction of the system state and the measurement are shown.

1. Calculation of the Jacobi matrix of f:

$$F_k = \nabla_{\boldsymbol{x}^T} f(\boldsymbol{x})|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k-1|k-1}} \tag{2.20}$$

2. Prediction of the system state with corresponding covariance:

$$\hat{\boldsymbol{x}}_{k|k-1} = f(\hat{\boldsymbol{x}}_{k-1|k-1}) \tag{2.21}$$

$$P_{k|k-1} = F_k \cdot P_{k-1|k-1} \cdot F_k^T + Q_k \tag{2.22}$$

3. Calculation of the Jacobi matrix of h:

$$H_k = \nabla_{\boldsymbol{x}^T} h(\boldsymbol{x})|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k|k-1}} \tag{2.23}$$

4. Prediction of the measurement with corresponding Innovation covariance:

$$\hat{\boldsymbol{z}}_k = h(\hat{\boldsymbol{x}}_{k|k-1}) \tag{2.24}$$

$$S_k = H_k \cdot P_{k|k-1} \cdot H_k^T + R_k \tag{2.25}$$

5. Update of the system state and the corresponding covariance according to (2.17) – (2.19)

When processing systems with arbitrary complex distributions, the performance of a Kalman filter can become very poor. In these cases, the distributions often have several maxima, so they cannot be described accurate by a normal distribution. If such a distribution is present, sequential Monte Carlo methods [2, pp. 46-53] [13] [14] or particle filters lead to the desired result. The idea of this method is to represent the distribution by a large number of samples, also called particles. In regions of high probability many particles should be present and in regions of low probability few particles. With this method, the performance gain is bought with a significantly higher computational effort, that can be parallelized very well.

3 Extended object tracking

This chapter gives an overview of the problem of modeling extended objects [15]. In the first section a precise definition of the problem is given. In the second section, the best-known approaches to solve this problem are presented. Finally, the measurement model used for comparison, the rectangular shape estimator, is presented in section 3.3.

3.1 Problem definition

By expanding the field of applications and improving sensor technology, different tracking scenarios must be distinguished in the area of environment perception [15]. Definitions for these different scenarios are given below.

- Point object tracking: Each object within the surveillance area generates at most one measurement per time step.
- Extended object tracking: Each object within the surveillance area can generate several measurements per time step with the measurements distributed spatially around the object. Each measurement is generated by a measurement source located on the surface of the extended object.
- Group object tracking: Some objects within the surveillance area can consist of multiple objects where each object can be modeled as an extended object or a point object. The objects in the group share some motion characteristics e.g. moving in the same direction with a similar velocity. They can therefore be treated as one object and are not tracked individually.

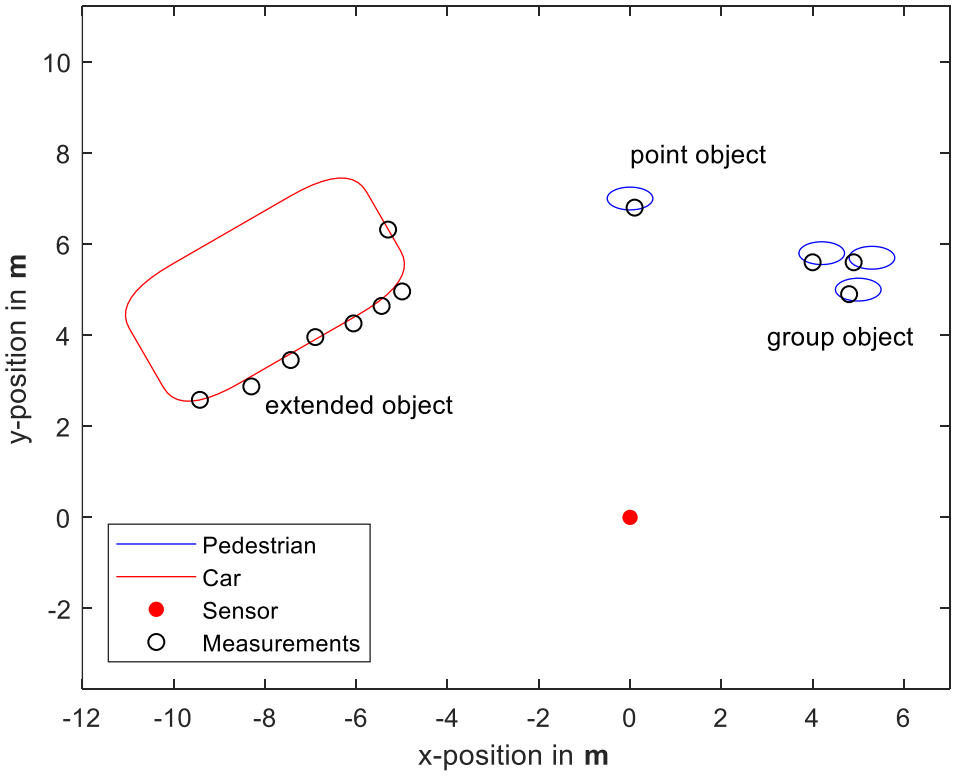

Figure 3.1: Illustration of different tracking scenarios

The different scenarios are shown in Figure 3.1 with a car as extended object, a pedestrian as point object and a group of pedestrians as group object. The group tracking scenario is illustrated with three point objects in this example, but could also be a group of extended objects, or a mixture of extended and point objects.

3.1.1 The object state

When modeling the state of an extended object, the position, kinematics and extension must be represented in a state vector $\boldsymbol{x_k}$ for time step k [15]. In the following example, the extension is assumed to be a rectangle to illustrate the state vector which, in this example, would be given as

$$\boldsymbol{x_k} = (x_k\,, y_k\,, v_k\,, \psi_k\,, L_k\,, W_k)^T\,. \tag{3.1}$$

The object state is illustrated in Figure 3.2.

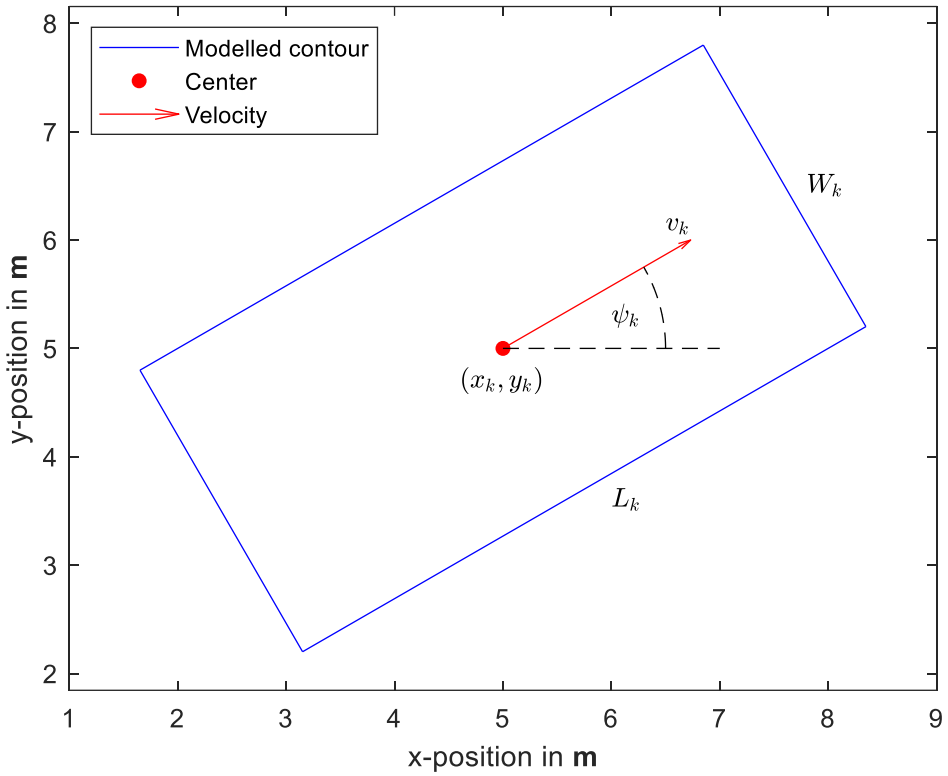

Figure 3.2: Illustration of the object state

In this representation, the vector $(x_k\,, y_k)^T$ represents the position of the center point. The kinematic is represented by the velocity v_k, the orientation by the heading angle ψ_k and the extension of the object is represented by the length L_k and the width W_k of the rectangle.

The composition of the object state depends on many factors. In many applications it is sufficient to model the objects position in 2D coordinates, however, the third spatial dimension cannot be ignored, when tracking flying objects. The object state also depends on the choice of the motion model. In a CV model the orientation would be neglected, while in other models the steering angle of the moving object would be added. Furthermore, the extension state depends on the choice of the shape to be modeled. In many applications it is sufficient to model the extent with a simple geometric shape like a rectangle or an ellipse. In

other applications the choice of a more general shape would be more appropriate. So, the composition of the object state must be reconsidered in every application separately.

3.1.2 Modeling the measurements

Finding a suitable measurement model is a crucial part when tracking extended objects. Depending on the sensor and the relative positioning of the sensor and the object to be measured, a different number of measurements is generated from different measurement sources. In addition, all detections are measured with an additive random noise and the assignment of a measurement to a specific measurement source is unknown. Modeling the measurement process, summarized in [15], is therefore a highly complex problem. When tracking extended objects, it is assumed that a measurement set $\boldsymbol{Z}_k = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \dots, \boldsymbol{z}_{n_k}\}$ of n_k measurements is available at each time step k. In this case the conditional distribution

$$p(\boldsymbol{Z}_k|\boldsymbol{x}_k) \tag{3.2}$$

is searched for modeling the measurements. Thus, the distribution of all measurements given a specific object state is to be calculated. In a first step, it can be assumed that the detections are measured independently of each other so the measurement likelihood can be stated as

$$p(\boldsymbol{Z}_k|\boldsymbol{x}_k) = \prod_{i=1}^{n_k} p(\boldsymbol{z}_{k,i}|\boldsymbol{x}_k)\,. \tag{3.3}$$

Further considerations can be divided into two different approaches. In the first approach, the detections are modeled as Poisson point process [16]. It is assumed that the number of detections generated by the measurement object follows a Poisson distribution that depends on the object state. The localization of the measurements then follows a spatial distribution. This approach completely avoids a direct assignment of the measurements to the corresponding measurement sources. In the literature this assignment is referred to the data association problem in extended object tracking. The Poisson point process approach is used to model the extension of the tracked object as a random matrix [17], which will be briefly introduced later.

In the second approach, each measurement $\boldsymbol{z}_i$ is modeled as a detection of a measurement source $\boldsymbol{y} \in \boldsymbol{Y}_k = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_{n_k}\}$ located on the surface of the object. The distribution of a measurement given an object state $\boldsymbol{x}_k$ is then calculated as the total probability

$$p(\boldsymbol{z}_i|\boldsymbol{x}_k) = \int p(\boldsymbol{z}_i|\boldsymbol{y}) \cdot p(\boldsymbol{y}|\boldsymbol{x}_k)\, d\boldsymbol{y} \tag{3.4}$$

of all measurement sources $\boldsymbol{y}$ that could have caused the measurement. The data association problem is a crucial part in the use of this approach. A well-known possibility of modeling measurement sources is the random hypersurface model [18], where each measurement source is assumed to be an element of a randomly generated hypersurface. In mathematics, hypersurfaces denote objects of codimension one. In 3-dimensional space this object would be a surface and in 2-dimensional space a curve. The random hypersurface model will also be briefly introduced later.

3.1.3 Modeling the shape

When modeling the shape of an extended object, different levels of difficulty must be distinguished. The description of the shape can therefore be less to highly complex. These

different levels are reflected in different approximations and algorithms [15]. The question of the choice of the appropriate modeling of the shape for the application cannot be answered easily. With highly complex approaches, the calculation effort increases considerably, whereas less complex approaches often do not generate the information that is needed. The goal is therefore to choose an approach that generates the necessary information with as less computational effort as possible. In the following, the different levels of complexity are briefly explained.

- No shape modeling: For example, if an application does not require information's about the extension of the tracked objects, a model can be selected that completely neglects the extension and estimates only the kinematic components. Since the sensor can generate a large number of measurements per object and time step, information is deliberately discarded in this case.
- Standard geometrical shape modeling: A common approach of describing the shape of an extended object is to use a basic geometric shape for the object such as ellipses or rectangles. These models are sufficient for some applications and are not too complex, so that the computation can be carried out in a reasonable time.
- Arbitrary shape modeling: The most complex models are those that are able to describe an arbitrary shape of the extended object. These models are able to describe objects with unusual shapes. The generality of the approach, however, also entails a high computational effort.

The choice of the complexity level depends on the application and the objects to be tracked. In applications for tracking cars or ships, it is normally sufficient to describe the extension with a standard shape, such as a rectangle or an ellipse respectively. On the other hand, the extension of a pedestrian can be neglected due to size differences in these applications. If objects are to be tracked whose shape resembles neither a rectangle nor an ellipse, models must be selected to describe an arbitrary shape. An example of such an object can be a plane whose shape resembles a cross. When describing a cross with a rectangle or an ellipse, big mistakes can be made. In the literature there are two different approaches for the description of arbitrary shapes. The first describes the shape by a curve, while the second describes the shape by composite ellipses. Literature examples for all three complexity levels can be found in [15].

3.2 Popular approaches to extended object tracking

In this section two popular approaches to the problem of tracking extended objects are addressed. In the first subsection the random matrix approach, where the extension of the object is represented as ellipse using a spatial distribution, is briefly outlined. The random matrix extension model has been investigated in a single object tracking framework [17], as well as in a multi object tracking scenario [19]. In the second subsection the random hypersurface model is addressed [18]. Using this approach, the state of arbitrary shaped objects can be estimated in a tracking filter.

3.2.1 The random matrix approach

The random matrix approach is an example of modeling the measurements originating from an extended object using a spatial distribution. The state of the extended object is split up to a state vector $\boldsymbol{x}_k$, containing the position, orientation and kinematics, and an extent matrix X_k as $\xi_k = (\boldsymbol{x}_k, X_k)$. The extent matrix is modeled as symmetric and positive definite square

matrix with $X_k \in \mathbb{R}^{d\times d}$, where d is the objects position dimension. With the position e.g. in 2-dimensional space, the extent matrix is of dimension $X_k \in \mathbb{R}^{2\times 2}$. A symmetric and positive definite square matrix describes an ellipse with the half axes defined using the Eigen values of the matrix X_k. Integrating the random matrix measurement model in a Bayesian tracking framework, the aim is to jointly estimate the objects position, kinematics and extension as $p(\xi_k|\boldsymbol{Z}^k)$ with the set of measurements up to time k denoted as $\boldsymbol{Z}^k$. This probability density function can be split up to the product

$$p(\xi_k|\boldsymbol{Z}^k) = p(\boldsymbol{x}_k, X_k|\boldsymbol{Z}^k) = p(\boldsymbol{x}_k|X_k, \boldsymbol{Z}^k) \cdot p(X_k|\boldsymbol{Z}^k) \tag{3.5}$$

where $p(\boldsymbol{x}_k|X_k, \boldsymbol{Z}^k)$ is a vector variate density and $p(X_k|\boldsymbol{Z}^k)$ a matrix variate density. In Bayesian probability theory the posterior and prior probability are called conjugate distributions if the probability distributions are in the same probability family. In the case of Gaussian distributed measurements the conjugate prior for the mean of the measurement is Gaussian distributed, whereas the covariance is inverse Wishart distributed [15] [17]. The original measurement model for the random matrix approach [17] applies the extent matrix X_k as covariance matrix of the measurement process. As the measurement process is modeled using a Gaussian distribution it is given as

$$p(\boldsymbol{z}_k|\boldsymbol{x}_k, X_k) = \mathcal{N}(\boldsymbol{z}_k; H_k\boldsymbol{x}_k, X_k)\,. \tag{3.6}$$

Using this expression, the spatial distribution of the measurements is given through the covariance matrix X_k that implies an elliptical extent. Furthermore, the probability density functions of (3.5) can be modeled as product of a Gaussian and inverse Wishart distribution. The random matrix approach provides a robust possibility of tracking extended objects as it completely avoids a measurement to the objects contour or measurement source association. Approaches that apply this association can completely fail if the prediction step is not precise enough. The way of modeling the objects shape as ellipse seems limiting, indeed it is applicable to many real-world scenarios like tracking of pedestrians, cyclists, ships or boats. An overview of investigated scenarios is given in [15]. However, the elliptical shape still can reach a limit if the shape of the object is too disparate to an ellipse, or in the case of closely spaced objects where the shape needs to be estimated very accurately. In those cases, it is necessary to be able of estimating an arbitrary shaped object, which is addressed in the next subsection.

3.2.2 The random hypersurface approach

The random hypersurface model originally proposed in [18] is a method of modeling the measurement source as an element of a randomly generated hypersurface. In a d-dimensional space a hypersurface is a surface of dimension $d-1$. Thus, a hypersurface in 2-dimensional space is a line. For a random set measurement model, the shape of the extended object is assumed to be of the form

$$\mathcal{O}(\boldsymbol{p}_k) = \{\boldsymbol{z} \in \mathbb{R}^2 | g(\boldsymbol{z}, \boldsymbol{p}_k) \leq 0\} \tag{3.7}$$

where $\boldsymbol{p}_k$ is a parameter vector and $g(\boldsymbol{z}, \boldsymbol{p}_k)$ is the function defining the shape of the extended object. The points belonging to the object are therefore all the points on the boundary or with a distance smaller than the boundary. For a circle the parameter vector would contain the center and the radius. Now the object generates a set of measurements

$\boldsymbol{Z}_k = \{\boldsymbol{z}_{k,l}\}_{l=1}^{n_k}$ in a specific time step. The measurement source to each measurement is assumed to be an element of the measurement set

$$\mathcal{M}(\boldsymbol{p}_{k,l}^m) = \{\boldsymbol{y} \in R^2 | C(\boldsymbol{y}, \boldsymbol{p}_{k,l}^m)\} \tag{3.8}$$

where $\boldsymbol{p}_{k,l}^m$ is the parameter vector similar to the parameter vector of the set defining the objects shape and $C(\boldsymbol{y}, \boldsymbol{p}_{k,l}^m)$ is a constraint similar to the function specifying the objects shape. Given the two parameter vectors $\boldsymbol{p}_k$ and $\boldsymbol{p}_{k,l}^m$ of the shape and measurement set respectively, the random set measurement model is specified as $f(\boldsymbol{p}_{k,l}^m | \boldsymbol{p}_k)$. The random set measurement model is a generalization of the spatial distribution model like the random matrix approach, as the measurement set would be the singleton $\mathcal{M}(\boldsymbol{p}_{k,l}^m) = \{\boldsymbol{p}_{k,l}^m\}$ for a spatial distribution model. In the case of a random hypersurface model the measurement set is assumed to be a hypersurface respectively a scaled version of the object's boundary specified by a random scaling factor $s_{k,l}$. The measurement source is then specified by a 1-dimensional distribution over the hypersurface. A summary to the random hypersurface model is also given in [15]. In [18] the random hypersurface model is applied to elliptical shaped extended objects. The measurement set is therefore a scaled version of the boundary ellipse. A comparison of the random matrix approach and the random hypersurface model for extended objects is given in [20]. An expansion of the random hypersurface model for elliptical shape to star convex shapes is addressed in [21]. A star convex set is a set where the connecting line of the center to a point on the boundary is completely in the set. A star convex boundary can be specified using a radial function $r(\phi)$. A radial function gives the distance between the center and the contour point depending on the given angle ϕ. Using a Fourier series expansion of degree N_F, a periodic radial function can be specified as

$$r(\boldsymbol{a}_k, \boldsymbol{b}_k, \phi) = \frac{a_k^{(0)}}{2} + \sum_{j=1}^{N_F} a_k^{(j)} \cos(j\phi) + b_k^{(j)} \sin(j\phi)\,. \tag{3.9}$$

When tracking an arbitrary star convex shaped extended object, the degree of the Fourier series expansion can be fixed, and the parameters can be estimated in a Bayesian tracking filter. As the random hypersurface model is able to model arbitrary shaped objects it comes with a much higher computational effort than a spatial distribution model.

3.3 The rectangular shape estimator

When tracking vehicles in the road traffic, the position, orientation, kinematics and extension is to be estimated using sensor data. An intuitive approach is to model the extension of a vehicle as rectangle, also known as bounding box. A tracking filter for rectangular objects briefly addressed in this section is given in [22]. The object state $\boldsymbol{x}_k$ can be used according to (3.1) in this algorithm. The measurements $\boldsymbol{Z}_k = \{\boldsymbol{z}_{k,i}\}_{i=1}^{n_k}$ are assumed to be generated from specific points on the object's contour. The measurement sources $\boldsymbol{Y}_k = \{\boldsymbol{y}_{k,i}\}_{i=1}^{n_k}$ on the contour are assumed to be the closest points to the measurements, which can be computed using an orthogonal projection on the bounding box. An illustration of the measurement prediction is given in Figure 3.3. Using the distance $d_{k,i}$ between the measurement $\boldsymbol{z}_{k,i} = \left(x_{k,i}^{(z)}, y_{k,i}^{(z)}\right)^T$ and the measurement source $\boldsymbol{y}_{k,i} = (x_{k,i}^{(y)}, y_{k,i}^{(y)})$, as well as the orientation ψ_k of the bounding box, the measurement source can be computed as

$$\begin{pmatrix} x_{k,i}^{(y)} \\ y_{k,i}^{(y)} \end{pmatrix} = \begin{pmatrix} d_{k,i} \cdot \cos\left(\hat{\psi}_{k,i}\right) + x_{k,i}^{(z)} \\ d_{k,i} \cdot \sin\left(\hat{\psi}_{k,i}\right) + y_{k,i}^{(z)} \end{pmatrix}. \tag{3.10}$$

By denoting the box sides with a counter clockwise line identification index $j \in \{0, \dots, 3\}$, where 0 is the front, the angle $\hat{\psi}_{k,i}$ of (3.10) can be calculated as $\hat{\psi}_{k,i} = \psi_k + \frac{\pi}{2} \cdot j$.

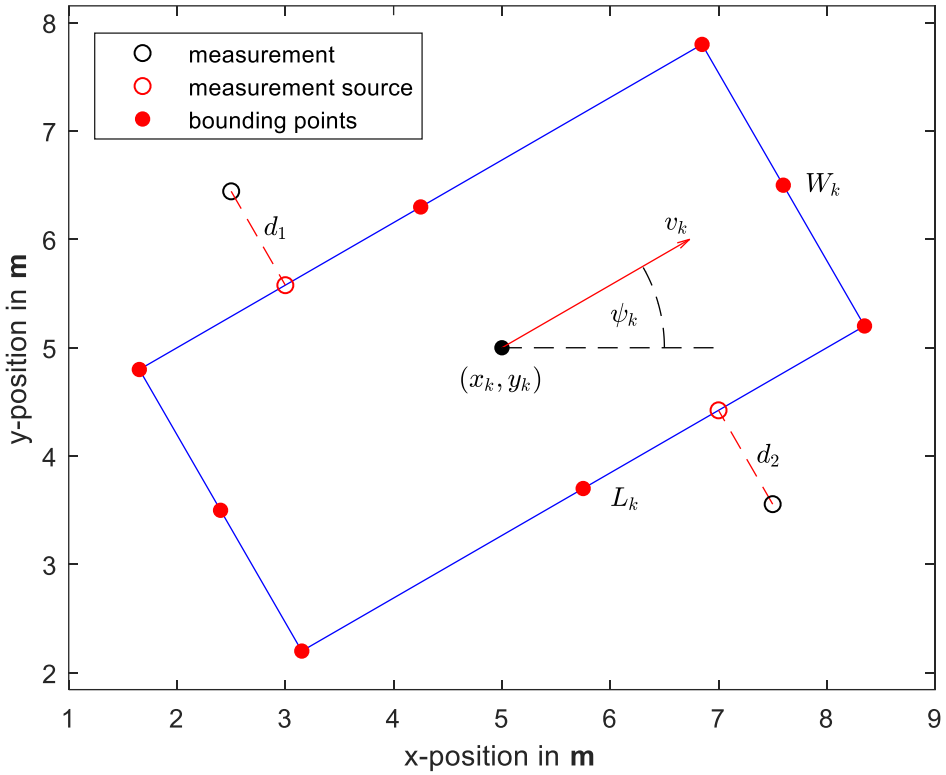

Figure 3.3: Measurement prediction for the rectangular shape estimator

Given a set of measurements and a predicted bounding box, the set of measurement sources can be computed. The aim of the filter is to adjust the bounding box parameters according to an optimal fit to the given measurement set. Therefore, the residuals

$$v_{k,i}^{(x)} = x_{k,i}^{(y)} - x_{k,i}^{(z)}, \tag{3.11}$$

$$v_{k,i}^{(y)} = y_{k,i}^{(y)} - y_{k,i}^{(z)} \tag{3.12}$$

need to be minimized with respect to the equation system

$$\begin{pmatrix} d_{k,i} \cdot \cos\left(\hat{\psi}_{k,i}\right) + x_{k,i}^{(z)} \\ d_{k,i} \cdot \sin\left(\hat{\psi}_{k,i}\right) + y_{k,i}^{(z)} \end{pmatrix} - \begin{pmatrix} \left(x_{k,i}^{(z)} + v_{k,i}^{(x)}\right) \\ \left(y_{k,i}^{(z)} + v_{k,i}^{(y)}\right) \end{pmatrix} = 0\,. \tag{3.13}$$

The calculation of the full covariance matrix P_k for the estimated system state $\boldsymbol{x}_k$, containing the variances as well as the corresponding correlation coefficients, is described in [22]. In the case of fast changes in the parameters of the bounding box the adjustment fails. This problem can be addressed by transforming the rectangle distribution to the measurement space using the bounding points illustrated in Figure 3.3. In the remainder of this thesis the rectangular

shape estimator is used for comparison with the investigated spline-based measurement model. To compare the rectangular shape estimator with the presented spline-based measurement model, an implementation of the rectangular shape estimator is provided by Stefan Wirtensohn.

4 Multi object tracking

This chapter describes a solution to the problem of tracking multiple objects in a cluttered environment. In classic approaches each track is assigned to a measurement in order to perform a single object tracker. The main challenge is therefore the measurement to track assignment and the track management in these approaches. The track management is needed since the disappearance or appearance of an object can occur in each time step. A solution to the multi object tracking problem investigated in recent years is an approach describing the problem in a multi object Bayes filter using so called random finite sets (RFS) and finite set statistics (FISST), leading to the probability hypothesis density (PHD) filter. The remainder of this chapter is organized as follows. In section 4.1 the problem of tracking multiple objects is described in detail. Also, the classic approaches solving this problem and their limitations leading to the necessity of a new approach are briefly addressed. In order to be able to understand the generalization of the single object Bayes filter to a multi object Bayes filter, the basic ideas of a RFS and FISST are discussed in section 4.2. The derivation of the multi object Bayes recursion using RFSs is then presented in section 4.3. Since the general multi object Bayes equation is computationally intractable in most scenarios, approximations are needed to propagate the posterior multi object distribution. Two famous approximations to the multi object Bayes recursion are the PHD filter and the Multi-Bernoulli filter. In section 4.4 the PHD framework is therefore described. Since the aim of this thesis is about to track multiple extended objects the modifications of the point object PHD filter to an extended object PHD filter are summarized in section 4.5.

4.1 Problem description

In a first consideration of the multi object tracking problem the objects are assumed to be point objects generating at most one measurement per time step. Using this assumption several problems need to be considered when tracking multiple objects in a cluttered environment.

- The number of objects being present in the surveillance area is unknown. Besides, it cannot be assumed that every object being present generates a measurement in every time step.
- The number of objects being present in the surveillance area can change within every time step since new objects can appear and present objects can disappear. Also, the spawning of one object to multiple objects is possible.
- The assignment of a measurement to a specific track is a very complex problem. Since the number of objects being present is unknown the measurement can occur from a real object or can be a clutter measurement. Assigning a measurement to a wrong track can lead to a filter divergence.

A visualization of the multi object tracking problem is shown in Figure 4.1. In the figure eight measurements originating from nine objects and five clutter measurements are shown. For the sensor the real measurements and clutter measurements are indistinguishable. Therefore, the measurement to track assignment is a crucial problem in multi object tracking. Since multi

object tracking is a problem studied for decades' various approaches exist to solve it. The most widely used classic approaches are the joint probabilistic data association (JPDA) filter and the multi hypothesis tracking (MHT) filter. Summaries to both filters and references to other approaches can be found in [23]. The basic idea to classic solutions of the multi object tracking problem is always to assign a measurement to a track and perform one of the single object trackers presented in chapter 2. Since clutter measurements can occur, objects can appear or disappear and misdetections are possible, the measurement to track assignment is the key element of a classic multi object tracker. The assignment approaches are briefly summarized below. For detailed information the reader is referred to the literature given below.

The simplest possible data association is a nearest neighbor approach where each predicted track is assigned to the nearest measurement to perform the update step of a single object Bayes filter.

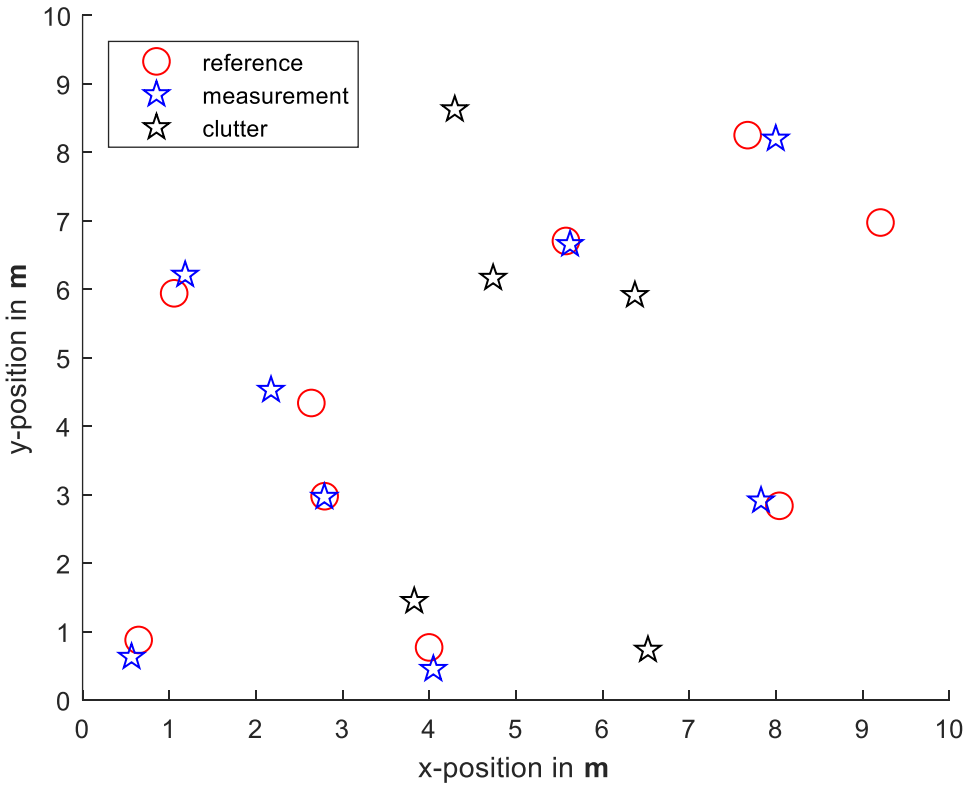

Figure 4.1: Multi object tracking problem visualization

Due to the possible unequal number of measurements and tracks, this approach can lead to a very poor performance since one measurement can be assigned to various tracks, causing the filter to diverge. An extension of the nearest neighbor approach is the global nearest neighbor tracker where each measurement is assigned to at most one track. Using a cost function like the total distance, where each track is assigned to a unique measurement, the goal is to minimize the cost function forming a joint data association. However, the global nearest neighbor approach is also susceptible to track loss leading to poor filter performances. Therefore, a data association approach for tracking a single object in a cluttered environment can be extended to track multiple objects. The probabilistic data association (PDA) filter [2, pp. 111-119] [11, pp. 163-184] uses all validated measurements per time step to update the prior density of the track using a weighting assumption to each measurement. The joint PDA (JPDA) filter [2, pp. 205-221] [11, pp. 222-238] is the extension to track a known number of

objects in a cluttered environment. Since the number of objects actually being present can be unknown in many scenarios, the JPDA filter can be extended to the joint integrated PDA filter [24], in order to track an unknown number of objects in a cluttered environment. When tracking closely spaced objects, the JPDA filter reaches its limits leading to the necessity of an approach considering all possible progressions of the track.

Given a set of tracks and a set of measurements in one time step, the multi hypothesis tracking (MHT) [23] filter scores every association hypothesis and performs a single object Bayes update for every probable association hypothesis. In the next time step the new set of measurements can be used to reassess every association hypothesis and delete the unlikely ones. This procedure automatically handles the appearance and disappearance of a single object, as one association hypothesis for every measurement is always a non-existing track. Also, a non-detection hypothesis is always considered. In order to reduce the number of hypotheses, merging and pruning is performed within every time step. Otherwise the number of hypotheses would increase exponentially. In a pruning step all hypotheses with a weighting score lower than a specific threshold are rejected, while a merging step is used to melt closely spaced hypotheses. However, the number of hypotheses increases very fast, although merging and pruning is performed causing the filter to become very slow.

The novel approach investigated in the last one and a half decades was first introduced by Mahler [25] using RFSs and FISST methods to describe a closed multi object Bayesian filter approach. An approximation of the multi object Bayes equation is used in this thesis to track several extended objects. Using the RFS approach, comparable results to MHT filters with a lower computational effort can be achieved. In order to be able to understand the multi object Bayesian filter approach, the next section is about to recap the theory of RFSs and FISST methods.

4.2 Random finite sets

When tracking multiple objects in a cluttered environment the number of tracks as well as the number of measurements are random. The set of tracks and measurements therefore need to be modeled random in the number of elements and the elements itself. Such a tool is provided by the theory of RFSs and FISST [26, pp. 343-394] [4, pp. 19-30] [3, pp. 61-64] [27]. A RFS $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ represents an unordered set of elements drawn of the space $\mathcal{X}$ with $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathcal{X}$. In general, the space $\mathcal{X}$ can be any measurable space, however, this thesis is restricted to the space being an Euclidean space with $\mathcal{X} = \mathbb{R}^n$. Using this representation, the multi object state can be stated as

$$\boldsymbol{X}_k = \left\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{r_k}\right\}, \boldsymbol{x}_1, \dots, \boldsymbol{x}_{r_k} \in \mathbb{R}^n \tag{4.1}$$

with a varying number of tracks r_k possibly changing every time step and the set elements drawn from the state space $\mathbb{R}^n$. Analog to this, the measurement set can be stated as

$$\boldsymbol{Z}_k = \left\{\boldsymbol{z}_1, \dots, \boldsymbol{z}_{l_k}\right\}, \boldsymbol{z}_1, \dots, \boldsymbol{z}_{l_k} \in \mathbb{R}^m \tag{4.2}$$

with a varying number of measurements l_k possibly changing every time step and the set elements drawn out of the measurement space $\mathbb{R}^m$. The number of elements in a RFS is denoted as the cardinality $|\boldsymbol{X}|$ and modeled using a discrete distribution $\rho(n) = P\{|\boldsymbol{X}| = n\}$ to clarify the varying number of tracks and measurements in every time step. Furthermore, the elements of the RFS itself need to be modeled using a symmetric joint distribution $p_n(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$, to represent the random draw of elements out of the Euclidean space. A

symmetric joint distribution delivers the same value if the order of arguments is swapped. The distribution therefore is equal for every permutation of the input values. In summary a RFS draws its instantiations out of the set of all finite subsets of $\mathbb{R}^n$ denoted as $\mathcal{F}(\mathbb{R}^n)$ as $\boldsymbol{X} \in \mathcal{F}(\mathbb{R}^n)$. By specifying the discrete distribution $\rho(n)$, modeling the cardinality, and the symmetric joint distribution p_n, modeling the drawing of the set elements, a RFS is well-defined. In order to use a RFS in a multi object Bayesian filter, the usual probabilistic descriptors like the probability density function (PDF) and its moments need to be defined for a RFS. In Mahler [26, pp. 343-394], these descriptors are summarized under the term FISST. A FISST PDF is then defined as

$$f(\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}) := n! \cdot \rho(n) \cdot p_n(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) \tag{4.3}$$

with the FISST PDF denoted as $f(\boldsymbol{X})$. In order to verify the well-definition of the FISST PDF it needs to integrate to one. Since a RFS is a set and therefore the FISST PDF is a function defined on sets it must be integrated using a set integral defined as

$$\int f(\boldsymbol{X})\, \delta\boldsymbol{X} := f(\emptyset) + \sum_{n=1}^{\infty} \frac{1}{n!} \int f(\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}) d\boldsymbol{x}_1 \dots d\boldsymbol{x}_n \,. \tag{4.4}$$

By inserting (4.3) in (4.4), the set integral can be stated as $\int f(\boldsymbol{X})\delta\boldsymbol{X} = f(\emptyset) + \sum_{n=1}^{\infty} \frac{n!}{n!} \cdot \rho(n) \cdot \int p_n(\boldsymbol{x}_1 \dots \boldsymbol{x}_n) d\boldsymbol{x}_1 \dots d\boldsymbol{x}_n$. The FISST PDF is equal to zero inserting the empty set and the joint distribution p_n integrates to one since it is a well-defined PDF. The set integral therefore simplifies to $\int f(\boldsymbol{X})\delta\boldsymbol{X} = \sum_{n=1}^{\infty} \rho(n) = 1$, which is equal to one since $\rho(n)$ is a well-defined discrete PDF that sums up to one for all $n \in \mathbb{N}$. This calculation proves $f(\boldsymbol{X})$ to be a well-defined PDF.

Another tool needed for the derivation of a tractable multi object Bayes filter is the first order moment of a RFS [4, p. 23]. As a reminder, the first order moment of a random variable is the expectation value. Given a RFS $\boldsymbol{X} \subseteq \mathbb{R}^n$, its first order moment is defined as

$$V(\boldsymbol{B}) := \mathbb{E}(|\boldsymbol{X} \cap \boldsymbol{B}|) = \int_{\boldsymbol{B}} v(\boldsymbol{x}) dx \tag{4.5}$$

for any $\boldsymbol{B} \subseteq \mathbb{R}^n$. In this definition $v(\boldsymbol{x})$ is called the intensity function. The first order moment over a set $\boldsymbol{B}$ can be interpreted as the expected number of elements of $\boldsymbol{X}$ that are elements of $\boldsymbol{B}$ as well. In the remainder of this thesis the first order moment will be used as an approximation of the posterior density within a multi object Bayes filter, propagating only the intensity function over time, comparable to the expectation value within a Kalman filter propagating over time.

Following, two RFSs needed in this thesis are summarized [27]. Therefore, only the cardinality distribution $\rho(n)$ needs to be specified. The first RFS needed is one to model whether the set involves an element or not, called Bernoulli RFS. The cardinality distribution accordingly is represented by a Bernoulli distribution. Thus, the Bernoulli RFS involves an element with probability q and is empty with probability $1 - q$. The Bernoulli RFS is then given as

$$f(\boldsymbol{X}) = \begin{cases} 1-q & if\ \boldsymbol{X} = \emptyset \\ q \cdot p(x) & if\ \boldsymbol{X} = \{\boldsymbol{x}\} \end{cases}. \tag{4.6}$$

Other important RFSs, considering a multi object Bayes equation, are independent identically distributed (IID) cluster RFSs. Within this special case each element of the RFS $\boldsymbol{x} \in \boldsymbol{X}$ is independent but identically distributed according to the PDF $p(\boldsymbol{x})$. Using this property, the joint probability of (4.3) can be rewritten using the product over the identically distributed probability of each element within the RFS. The FISST PDF of an IID cluster RFS is then given as

$$f(\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}) = n! \cdot \rho(n) \cdot \prod_{\boldsymbol{x} \in \boldsymbol{X}} p(\boldsymbol{x}) . \tag{4.7}$$

The set of measurements as well as the set of objects are modeled using IID cluster RFSs, since the measurements are assumed to be generated independently of each other and the objects are assumed to evolve independently over time. A special case of an IID cluster RFS is a Poisson RFS, where the cardinality distribution $\rho(n)$ is assumed to be Poisson distributed. The cardinality distribution of a Poisson RFS is then given as

$$\rho(n) = \frac{e^{-\lambda} \cdot \lambda^n}{n!}. \tag{4.8}$$

The Poisson FISST PDF is then given by inserting (4.8) in (4.7) and can be stated as

$$f(\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}) = n! \cdot \frac{e^{-\lambda} \cdot \lambda^n}{n!} \cdot \prod_{\boldsymbol{x} \in \boldsymbol{X}} p(\boldsymbol{x}) = e^{-\lambda} \cdot \prod_{\boldsymbol{x} \in \boldsymbol{X}} \lambda \cdot p(\boldsymbol{x}) . \tag{4.9}$$

The theory elaborated in this section can now be used to derive the multi object Bayes recursion in the next section.

4.3 The multi object Bayes filter

By modeling the set of tracks and the set of measurements as RFSs, a multi object Bayes filter [26, pp. 483-539] [3, pp. 62-65] [4, pp. 30-40] can now be formulated using the derivation of section 2.1 resulting in equations (2.3)-(2.5). However, first of all the measurement RFS and the multi object RFS need to be specified [4, pp. 30-35].

Assuming the multi object state $\boldsymbol{X}_{k-1}$ from the previous time step is given, then each object $\boldsymbol{x}_{k-1} \in \boldsymbol{X}_{k-1}$ within the set either survives and moves on or disappears. For the sake of simplicity, the survival probability p_S is assumed to be constant in this thesis but can also be formulated more generally. Accordingly, each object either survives with probability p_S or disappears with probability $1 - p_S$. The behavior of one object can therefore be modeled as a Bernoulli RFS $\boldsymbol{S}_{k|k-1}(\{\boldsymbol{x}_{k-1}\})$, using the state transition $f(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$. Also, the state transition is assumed to be constant over time, but can be formulated more generally. Using (4.6), the single object Bernoulli RFS is then given as

$$\boldsymbol{S}_{k|k-1}(\{\boldsymbol{x}_{k-1}\}) = \begin{cases} 1-p_S & if\ \boldsymbol{x}_{k-1} = \emptyset \\ p_S \cdot f(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) & if\ \boldsymbol{x}_{k-1} \neq \emptyset \end{cases}. \tag{4.10}$$

Since $\boldsymbol{X}_{k-1}$ contains multiple objects the appearance or disappearance process needs to be modeled using the union of all Bernoulli RFSs as specified in (4.10). This union is then called a

multi-Bernoulli RFS containing a set of cardinality distributions and state transitions. More details on a multi-Bernoulli RFS can be found in [4, pp. 29-30] [26, p. 368]. The multi-Bernoulli RFS is then given as

$$\boldsymbol{T}_{k|k-1}(\boldsymbol{X}_{k-1}) = \bigcup_{\boldsymbol{x}_{k-1} \in \boldsymbol{X}_{k-1}} \boldsymbol{S}_{k|k-1}(\{\boldsymbol{x}_{k-1}\}) \,. \tag{4.11}$$

Another component of the multi object state at time k are objects of spontaneous appearances, called object births. Object spawning is also possible but will be neglected in this thesis. According to (4.8) and (4.9), the RFS of spontaneous births is modeled using a Poisson RFS called $\boldsymbol{\Gamma}_k$. The multi object state RFS can then be specified as union of the transition RFS in (4.11) and the birth RFS and is given as

$$\boldsymbol{X}_k = \boldsymbol{T}_{k|k-1}(\boldsymbol{X}_{k-1}) \cup \boldsymbol{\Gamma}_k \,. \tag{4.12}$$

The next step is to specify the measurement RFS, which consists of the measurements generated from an object of interest and clutter measurements. The clutter measurements $\boldsymbol{K}_k$ can be modeled by a Poisson RFS using (4.8) and (4.9). The remaining measurements in the measurement set are those originating from a moving object. Those objects can be detected or stay undetected according to a detection probability at time k. The detection probability p_D is assumed to be constant in this thesis but can also be formulated more generally. Assuming each object generates a measurement with probability p_D and stays undetected with probability $1 - p_D$, the detection of a single object can be modeled using a single object Bernoulli RFS. The measurement likelihood $h(\boldsymbol{z}_k|\boldsymbol{x}_k)$ is assumed to be constant over time but can also be formulated more generally. Using (4.6), the single object detection RFS $\boldsymbol{D}_k(\{\boldsymbol{x}_k\})$ is given as

$$D_k(\{\boldsymbol{x}_k\}) = \begin{cases} 1 - p_D & if\ \boldsymbol{x}_k = \emptyset \\ p_D \cdot h(\boldsymbol{z}_k|\boldsymbol{x}_k) & if\ \boldsymbol{x}_k \neq \emptyset \end{cases} . \tag{4.13}$$

Since the measurement set contains measurements of multiple objects, the measurement process needs to be modeled using the multi-Bernoulli RFS as union of all single Bernoulli RFSs as specified in (4.13) and is then given as

$$\boldsymbol{\Theta}(\boldsymbol{X}_k) = \bigcup_{\boldsymbol{x}_k \in \boldsymbol{X}_k} \boldsymbol{D}_k(x_k) \,. \tag{4.14}$$

Using the RFS of clutter and the RFS modeling the measurement process, the complete measurement set can be stated as the union of those RFSs and is given as

$$\boldsymbol{Z}_k = \boldsymbol{\Theta}(\boldsymbol{X}_k) \cup \boldsymbol{K}_k \,. \tag{4.15}$$

Now the multi object state RFS and the measurement RFS are completely specified and can be used to form a multi object Bayes equation. The remaining missing components to formulate a multi object Bayes filter are the multi object transition density and the multi object measurement likelihood. Both those functions can be specified using the convolution formula [26, p. 385] for the union of independent RFSs as given in (4.12) and (4.15). Starting with the multi object transition density $f(\boldsymbol{X}_k|\boldsymbol{X}_{k-1})$, the probability densities $p_{\boldsymbol{T}_{k|k-1}}$ and $p_{\boldsymbol{\Gamma}_k}$

are those of the transition RFS $\boldsymbol{T}_{k|k-1}(\boldsymbol{X}_{k-1})$ and the birth RFS $\boldsymbol{\Gamma}_k$ respectively. The multi object transition density is then given as

$$f(\boldsymbol{X}_k|\boldsymbol{X}_{k-1}) = \sum_{\boldsymbol{W}\subseteq\boldsymbol{X}_k} p_{\boldsymbol{T}_{k|k-1}}(\boldsymbol{W}|\boldsymbol{X}_{k-1}) \cdot p_{\boldsymbol{\Gamma}_k}(\boldsymbol{X}_k\backslash\boldsymbol{W}) \,. \tag{4.16}$$

Since the multi object state RFS is an unordered set, it is not clear which states within the RFS can be assigned to the RFS of already existing objects from the previous time step, and which objects belong to the RFS of spontaneous births. Therefore, the summation of (4.16) considers every subset $\boldsymbol{W}$ of $\boldsymbol{X}_k$ to be the RFS of existing objects from the previous time step and the remaining elements, represented as the set difference of $\boldsymbol{X}_k$ and the subset of $\boldsymbol{X}_k$, belong to the RFS of spontaneous births. The last component to be specified is the multi object measurement likelihood $h(\boldsymbol{Z}_k|\boldsymbol{X}_k)$, which can be represent using the probability densities $p_{\boldsymbol{\Theta}_k}$ and $p_{\boldsymbol{K}_k}$ of the measurement process RFS $\boldsymbol{\Theta}(\boldsymbol{X}_k)$ and the clutter RFS $\boldsymbol{K}_k$ respectively within the convolution formula. The multi object measurement likelihood is then given as

$$h(\boldsymbol{Z}_k|\boldsymbol{X}_k) = \sum_{\boldsymbol{W}\subseteq\boldsymbol{Z}_k} p_{\boldsymbol{\Theta}_k}(\boldsymbol{W}|\boldsymbol{X}_k) \cdot p_{\boldsymbol{K}_k}(\boldsymbol{Z}_k\backslash\boldsymbol{W}) \tag{4.17}$$

Using the multi object measurement likelihood and the multi object transition density, the multi object Bayes recursion can now be specified. Denote $\boldsymbol{Z}^{k-1}$ the set of measurement RFSs up to time $k-1$ and $p(\boldsymbol{X}_k|\boldsymbol{Z}^{k-1})$ and $p(\boldsymbol{X}_k|\boldsymbol{Z}^k)$ the multi object predicted density and the multi object posterior density respectively. The multi object Bayes recursion can then be subdivided in the following steps [4, pp. 35-37]:

1. Multi object prediction step:

$$p(\boldsymbol{X}_k|\boldsymbol{Z}^{k-1}) = \int f(\boldsymbol{X}_k|\boldsymbol{X}_{k-1}) \cdot p(\boldsymbol{X}_{k-1}|\boldsymbol{Z}^{k-1})\, \delta\boldsymbol{X}_{k-1} \,. \tag{4.18}$$

2. Multi object update step:

$$p(\boldsymbol{X}_k|\boldsymbol{Z}^k) = \frac{1}{\eta} \cdot h(\boldsymbol{Z}_k|\boldsymbol{X}_k) \cdot p(\boldsymbol{X}_k|\boldsymbol{Z}^{k-1}) \,. \tag{4.19}$$

3. Normalization:

$$\eta = \int h(\boldsymbol{Z}_k|\boldsymbol{X}_k) \cdot\ p(\boldsymbol{X}_k|\boldsymbol{Z}^{k-1})\, \delta\boldsymbol{X}_k \,. \tag{4.20}$$

4.4 The probability hypothesis density filter

The derivation of the multi object Bayes recursion, presented in the previous section, provides the best possible estimate of the multi object state from a Bayesian point of view. However, the set integrals within those equations operate on the space of all finite subsets of the single object space $\mathcal{F}(\mathbb{R}^n)$ what makes them computationally intractable to propagate the multi object state over time. A reasonable approximation to the multi object state is the intensity, presented in (4.5), as a first order moment of the multi object state. In the context of object tracking the name probability hypothesis density (PHD) has prevailed for the first order moment of the multi object probability density function. The idea of the PHD filter is therefore to propagate the PHD over time, instead of the multi object probability density function. This

approximation makes the filter a computationally tractable algorithm since the PHD operates on the single object space $\mathbb{R}^n$. In order to derive the equations of the PHD filter [28] [26, pp. 565-632] the following assumptions need to be considered:

- Independence assumption: It is assumed that each object of interest within the surveillance area operates independent to all the other objects regarding the motion and the generation of measurements.
- Clutter assumption: The clutter measurement set is assumed to be a Poisson RFS. Furthermore, the clutter measurements are assumed to be independent of object generated measurements.
- Prediction assumption: The predicted multi object state RFS is assumed to be a Poisson RFS.

Using these assumptions, the prediction of the PHD as an approximation of the predicted multi object probability density of (4.18), avoiding spawned objects, can be stated as [29]

$$v_{k|k-1}(\boldsymbol{x}) = \int p_S \cdot f(\boldsymbol{x}|\boldsymbol{\zeta}) \cdot v_{k-1|k-1}(\boldsymbol{\zeta})\, d\boldsymbol{\zeta} + \gamma_k(\boldsymbol{x}) \tag{4.21}$$

with $\gamma_k(\boldsymbol{x})$ as the intensity of the birth RFS $\boldsymbol{\Gamma}_k$ and $\boldsymbol{\zeta}$ as the previous system state at time k. Given the predicted intensity, the update step of the PHD filter as an approximation of the posterior multi object probability density of (4.19) and (4.20) is given as

$$v_{k|k}(\boldsymbol{x}) = v_{k|k-1}(\boldsymbol{x}) \cdot (1 - p_D) + \sum_{\boldsymbol{z} \in \boldsymbol{Z}_k} \frac{p_D \cdot h(\boldsymbol{z}|\boldsymbol{x}) \cdot v_{k|k-1}(\boldsymbol{x})}{\kappa_k(\boldsymbol{z}) + \int p_D \cdot h(\boldsymbol{z}|\boldsymbol{\zeta}) \cdot v_{k|k-1}(\boldsymbol{\zeta})\, d\boldsymbol{\zeta}} \tag{4.22}$$

with $\kappa_k(\boldsymbol{z})$ as the intensity of the clutter RFS $\boldsymbol{K}_k$ given as $\kappa_k(\boldsymbol{z}) = \lambda_k c(\boldsymbol{z})$. The rate of the Poisson distributed random number modeling the cardinality of the clutter RFS is λ_k, while the spatial distribution of a clutter measurement over the surveillance area is $c(\boldsymbol{z})$. Within the sum of (4.22) every single object state is compared to every measurement. However, the recursion specified in (4.21) and (4.22) does not provide a closed form solution that can be implemented. One possibility is to represent the intensity function using sequential Monte Carlo methods [30]. The maxima of the intensity function, representing the most likely positions of the objects, are depicted as the points with the largest number of particles after the resampling step in this approach. A closed form solution of the PHD recursion is provided by approximating the PHD using GMs [29]. A GM is a weighted sum of normal distributions. As the PHD filter is applied to a nonlinear measurement model in the further course of this thesis, the GM approximation of the PHD recursion is presented using nonlinear motion and measurement models. For the derivation further assumptions need to be considered:

- Modeling assumption: The motion and measurement models are assumed to be nonlinear models as given in (2.6) and (2.7).
- Birth RFS assumption: The birth RFS is assumed to be a GM of the form

$$b_k(\boldsymbol{x}) = \sum_{j=1}^{J_{b,k}} \omega_{b,k}^{(j)} \cdot \mathcal{N}\left(\boldsymbol{x}; \hat{\boldsymbol{x}}_{b,k}^{(j)}, P_{b,k}^{(j)}\right) \tag{4.23}$$

where $J_{b,k}$ is the number of GMs, $\omega_{b,k}$ is the weight of a GM component and $\hat{\boldsymbol{x}}_{b,k}$ and $P_{b,k}$ are the mean and covariance respectively of the GM component.

In order to specify the prior and posterior intensities at time k, the posterior intensity at time $k-1$ is assumed to be a GM of the form

$$v_{k-1|k-1}(\boldsymbol{x}) = \sum_{j=1}^{J_{k-1|k-1}} \omega_{k-1|k-1}^{(j)} \cdot \mathcal{N}\left(\boldsymbol{x}; \hat{\boldsymbol{x}}_{k-1|k-1}^{(j)}, P_{k-1|k-1}^{(j)}\right) \tag{4.24}$$

with $J_{k-1|k-1}$ as the amount of GM components at $k-1$, $\omega_{k-1|k-1}^{(j)}$ as the weight of a GM component, and $\hat{\boldsymbol{x}}_{k-1|k-1}^{(j)}$ and $P_{k-1|k-1}^{(j)}$ as the mean and covariance respectively of the GM component at $k-1$. The prior intensity is given as

$$v_{k|k-1}(\boldsymbol{x}) = v_{S,k|k-1}(\boldsymbol{x}) + b_k(\boldsymbol{x}) \tag{4.25}$$

with $v_{S,k|k-1}(\boldsymbol{x})$ as the prediction of the GM components of $k-1$. While $b_k(\boldsymbol{x})$ is given using (4.23), the prediction of the intensity from the previous time step is given as

$$v_{S,k|k-1}(\boldsymbol{x}) = p_S \cdot \sum_{j=1}^{J_{k-1|k-1}} \omega_{k-1|k-1}^{(j)} \cdot \mathcal{N}\left(\boldsymbol{x}; \hat{\boldsymbol{x}}_{S,k|k-1}^{(j)}, P_{S,k|k-1}^{(j)}\right). \tag{4.26}$$

As the motion model is assumed to be a nonlinear model, the mean and covariance of the GM components of (4.26) need to be calculated using the extended transform presented in section 2.4. According to (2.20) – (2.22) the mean and covariance are given as

$$\hat{\boldsymbol{x}}_{S,k|k-1}^{(j)} = f\left(\hat{\boldsymbol{x}}_{k-1|k-1}^{(j)}\right) \tag{4.27}$$

and

$$P_{S,k|k-1}^{(j)} = F_k^{(j)} \cdot P_{k-1|k-1}^{(j)} \cdot {F_k^{(j)}}^T + Q_k \tag{4.28}$$

respectively, where $F_k^{(j)}$ denotes the Jacobian matrix evaluated at the jth posterior mean of the previous time step. The Jacobian matrix is given as

$$F_k^{(j)} = \nabla_{\boldsymbol{x}^T} f(\boldsymbol{x})|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k-1|k-1}^{(j)}}. \tag{4.29}$$

By multiplying the predicted intensity $v_{S,k|k-1}(\boldsymbol{x})$ with the probability of survival p_S, the death of an object is taken into account. The predicted intensity $v_{k|k-1}(\boldsymbol{x})$ is constructed as the sum of the intensity of the objects that survived, with the intensity of spontaneous births. Given the predicted intensity the posterior intensity can be calculated. As the predicted intensity is a GM, the posterior intensity is also a GM and can be specified as

$$v_{k|k}(\boldsymbol{x}) = (1 - p_D) \cdot v_{k|k-1}(\boldsymbol{x}) + \sum_{\boldsymbol{z} \in \boldsymbol{Z}_k} v_{D,k}(\boldsymbol{x}, \boldsymbol{z}) \,. \tag{4.30}$$

The first summand of the posterior intensity considers that none of the objects was detected, by multiplying the predicted intensity with the probability of non-detection. The second summand updates the whole predicted intensity with every measurement within the measurement RFS, assuming the objects were detected. As the measurement model is assumed to be a nonlinear model, (2.7) must be taken into account when predicting the measurements. The update equation of the intensity component presenting the detected objects is given as

$$v_{D,k}(\boldsymbol{x}, \boldsymbol{z}) = \sum_{j=1}^{J_{k|k-1}} \omega_{k|k}^{(j)}(\boldsymbol{z}) \cdot \mathcal{N}\left(\boldsymbol{x}; \hat{\boldsymbol{x}}_{k|k}^{(j)}(\boldsymbol{z}), P_{k|k}^{(j)}\right) \tag{4.31}$$

where the number of GM components is given as $J_{k|k-1} = J_{k-1|k-1} + J_{\gamma,k}$. Due to the nonlinearity of the measurement model, the extended transform is used to update the GM components of (4.31). Therefore, the measurement matrix is given as the Jacobian matrix evaluated at the jth predicted mean and is given as

$$H_k^{(j)} = \nabla_{\boldsymbol{x}^T} h(\boldsymbol{x})|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k|k-1}^{(j)}} \,. \tag{4.32}$$

Using the measurement matrix and the nonlinear measurement model, the update equations of the mean and the covariance matrix can be specified according to (2.24), (2.25) and (2.17) – (2.19) and are given as

$$\begin{aligned} \hat{\boldsymbol{x}}_{k|k}^{(j)}(\boldsymbol{z}) &= \hat{\boldsymbol{x}}_{k|k-1}^{(j)} + K_k^{(j)} \cdot \left(\boldsymbol{z} - h\left(\hat{\boldsymbol{x}}_{k|k-1}^{(j)}\right)\right), \\ P_{k|k}^{(j)} &= P_{k|k-1}^{(j)} - K_k^{(j)} \cdot H_k^{(j)} \cdot P_{k|k-1}^{(j)} \end{aligned} \tag{4.33}$$

respectively. The Kalman gain is computed using

$$K_k^{(j)} = P_{k|k-1}^{(j)} \cdot H_k^{(j)} \cdot \left(H_k^{(j)} \cdot P_{k|k-1}^{(j)} \cdot {H_k^{(j)}}^T + R_k\right)^{-1} . \tag{4.34}$$

In (4.31) each GM component is updated using the same measurement. The probability that the measurement actually belongs to the GM component is expressed in the weight of the GM component. By definition those weights need to sum up to one for any $v_{D,k}(\boldsymbol{x}, \boldsymbol{z})$. The weights are calculated using

$$\omega_{k|k}^{(j)}(\boldsymbol{z}) = \frac{p_D \cdot \omega_{k|k-1}^{(j)} \cdot q_k^{(j)}(\boldsymbol{z})}{\kappa_k(\boldsymbol{z}) + p_D \cdot \sum_{l=1}^{J_{k|k-1}} \omega_{k|k-1}^{(l)} q_k^{(l)}(\boldsymbol{z})} \tag{4.35}$$

where the denominator is responsible for the normalization of the weights. The factor $q_k^{(j)}(\boldsymbol{z})$ is given as the multivariate normal distribution

$$q_k^{(j)}(\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{z}; h\left(\hat{\boldsymbol{x}}_{k|k-1}^{(j)}\right), H_k^{(j)} \cdot P_{k|k-1}^{(j)} \cdot {H_k^{(j)}}^T + R_k\right). \tag{4.36}$$

The value of the normal distribution gets low if the predicted measurement is far away from the given measurement. On the other hand, the value of the normal distribution gets high if the predicted measurement is close to the given measurement. The presented equations (4.23) - (4.36) specify the GM-PHD filter proposed in [29]. A generalization of the PHD filter is the cardinalized PHD (CPHD) filter [31] [32]. This extension propagates not only the PHD through time but also the cardinality distribution. By jointly propagating the posterior PHD and cardinality distribution, the accuracy and stability of the filter is improved and the temporal variance in the estimated cardinality is reduced.

4.5 The extended object PHD filter

With the increasing sensor accuracy and resolution, the small object assumption modeling the objects as points is not reasonable anymore. If an object is sufficiently close to the sensor it gives rise to more than one measurement per time step and is therefore called extended object in the context of object tracking. The sufficient proximity of the object to the sensor depends on the resolution of the sensor and the objects extension. In this section the extension of the PHD filter for point objects to the PHD filter for extended objects is addressed. To achieve this extension, the update equation of the PHD filter according to (4.22) needs to be adapted. Equation (4.21), handling the prediction step, remains unchanged. The difference of both equations is in the handling of measurements originating from the same object. Assuming to measure extended objects, the number of measurements originating from the same object depends on the distance between the object and sensor, the extension, orientation and material composition of the object, as well as unpredictable processes like misdetections. Furthermore, the assignment of a measurement to a specific object is unknown as well. These problems lead to the necessity to consider every possible partitioning of the measurement set, referred to as partition p in the following, in the update equation. Accordingly, every partition p divides the measurement set into subsets, referred to as cells $\boldsymbol{W}$ in the following, containing measurements possibly originating from the same object. With a given predicted PHD the update PHD for extended objects [33] can be calculated using

$$v_{k|k}(\boldsymbol{x}) = \left(1 - \left(1 - e^{-\gamma(\boldsymbol{x})}\right) p_D + e^{-\gamma(\boldsymbol{x})} p_D \sum_{p \angle \boldsymbol{Z}_k} \omega_p \sum_{\boldsymbol{W} \in p} \frac{\gamma(\boldsymbol{x})^{|\boldsymbol{W}|}}{d_{\boldsymbol{W}}} \Phi_{\boldsymbol{W}}(\boldsymbol{x}) \prod_{\boldsymbol{z}_k \in \boldsymbol{W}} \frac{1}{\kappa_k(\boldsymbol{z}_k)}\right) \cdot v_{k|k-1}(\boldsymbol{x}). \tag{4.37}$$

The number of measurements generated from an object is modeled as Poisson distributed random number with rate $\gamma(\boldsymbol{x})$. In some approaches this rate is modeled to be constant but in general the number of generated measurements depends on the object's state. According to (4.8), the probability of an object not generating a single measurement is given as $e^{-\gamma(\boldsymbol{x})}$. The complementary event of an object generating at least one measurement can then be calculated using $1 - e^{-\gamma(\boldsymbol{x})}$. Finally, the effective probability of detection $\left(1 - e^{-\gamma(\boldsymbol{x})}\right) p_D$ is given as the probability of an object generating at least one measurement multiplied with the probability of detection. Thus, the first summand of (4.37) is handling the undetected objects. The second summand considers each partition of the measurement set $\boldsymbol{Z}_k$ by taking the sum over all possible partitions $p \angle \boldsymbol{Z}_k$. Subsequently, the next sum considers each cell $\boldsymbol{W}$ in the

partition p. The normalization factors for each partition and cell are denoted as ω_p and $d_{\boldsymbol{W}}$ respectively. The extended object measurement likelihood $\Phi_{\boldsymbol{W}}(\boldsymbol{x})$ considers each cell in each partition for the calculation of the updated PHD.

The partitioning of the measurement set is done by considering every possible combination of subsets, excluding the null set, to form the measurement set. Assume $\boldsymbol{Z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3\}$ to be the measurement set in a specific time step. The possible partitions are given as [33]

$$\begin{aligned} p_1 = \big\{\{\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3\}\big\}, p_2 = \big\{\{\boldsymbol{z}_1\}, \{\boldsymbol{z}_2\}, \{\boldsymbol{z}_3\}\big\}, p_3 = \big\{\{\boldsymbol{z}_1, \boldsymbol{z}_2\}, \{\boldsymbol{z}_3\}\big\}, \\ p_4 = \big\{\{\boldsymbol{z}_1, \boldsymbol{z}_3\}, \{\boldsymbol{z}_2\}\big\}, p_5 = \big\{\{\boldsymbol{z}_2, \boldsymbol{z}_3\}, \{\boldsymbol{z}_1\}\big\}. \end{aligned} \tag{4.38}$$

With three elements in the measurement set all possible partitions can be considered in an update step. However, the number of possible partitions already grows up to 115975 with ten elements in the measurement set, leading to the impossibility to consider every partition in a tractable tracking algorithm. The number of partitions can be computed using the bell numbers [34]. To only consider a reasonable subset of all possible partitions in a tractable tracking algorithm, cluster algorithms can be used, as the measurements originating from the same object are spatially related.

As with the PHD filter for point objects, the PHD filter for extended objects does not provide a closed form solution. Therefore, a GM approximation of the PHD for extended objects [35] is presented as closed form solution that can be implemented. In addition to the assumptions presented in section 4.4 the following assumption is adopted in [35]:

- Object generated measurements assumption: The rate of the Poisson distributed random number modeling the number of object-generated measurements is given as

$$\gamma(\boldsymbol{x}) = \gamma\left(\widehat{\boldsymbol{x}}_{k|k-1}^{(i)}\right) \tag{4.39}$$

  for all $i = 1, \dots, J_{k|k-1}$. The rate is assumed to be dependent on the predicted object state.

The prediction step of the GM-PHD filter for extended objects is provided using the prediction step of the GM-PHD filter for point objects according to (4.24)-(4.29). The update step calculating the posterior extended object PHD is given as

$$v_{k|k}(\boldsymbol{x}) = v_{k|k}^{ND}(\boldsymbol{x}) + \sum_{p \angle \boldsymbol{Z}_k} \sum_{\boldsymbol{W} \in p} v_{k|k}^{D}(\boldsymbol{x}, \boldsymbol{W}) \tag{4.40}$$

with $v_{k|k}^{ND}(\boldsymbol{x})$ the GM components handling the undetected objects and $v_{k|k}^{D}(\boldsymbol{x}, \boldsymbol{W})$ the GM components handling the detected objects, assuming $\boldsymbol{W}$ is the cell of measurements belonging to the object. The GM components handling the undetected objects can be calculated using

$$v_{k|k}^{ND}(\boldsymbol{x}) = \sum_{j=1}^{J_{k|k-1}} \omega_{k|k}^{(j)} \cdot \mathcal{N}\left(\boldsymbol{x}; \widehat{\boldsymbol{x}}_{k|k}^{(j)}, P_{k|k}^{(j)}\right) \tag{4.41}$$

where the updated mean $\widehat{\boldsymbol{x}}_{k|k}^{(j)}$ and covariance $P_{k|k}^{(j)}$ are given as the predicted ones $\widehat{\boldsymbol{x}}_{k|k-1}^{(j)}$ and

$P_{k|k-1}^{(j)}$ respectively. The weight of the GM component handling the undetected objects is calculated as

$$\omega_{k|k}^{(j)} = \left(1 - \left(1 - e^{-\gamma\left(\hat{\boldsymbol{x}}_{k|k-1}^{(j)}\right)}\right) p_D\right) \omega_{k|k-1}^{(j)} \tag{4.42}$$

using the effective probability of detection and the predicted weights. As for the update equations of the PHD filter for point objects, the measurement model is assumed to be a nonlinear model. Therefore, the extended transform presented in section 2.4 is used in the update equations for the extended object PHD filter as well. The GM handling the detected objects is given as

$$v_{k|k}^{D}(\boldsymbol{x}, \boldsymbol{W}) = \sum_{j=1}^{J_{k|k-1}} \omega_{k|k}^{(j)} \cdot \mathcal{N}\left(\boldsymbol{x}; \hat{\boldsymbol{x}}_{k|k}^{(j)}, P_{k|k}^{(j)}\right). \tag{4.43}$$

According to subsection 3.1.2 the measurements of an extended object can be modeled as detections of measurement sources on the surface of the object. In the update equations of the extended object PHD filter the measurement set with assumed measurements from the same object are in the cell $\boldsymbol{W}$. In order to use the measurement set in a simultaneous update, the measurements and measurement matrices need to be stacked according to

$$\boldsymbol{z}_{\boldsymbol{W}} = \left(\boldsymbol{z}_{k,1}^{T}, \boldsymbol{z}_{k,2}^{T}, \dots \boldsymbol{z}_{k,|\boldsymbol{W}|}^{T}\right)^{T} \tag{4.44}$$

and

$$H_{\boldsymbol{W}}^{(j)} = \left(H_{k,1}^{(j)^T}, H_{k,2}^{(j)^T}, \dots, H_{k,|\boldsymbol{W}|}^{(j)^T}\right)^{T} \tag{4.45}$$

respectively. Since the measurement process is assumed to be nonlinear, the measurement matrices are computed as linearization using the Jacobi matrices as

$$H_{k,i}^{(j)} = \nabla_{\boldsymbol{x}^T} h_{k,i}^{(j)}(\boldsymbol{x})\big|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k|k-1}^{(j)}}. \tag{4.46}$$

The measurement sources or predicted measurements are computed using the nonlinear measurement equation and need to be stacked according to

$$\boldsymbol{y}_{\boldsymbol{W}}^{(j)} = \left(h_{k,1}^{(j)}\left(\boldsymbol{z}_{k,1}^{T}\right), h_{k,2}^{(j)}\left(\boldsymbol{z}_{k,2}^{T}\right), \dots, h_{k,|W|}^{(j)}\left(\boldsymbol{z}_{k,|\boldsymbol{W}|}^{T}\right)\right)^{T}. \tag{4.47}$$

The stacked measurement uncertainty matrices are given as

$$R_{\boldsymbol{W}} = \mathrm{blkdiag}\left(R_{k,1}, R_{k,2}, \dots, R_{k,|\boldsymbol{W}|}\right) \tag{4.48}$$

with the abbreviation “blkdiag” meaning to form a block diagonal matrix using the given matrices. With the stacked vectors and matrices, the update components can be computed using the following equations:

$$K_k^{(j)} = P_{k|k-1}^{(j)} \cdot {H_{\boldsymbol{W}}^{(j)}}^T \cdot \left(H_{\boldsymbol{W}}^{(j)} \cdot P_{k|k-1}^{(j)} \cdot {H_{\boldsymbol{W}}^{(j)}}^T + R_{\boldsymbol{W}} \right)^{-1}, \tag{4.49}$$

$$\hat{\boldsymbol{x}}_{k|k}^{(j)} = \hat{\boldsymbol{x}}_{k|k-1}^{(j)} + K_k^{(j)} \left(\boldsymbol{z}_{\boldsymbol{W}} - H_{\boldsymbol{W}}^{(j)} \cdot \hat{\boldsymbol{x}}_{k|k-1}^{(j)} \right), \tag{4.50}$$

$$P_{k|k}^{(j)} = P_{k|k-1}^{(j)} - K_k^{(j)} \cdot H_{\boldsymbol{W}}^{(j)} \cdot P_{k|k-1}^{(j)} . \tag{4.51}$$

Finally, the weighting of the GM component is calculated using

$$\omega_{k|k}^{(j)} = \omega_p \cdot \frac{\Gamma^{(j)} p_D}{d_{\boldsymbol{W}}} \cdot \Phi_{\boldsymbol{W}}^{(j)} \cdot \omega_{k|k-1}^{(j)}, \tag{4.52}$$

$$\Gamma^{(j)} = e^{-\gamma\left(\hat{\boldsymbol{x}}_{k|k-1}^{(j)}\right)} \left(\gamma\left(\hat{\boldsymbol{x}}_{k|k-1}^{(j)} \right) \right)^{|\boldsymbol{W}|}, \tag{4.53}$$

$$\Phi_{\boldsymbol{W}}^{(j)} = \mathcal{N}\left(\boldsymbol{z}_{\boldsymbol{W}}; H_{\boldsymbol{W}}^{(j)} \cdot \hat{\boldsymbol{x}}_{k|k-1}^{(j)}, H_{\boldsymbol{W}}^{(j)} \cdot P_{k|k-1}^{(j)} \cdot {H_{\boldsymbol{W}}^{(j)}}^T + R_{\boldsymbol{W}} \right) \cdot \prod_{\boldsymbol{z}_k \in \boldsymbol{W}} \frac{1}{\kappa_k(\boldsymbol{z}_k)} \tag{4.54}$$

with the normalization factors for each cell and partition as

$$d_{\boldsymbol{W}} = \delta_{|\boldsymbol{W}|,1} + p_D \sum_{l=1}^{J_{k|k-1}} \Gamma^{(l)} \cdot \Phi_{\boldsymbol{W}}^{(l)} \cdot \omega_{k|k-1}^{(l)} \tag{4.55}$$

and

$$\omega_p = \frac{\prod_{\boldsymbol{W} \in p} d_{\boldsymbol{W}}}{\sum_{p' \measuredangle \boldsymbol{z}_k} \prod_{\boldsymbol{W}' \in p'} d_{\boldsymbol{W}'}} \tag{4.56}$$

respectively. The cell normalization factor is taken as the sum over all weights in the cell $\boldsymbol{W}$ multiplied with the probability of detection. Additionally, the Kronecker delta $\delta_{|\boldsymbol{W}|,1}$ is added to the sum, which is defined as

$$\delta_{i,j} = \begin{cases} 1 & if\ i = j \\ 0 & else \end{cases}. \tag{4.57}$$

Therefore, the Kronecker delta is one if the cell contains only one measurement in the case of a clutter cell. The partition normalization is calculated using the product over all cell normalization factors in the partition p divided by the sum over the products of every cell normalization factor, calculating how likely the partition p is.

5 Spline functions

A spline is a piecewise defined curve where each piece, named span, is represented by a polynomial. These polynomials are joined together at the ends, allowing any continuous curve to be generated. For the use of spline functions, a walk parameter s is introduced, which increases as the curve is traversed. The creation of a curve f_s in 2-dimensional space is then done by using 2 spline functions and is given by

$$f_s(s) = \big(x(s), y(s)\big)^T . \tag{5.1}$$

The use of spline functions for the representation of arbitrary curves is a common tool in computational image processing. By their use, those curves can be represented analytically. They are an elegant and easy to use tool. By creating closed curves, any shape and thus the edge of any object can be adjusted. The spline functions considered in this thesis are basis spline or shortened B-spline functions [36, pp. 46-68].

Within this chapter, the basics of B-Splines are explained in section 5.1. In section 5.2 the contour, which is used as an extension model of a vehicle in the further course of this thesis, is then introduced. The possible representation of other objects is also explained.

5.1 Basis spline functions

Basis spline or B-spline functions are a weighted sum of N_b basis functions $B_n(s), n = 0, \dots, N_b - 1$ that are joined together at knots [36, pp. 46-68]. The basis functions consist of polynomials of degree d. The B-spline curve of (5.1) is then given as

$$x(s) = \sum_{n=0}^{N_b-1} x_n \cdot B_n(s) \tag{5.2}$$

$$y(s) = \sum_{n=0}^{N_b-1} y_n \cdot B_n(s) \tag{5.3}$$

with weights x_n and y_n for both splines respectively. The walk parameter s is within the Interval $I := [0, N_b]$ and the knots are equally spaced over I with uniform length for each span, which makes f_s a uniform B-spline. The representation of (5.2) and (5.3) can be expressed compactly in matrix notation. The B-spline functions are then given as

$$x(s) = \boldsymbol{q}^x \cdot B(s) \tag{5.4}$$

$$y(s) = \boldsymbol{q}^y \cdot B(s) \tag{5.5}$$

with a vector of basis functions $B(s) = \Big(B_0(s), B_1(s), \dots, B_{N_b-1}(s)\Big)^T$ and two vectors of weights $\boldsymbol{q}^x = (x_0, x_1, \dots, x_{N_b-1})$ and $\boldsymbol{q}^y = (y_0, y_1, \dots, y_{N_b-1})$. For the representation of a point in 2-dimensional space $\big(x(s), y(s)\big)^T$ the weight vectors are combined to a weight matrix

$$Q^{xy} = \begin{pmatrix} x_0 & x_1 & \dots & x_{N_b-1} \\ y_0 & y_1 & \dots & y_{N_b-1} \end{pmatrix} \tag{5.6}$$

so every point on the spline function can be represented as

$$f_s(s) = Q^{xy} \cdot B(s)\,. \tag{5.7}$$

The basis functions are constructed to sum up to 1 at each point. When choosing quadratic basis functions and assuming equally spaced knots with uniform length, the first basis function is given as

$$B_0(s) = \begin{cases} 0.5 \cdot s^2 & if\ 0 \le s < 1 \\ 0.75 - (s-1.5)^2 & if\ 1 \le s < 2 \\ 0.5 \cdot (s-3)^2 & if\ 2 \le s < 3 \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

while the other basis functions are the first basis function simply shifted by n and are given as

$$B_n(s) = B_0(s-n)\,. \tag{5.9}$$

An illustration of quadratic uniform spline basis functions is given in Figure 5.1. In this figure 7 weighted basis functions and the resulting sum of those weighted basis functions is shown.

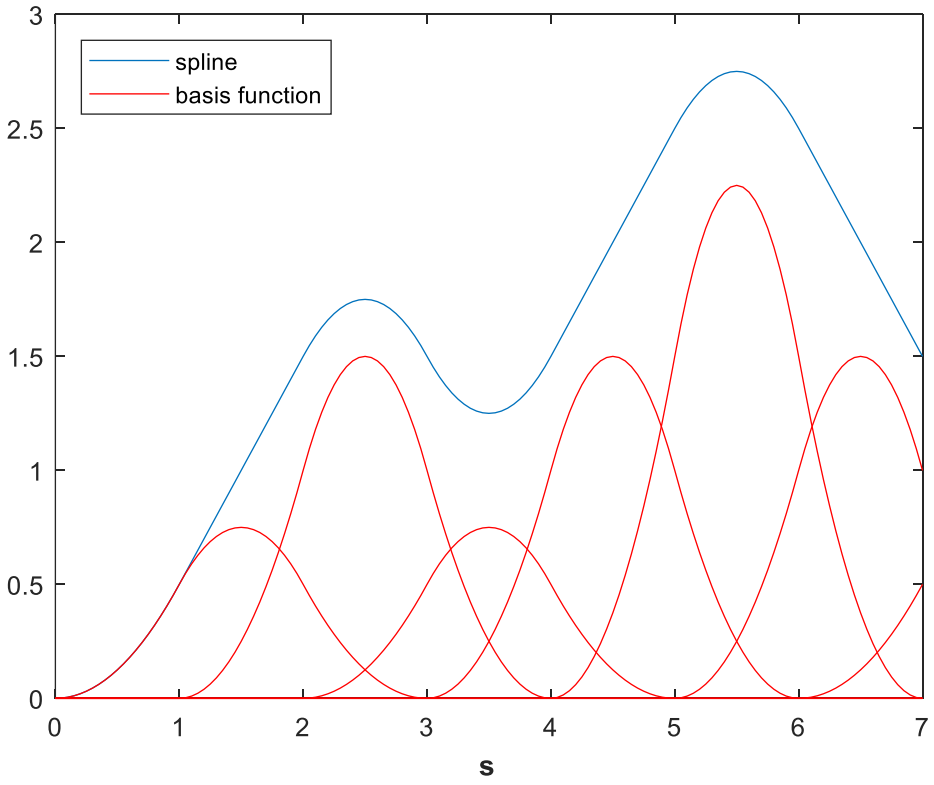

Figure 5.1: Weighted sum of spline basis functions

The weights in this example are $\boldsymbol{q}^x = (1,2,1,2,3,2,1)$. The last two basis functions are only partially visible in this figure, since the interval I only goes up to $s = 7$ and is given as $I = [0,7]$.

The figure also shows that only one basis function is active in the range from $s \in [0,1[$ and two are active in the range from $s \in [1,2[$, while three basis functions are always active in the remaining interval. This means, that the boundary conditions of the B-spline function cannot be controlled completely. In order to be able to create a closed contour, however, this

control must be given. This can be achieved by using periodic basis functions. To create a periodic B-spline, d basis functions must be added, so that the number of basis functions and weights are changed to $N_b' = N_b + d$, while the interval I remains unchanged. In the case of quadratic basis functions $B_{-1}(s)$ and $B_{-2}(s)$ must be added to the set of basis functions. The weights are also used periodically. The difference between periodic and non-periodic B-spline functions is illustrated in Figure 5.2. The colors in the figure are used equally to Figure 5.1. The left part of the figure shows a B-Spline function with non-periodic basis functions, while the right part of the figure shows a B-Spline function with periodic basis functions. In the right part of the figure all functions that are truncated at $s = 4$ are continued at the beginning of the interval, creating a periodic B-spline with full control over the boundary conditions.

Finally, there are multiple knots to mention in this chapter. For a polynomial, $d - 1$ derivatives can be determined. To reduce this value, a knot can be specified more than once within the interval. With each further introduction of the same knot, the number of possible derivatives is reduced by 1. In the case of quadratic basis functions, the introduction of a double knot means that the function can no longer be derived at this point.

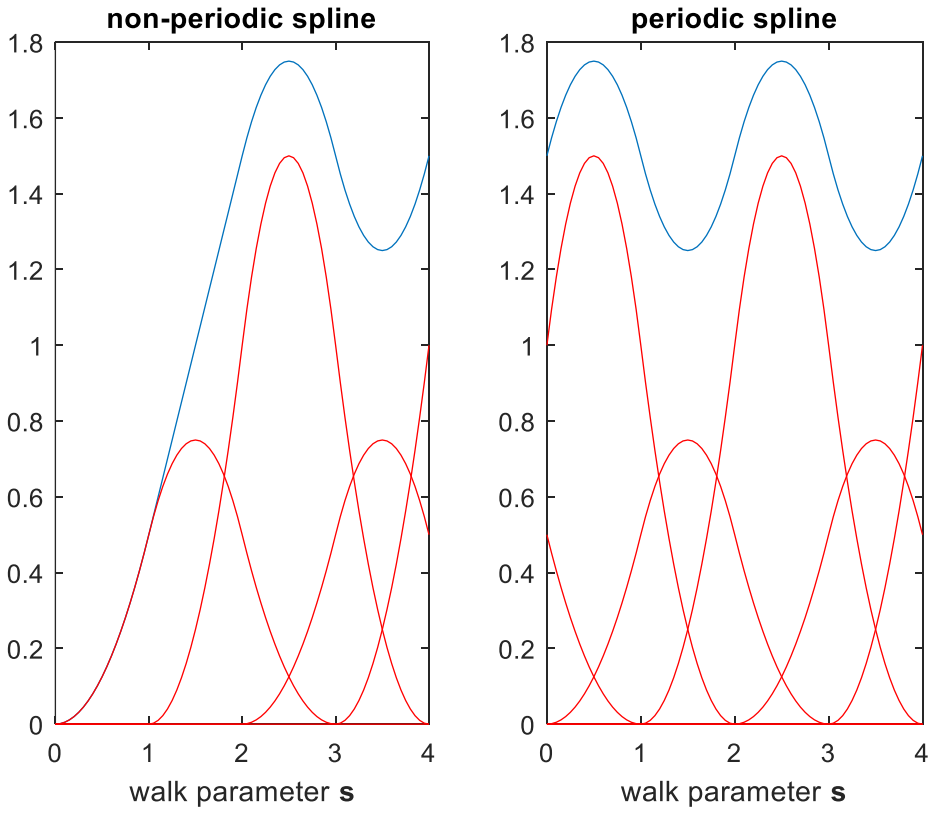

Figure 5.2: Difference between periodic and no periodic B-splines

This is shown as an edge in the plot of the function, which represents the purpose of multiple knots.

5.2 B-spline vehicle contour function

Using quadratic periodic uniform B-spline functions, this section describes the contour function $C(s)$ [1] that will be used later in the thesis as shape model for the extended object tracker. Such a contour function must meet some conditions in order to be effectively integrated into a tracking algorithm. It must be movable, rotatable and scalable in the 2-dimensional plane. The contour is created by selecting weights that represent a rectangular

shape. Those weights are called basis points and are denoted as P in the remainder of this thesis. Since the contour should be scalable in both dimensions, the size of the rectangle does not matter at first. The basis points are selected as

$$P = \begin{pmatrix} 1 & 1 & 0 & -1 & -1 & -1 & 0 & 1 \\ 0 & 0.5 & 0.5 & 0.5 & 0 & -0.5 & -0.5 & -0.5 \end{pmatrix}. \tag{5.10}$$

Using these basis points, the walk parameter s on the contour is given as $s \in [0, 8]$, so $N_b = 8$. The contour function is then given as

$$C(s) = P \cdot B(s). \tag{5.11}$$

This contour must now be able to be moved, rotated and scaled. To move the center $\boldsymbol{m}$ of the contour, the new center is simply added to each point of the contour.

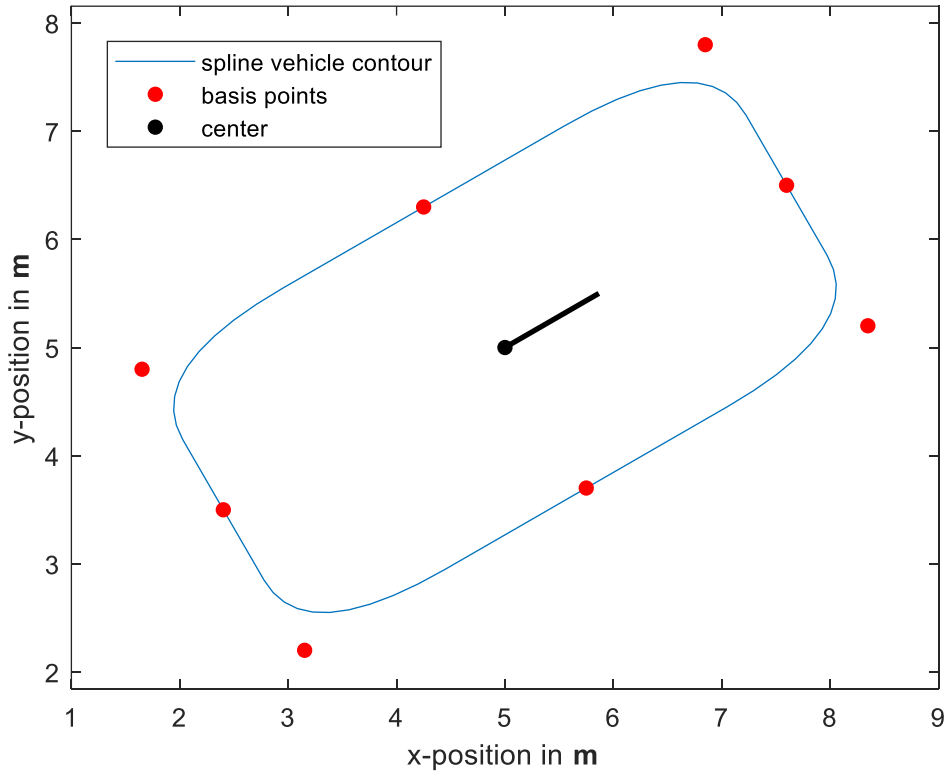

Figure 5.3: Spline vehicle contour model

In order to achieve a rotation, a rotation matrix R_φ with the corresponding rotation angle φ is calculated. The rotation matrix is given as

$$R_\varphi = \begin{pmatrix} cos(\varphi) & -sin(\varphi) \\ sin(\varphi) & cos(\varphi) \end{pmatrix}. \tag{5.12}$$

Furthermore, the scaling factors s_x and s_y are introduced for scaling the contour in x and y dimension respectively. The scaling matrix S^C is then given as

$$S^C = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix}. \tag{5.13}$$

These tools can now be used to determine any point $\boldsymbol{y}$ on the contour. With a given fixed walk parameter s, the searched point on the contour can be specified as

$$\boldsymbol{y} = \boldsymbol{m} + R_{\varphi} \cdot S^{C} \cdot C(s) \,. \tag{5.14}$$

A moved, rotated and scaled spline vehicle contour is shown in Figure 5.3. As basis points the given basis points from (5.10) are used. The center is set to $\boldsymbol{m} = (5,5)^{T}$, the heading angle is $\varphi = 30°$ and the scaling factors are set to $s_x = s_y = 3$.

The spline vehicle contour is an elegant and easy to use contour model, because the basis points simply need to be specified in Cartesian coordinates. Afterwards, this contour can easily be scaled and moved in x and y dimension and rotated with a given heading angle. So, this model can be used to represent different vehicles. The simple scaling enables to use the vehicle contour for passenger cars, as well as for trucks. To track other objects than vehicles, other contour models could be used, which is briefly presented in the next section.

5.3 Other B-spline contour models

In order to represent other objects than vehicles, other spline contour models must be used. This can easily be achieved by changing the basis points of the contour model. In this section the usability of other contours for the developed tracking approach will be investigated.

First, the basis points for two other shapes are given, then it is explained why these shapes can be used for a tracking algorithm and which property must be given for the use.

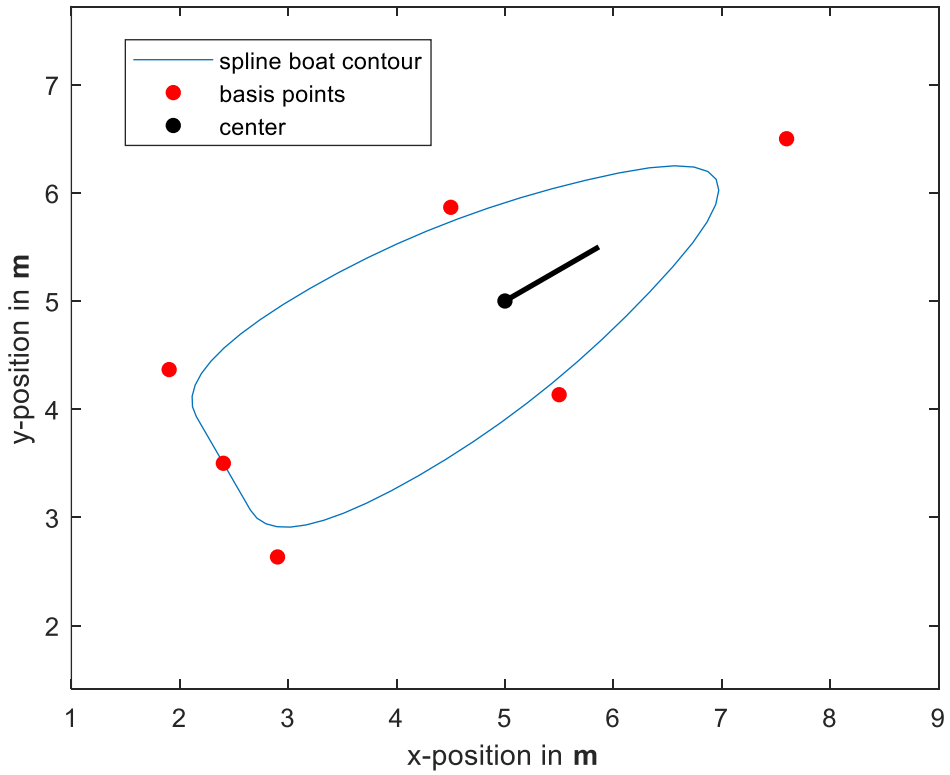

Figure 5.4: Spline boat contour model

The first further contour is shown in Figure 5.4. Here the shape of a boat is shown, which can be generated by specifying 6 basis points. This contour is also shifted, rotated and scaled. However, the scaling factors in this figure are $s_x = 3$ and $s_y = 2$, the center point and orientation are identical to Figure 5.3. The basis points for the boat contour are

$$P_{boat} = \begin{pmatrix} 1 & 0 & -1 & -1 & -1 & 0 \\ 0 & 0.5 & 0.5 & 0 & -0.5 & -0.5 \end{pmatrix}. \tag{5.15}$$

It is also possible to create contours for objects that do not occur in everyday scenarios, such as a cross. The contour of a cross is not shown here in any illustration but can be created by using the basis points of a cross. Those basis points could be given as

$$P_{cross} = \begin{pmatrix} 5 & 1 & 1 & -1 & -1 & -5 & -5 & -1 & -1 & 1 & 1 & 5 \\ 1 & 1 & 5 & 5 & 1 & 1 & -1 & -1 & -5 & -5 & -1 & -1 \end{pmatrix}. \tag{5.16}$$

By specifying different basis points, various contours can be created. In order to use this contour for a tracking algorithm, it must enclose a star-convex set. A star-convex set is a set in which each connecting line from the center to the edge of the set is also completely within the set. This property is important in order to be able to assign exactly one point on the contour to each angle.

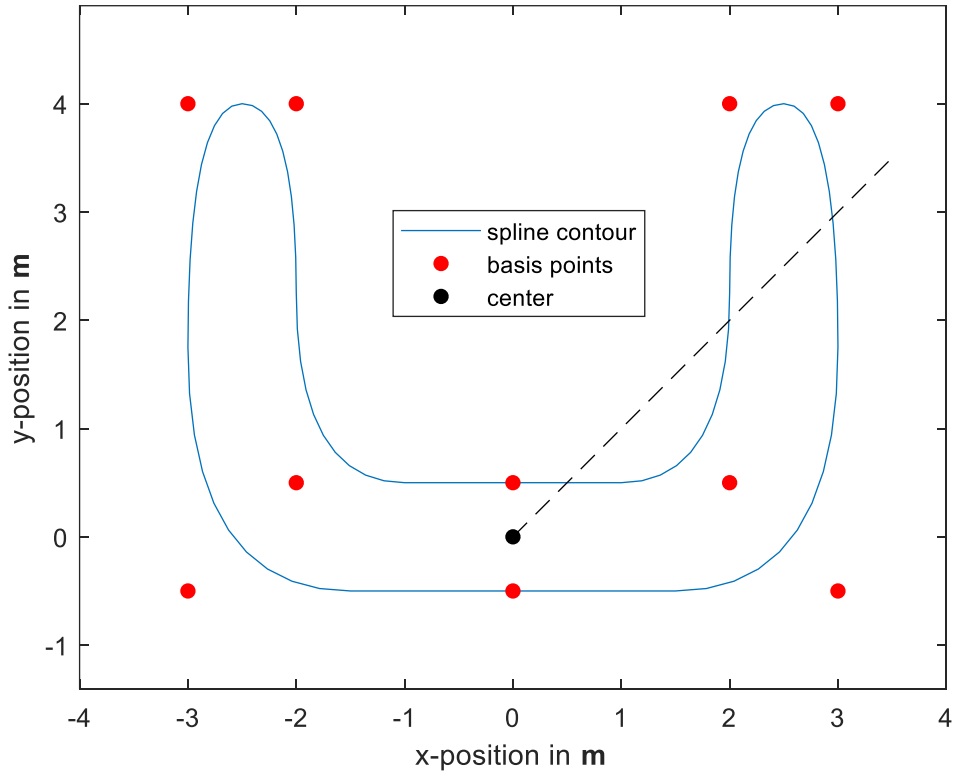

Figure 5.5: Non-star-convex spline contour

An illustration of this property is shown in Figure 5.5. In this figure a non-star-convex spline contour is illustrated. If the contour is to be scanned from the center point, three points on the contour can be assigned to the 45° angle, which leads to problems in the further course of the tracking algorithm. In the remainder of this thesis, therefore, only star-convex contours are considered [1].

6 The spline vehicle tracking algorithm

The aim of this section is to derive a measurement model for vehicles using quadratic periodic uniform B-spline functions [1]. This measurement model will then be used to track vehicles measured with 2D-LIDAR sensors. Since a LIDAR sensor is a high-resolution sensor, it can be assumed that more than one measurement per time step is generated by the object, which is why a measurement model for extended objects is presented. As an introduction, section 6.1 introduces the system state and some other notations used in the further course of this chapter. Afterwards, section 6.2 explains the assignment of each measurement to a measurement source on the surface of the vehicle, or the prediction of the measurements. In order to track the extended object, the measurement model must be integrated into a Bayesian filter. To perform a Kalman update, the predicted measurements must be derived with respect to the object state, as described in section 6.3. In the last two sections of the chapter, the measurement model for tracking one or more objects is integrated into an EKF and a PHD filter framework respectively.

6.1 Notations

First, the system state of the extended object, where the extension is modeled with a spline contour, is defined in this section. Based on (3.1) and Figure 3.2, the state is given as

$$\boldsymbol{x}_k = \left(x_k, y_k, v_k, \varphi_k, \omega_k, s_{x,k}, s_{y,k}\right)^T \tag{6.1}$$

with the position $\boldsymbol{m}_k = (x_k, y_k)^T$ of the center, the polar velocity v_k, the heading angle φ_k, the turn rate $\omega_k = \frac{d\varphi_k}{dt}$ and the scaling factors $s_{x,k}$ and $s_{y,k}$ of the given spline contour forming the scaling matrix introduced in (5.13). In the further course of this thesis the spline vehicle contour is defined by the basis points P introduced in (5.10). The goal of the tracking algorithm is to provide an optimal state estimation within a Bayesian filter at any time step.

The predicted system state is updated at time k by a given measurement set $\boldsymbol{Z}_k = \left\{\boldsymbol{z}_{k,l}\right\}_{l=1}^{n_k}$. Thus, n_k measurements are available at each time step. Note that n_k can always be different at each time step. The measurements within the measurement set are assumed to be generated from a certain measurement source on the surface of the vehicle. The first step is to associate each measurement to one measurement source, it's the prediction step of the measurements. The set of predicted measurements within a time step k are denoted as $\boldsymbol{Y}_k = \left\{\boldsymbol{y}_{k,l}\right\}_{l=1}^{n_k}$.

To avoid confusions caused by similar names, the walk parameter on the spline contour is called τ within the tracking algorithm. The contour is still called C, so with a given $\tau \in [0,8]$ the contour point is given as $C(\tau)$.

6.2 Measurement prediction

To assign a measurement source to each measurement, it is assumed that the measurement and the measurement source have the same angle relative to the center of the object, which is illustrated in Figure 6.1 [1]. The orientation of the object must be included in this assignment. By stating this rule, it becomes clear why only star-convex contours are considered in this thesis. Without this restriction no unambiguous assignment can be made. Note that other assignments like taking the nearest point on the contour are also possible, where also non-star-convex contours could be considered. However, this does not represent

a significant limitation, as typical objects of interest in automotive applications, such as vehicles or pedestrians, do not suffer if only star-convex shapes are considered.

In order to find the measurement source to each measurement, the first step is to specify them in local coordinates, so that the origin is always at the center of the object. The rotation of the object must also be calculated out in this step. The measurement in local coordinates is then given as

$$\hat{\boldsymbol{z}} = R_{\varphi}^{-1} \cdot (\boldsymbol{z} - \boldsymbol{m}) \tag{6.2}$$

with the inverse of the rotation matrix R_{φ} specified in (5.12) and the difference between the measurement $\boldsymbol{z}$ and the center $\boldsymbol{m}$ of the object.

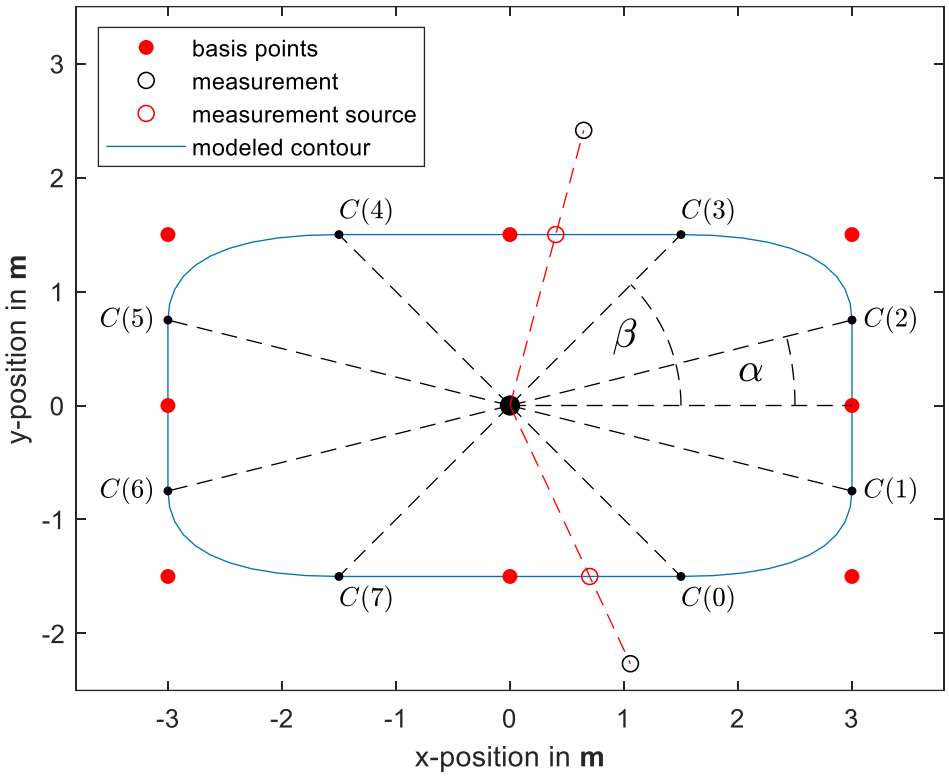

Figure 6.1: Measurement prediction and contour division

The measurement source can be calculated the same way in local coordinates and is then given as

$$\hat{\boldsymbol{y}} = R_{\varphi}^{-1} \cdot (\boldsymbol{y} - \boldsymbol{m}) \tag{6.3}$$

with the measurement source $\boldsymbol{y}$. Using (5.14) for the measurement source without the scaling matrix, the measurement source in local coordinates can be stated as

$$\hat{\boldsymbol{y}} = R_{\varphi}^{-1} \cdot \left(\boldsymbol{m} + R_{\varphi} \cdot C(\tau) - \boldsymbol{m}\right) = R_{\varphi}^{-1} \cdot R_{\varphi} \cdot C(\tau) = C(\tau) = P \cdot B(\tau) \tag{6.4}$$

where the last step is taken from (5.11). The point on the spline contour is therefore already given in local coordinates. In order to find the measurement source on the contour, the walk parameter τ must be specified for each measurement. The first step is to compute the active basis functions of the part of the spline contour where the measurement source is located.

The first step is therefore to calculate the angle of the measurement in local coordinates relative to the center as $\delta = \arctan\left(\frac{\hat{z}_y}{\hat{z}_x}\right)$ with the x and y coordinates $\hat{z}_x$ and $\hat{z}_y$ of $\hat{\boldsymbol{z}}$ respectively. As indicated at the beginning of this chapter, it is required that the measurement source in local coordinates $\hat{\boldsymbol{y}}$ has the same angle δ relative to the center. According to Figure 6.2, it becomes clear that only 3 basis functions are active for each τ.

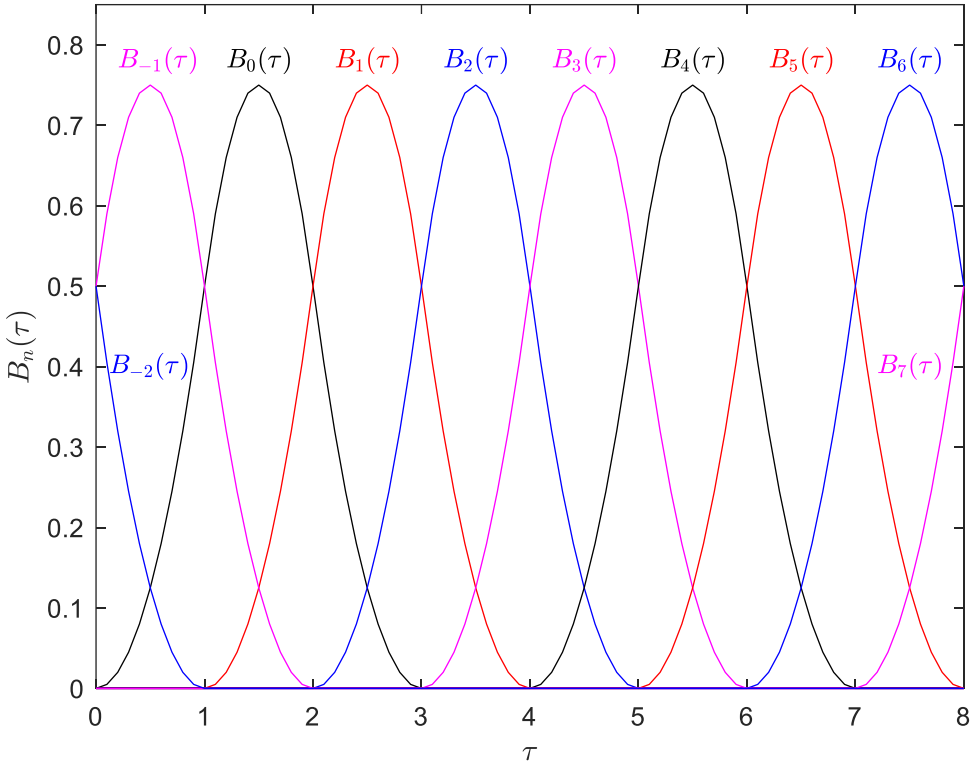

Figure 6.2: Non-weighted periodic spline basis functions

Thus, (6.4) can be stated as

$$\hat{\boldsymbol{y}} = P \cdot B(\tau) = \begin{pmatrix} p_{x,a} & p_{x,b} & p_{x,c} \\ p_{y,a} & p_{y,b} & P_{y,c} \end{pmatrix} \cdot \begin{pmatrix} B_a(\tau) \\ B_b(\tau) \\ B_c(\tau) \end{pmatrix} = P_\tau \cdot B_\tau(\tau) \tag{6.5}$$

with the x and y components of the basis points given as p_x and p_y respectively. To compute these active basis points, the contour can be divided in pieces illustrated in Figure 6.1. Every piece ends at a knot where the spans of the spline function are joined. Each knot is located at the center of the connection line of two basis points. Given this information, the angles α and β of Figure 6.1 can be calculated and the start and ending angles of the contour division can be determined. Note that the angles α and β must be calculated using the scaled basis points given as

$$P^c = S^c \cdot P\ . \tag{6.6}$$

Another feature of the basis functions is that the values of the individual basis functions is equal for every uniform interval starting at a natural number $\tau \in \mathbb{N}_0$. The different points on the contour are only calculated with different basis points. Therefore, a shift of τ into the

uniform interval starting at zero leads to the same measurement source in local coordinates using the active basis points. The shift of τ is done by finding the knot $k(\tau)$ defining the three active basis points. The knot $k(\tau)$ is always the knot starting the interval. For $\tau \in [3,4[$ the knot defining the three active basis points is $k(\tau) = 3$, so $k(\tau) \in \{0, 1, \dots, 7\}$. The shifted τ is denoted as

$$\tau' = \tau - k(\tau)\,. \tag{6.7}$$

Using τ', (6.5) can be stated as

$$\hat{\boldsymbol{y}} = P_\tau \cdot \begin{pmatrix} B_{-2}(\tau') \\ B_{-1}(\tau') \\ B_0(\tau') \end{pmatrix}. \tag{6.8}$$

By viewing Figure 6.2, the definition area of each basis function can be extracted. Using (5.8) and (5.9), (6.8) can be stated as

$$\hat{\boldsymbol{y}} = P_\tau \cdot \begin{pmatrix} 0.5\tau'^2 - \tau' + 0.5 \\ -\tau'^2 + \tau' + 0.5 \\ 0.5\tau'^2 \end{pmatrix} = P_\tau \cdot \begin{pmatrix} 0.5 & -1 & 0.5 \\ -1 & 1 & 0.5 \\ 0.5 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \tau'^2 \\ \tau' \\ 1 \end{pmatrix} = P_\tau \cdot M \cdot \begin{pmatrix} \tau'^2 \\ \tau' \\ 1 \end{pmatrix} \tag{6.9}$$

Given the matrix of active basis points P_τ and the spline representation matrix M, the product of these 2 matrices is combined as

$$P_\tau \cdot M = (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \tag{6.10}$$

with $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}^{2\times 1}$. In order to find the walk parameter τ defining the point on the contour used as measurement source in local coordinates of a measurement, the assigning assumption is now used. If two vectors have the same angle relative to the same point, they are linearly dependent, and their cross product is zero. Suppose there are two 2-dimensional vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ in 3-dimensional space then the cross product is defined as

$$\begin{pmatrix} x_1 \\ x_2 \\ 0 \end{pmatrix} \times \begin{pmatrix} y_1 \\ y_2 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ x_1 \cdot y_2 - x_2 \cdot y_1 \end{pmatrix} \tag{6.11}$$

Since the cross product is normally defined for vectors in 3-dimensional space, the cross product in 2-dimensional space is defined as the z component of the cross product of 2-dimensional vectors in 3-dimensional space in this thesis. In this case, this definition makes sense, since only linearly dependent vectors are considered for the following calculation. This cross product is also zero for these vectors, since no plane can be spanned by them. Therefore, the cross product of two vectors in 2-dimensional space is defined as

$$\boldsymbol{x} \times \boldsymbol{y} := \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \times \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = x_1 \cdot y_2 - x_2 \cdot y_1 = \det\big((\boldsymbol{x}, \boldsymbol{y})\big) \tag{6.12}$$

in this thesis. Given a measurement and using the assigning assumption, the measurement source can now be calculated using

$$0 = \hat{\boldsymbol{y}} \times \hat{\boldsymbol{z}}\,. \tag{6.13}$$

By inserting (6.9) and (6.10) for $\hat{\boldsymbol{y}}$, (6.13) can be stated as

$$0 = (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \cdot \begin{pmatrix} \tau'^2 \\ \tau' \\ 1 \end{pmatrix} \times \hat{\boldsymbol{z}} = (\boldsymbol{a} \times \hat{\boldsymbol{z}}) \cdot \tau'^2 + (\boldsymbol{b} \times \hat{\boldsymbol{z}}) \cdot \tau' + (\boldsymbol{c} \times \hat{\boldsymbol{z}}) \,. \tag{6.14}$$

The second equation of (6.14) is not intuitive but can easily be recalculated. To make the equation more compact the cross products are now defined as $u_q := \boldsymbol{q} \times \hat{\boldsymbol{z}}$. Using this definition, (6.14) can now be stated as

$$0 = u_a \cdot \tau'^2 + u_b \cdot \tau' + u_c \,. \tag{6.15}$$

If the active basis points define a line segment u_a is zero and τ' is given as

$$\tau' = -\frac{u_c}{u_b}. \tag{6.16}$$

Otherwise, τ' can be calculated using

$$\tau' = \frac{-\sqrt{u_b^2 - 4 \cdot u_a \cdot u_c} - u_b}{2 \cdot u_a}. \tag{6.17}$$

After calculating τ', (6.7) can be used to calculate τ. In a final step, the measurement source can now be determined with (5.14).

When using other spline contour models, the calculations established in this section can be used as well. The only thing that must be done individually for every spline model is the contour division illustrated in Figure 6.1 and the assignment of the active basis functions.

6.3 Deriving the predicted measurement

To use the spline measurement model for tracking an extended object, the model must be integrated in a Bayesian filter. Since the spline representation is a nonlinear problem, an EKF is used. According to (2.23), the measurement matrix H is the Jacobi matrix of the measurement equation, which is represented by (5.14) for the spline measurement model. So, the goal of this chapter is to derive the predicted measurements calculated in the previous section with respect to the system state [1] as

$$\frac{\partial h(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_k} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}_k}. \tag{6.18}$$

Due to the extended object assumption it can be assumed that several predicted measurements are present in each time step. The handling of all measurements and predicted measurements together is presented in the next section, while this section is about to calculate the derivative for one predicted measurement. To derivate the predicted measurement with respect to the whole extended object state given in (6.1), it must be derived with respect to all the state parameters as

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}_k} = \left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{m}}, \frac{\partial \boldsymbol{y}}{\partial v}, \frac{\partial \boldsymbol{y}}{\partial \varphi}, \frac{\partial \boldsymbol{y}}{\partial \omega}, \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}_{shape}} \right) \tag{6.19}$$

with $\boldsymbol{x}_{shape} = \left(s_x, s_y\right)^T$ and $\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}_k} \in \mathbb{R}^{2\times 7}$. Since (5.14) is only dependent on the position,

orientation and extension, the derivatives of the predicted measurements with respect to the velocity and turn rate are given as

$$\frac{\partial \boldsymbol{y}}{\partial v} = \frac{\partial \boldsymbol{y}}{\partial \omega} = (0,0)^T . \tag{6.20}$$

The three remaining derivatives can be calculated using the product rule and the chain rule. Since the walk parameter τ is also dependent of the object state, the chain rule must be used when deriving the contour function $C(\tau(\boldsymbol{x}_k))$. The first derivative with respect to the object center is then given as

$$\frac{\partial}{\partial \boldsymbol{m}}\Big(\boldsymbol{m} + R_\varphi \cdot S^C \cdot C(\tau)\Big) = I_2 + R_\varphi \cdot S^C \cdot \frac{\partial C(\tau)}{\partial \tau} \cdot \frac{\partial \tau}{\partial \boldsymbol{m}} \tag{6.21}$$

with the 2×2 identity matrix I_2 and $\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{m}} \in \mathbb{R}^{2\times 2}$. In this equation only the chain rule must be used for the derivative of the contour function. For the derivative of the predicted measurement with respect to the orientation, also the product rule must be used. The derivative is then given as

$$\frac{\partial}{\partial \varphi}\Big(\boldsymbol{m} + R_\varphi \cdot S^C \cdot C(\tau)\Big) = \frac{\partial R_\varphi}{\partial \varphi} \cdot S^C \cdot C(\tau) + R_\varphi \cdot S^C \cdot \frac{\partial C(\tau)}{\partial \tau} \cdot \frac{\partial \tau}{\partial \varphi} \tag{6.22}$$

with $\frac{\partial \boldsymbol{y}}{\partial \varphi} \in \mathbb{R}^{2\times 1}$. To complete the derivative of (6.19), the last formula needed is the derivative of the predicted measurement with respect to the extension. Therefore, also the product rule and the chain rule must be used. The derivative is then given as

$$\frac{\partial}{\partial x_{shape}}\Big(\boldsymbol{m} + R_\varphi \cdot S^C \cdot C(\tau)\Big) = R_\varphi \cdot \mathrm{diag}\big(C(\tau)\big) + R_\varphi \cdot S^C \cdot \frac{\partial C(\tau)}{\partial \tau} \cdot \frac{\partial \tau}{\partial \boldsymbol{x}_{shape}} \tag{6.23}$$

with $\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}_{shape}} \in \mathbb{R}^{2\times 2}$ and the function $\mathrm{diag}(\boldsymbol{x})$ forming a diagonal matrix with the entries of the vector $\boldsymbol{x}$. When considering (6.21)-(6.23) it becomes clear that several derivatives used in those equations have to be calculated as well. The first one presented is the derivative of the contour function. Since the functional values of the spline basis functions are the same for every unit interval, the derivative of the contour function with respect to the walk parameter τ is the same as the derivative with respect to τ'. Therefore, (6.9) and (6.10) can be used for the derivative of the contour function, which is then given as

$$\frac{\partial C(\tau)}{\partial \tau} = (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \cdot \begin{pmatrix} 2 \cdot \tau' \\ 1 \\ 0 \end{pmatrix}. \tag{6.24}$$

The next part is to calculate the derivatives of the walk parameter τ with respect to the extended object state. Here two cases must be distinguished. In the first case, the spline segment is a line, so (6.16) must be used to calculate the derivative. Again, the derivative can be done with τ' instead of τ. Using the quotient rule, the formula can be stated as

$$\frac{\partial \tau}{\partial g} = \frac{\partial}{\partial g}\left(-\frac{u_c}{u_b}\right) = -\frac{\frac{\partial u_c}{\partial g} \cdot u_b - u_c \cdot \frac{\partial u_b}{\partial g}}{u_b^2} . \tag{6.25}$$

The derivatives of the walk parameter will be very similar, so the calculation is only done once using the variable g as representation for the parameters of the object state. The numbers u_q with $q \in \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}\}$ where defined as $u_q := \boldsymbol{q} \times \hat{\boldsymbol{z}}$, so their derivatives using the product rule can be stated as

$$\frac{\partial u_q}{\partial g} = \frac{\partial}{\partial g}(\boldsymbol{q} \times \hat{\boldsymbol{z}}) = \frac{\partial \boldsymbol{q}}{\partial g} \times \hat{\boldsymbol{z}} + \boldsymbol{q} \times \frac{\partial \hat{\boldsymbol{z}}}{\partial g} = \boldsymbol{q} \times \frac{\partial \hat{\boldsymbol{z}}}{\partial g} \tag{6.26}$$

since $\frac{\partial \boldsymbol{q}}{\partial g} = 0$. Using (6.26), (6.25) can be stated as

$$-u_b^2 \cdot \frac{\partial \tau}{\partial g} = \left(\boldsymbol{c} \times \frac{\partial \hat{\boldsymbol{z}}}{\partial g}\right) \cdot u_b - u_c \cdot \left(\boldsymbol{b} \times \frac{\partial \hat{\boldsymbol{z}}}{\partial g}\right). \tag{6.27}$$

The last simplification of (6.27) can be done using the distributive law. The derivative of τ' can then be stated as

$$u_b^2 \cdot \frac{\partial \tau}{\partial g} = (\boldsymbol{b} \cdot u_c - \boldsymbol{c} \cdot u_b) \times \frac{\partial \hat{\boldsymbol{z}}}{\partial g}. \tag{6.28}$$

In the second case the spline segment is a curve, so (6.17) must be used to calculate the derivative. Since the equation gets quite big the square root of the quadratic formula is defined as $\Theta := \sqrt{u_b^2 - 4 \cdot u_a \cdot u_c}$. Using this definition, the quotient rule and the chain rule, the derivative can be stated as

$$-4u_a^2 \frac{\partial \tau}{\partial g} = \left(\frac{1}{2\Theta}\left(2u_b \frac{\partial u_b}{\partial g} - 4\left(\frac{\partial u_a}{\partial g} u_c + u_a \frac{\partial u_c}{\partial g}\right)\right) + \frac{\partial u_b}{\partial g}\right) 2u_a - 2(\Theta + u_b) \frac{\partial u_a}{\partial g}. \tag{6.29}$$

When multiplying this equation out and using (6.26), (6.29) can be stated as

$$\begin{aligned} -2u_a^2 \Theta \frac{\partial \tau}{\partial g} = {} & u_b u_a \left(\boldsymbol{b} \times \frac{\partial \hat{\boldsymbol{z}}}{\partial g}\right) - 2u_c u_a \left(\boldsymbol{a} \times \frac{\partial \hat{\boldsymbol{z}}}{\partial g}\right) - 2u_a^2 \left(\boldsymbol{c} \times \frac{\partial \hat{\boldsymbol{z}}}{\partial g}\right) \\ & + u_a \Theta \left(\boldsymbol{b} \times \frac{\partial \hat{\boldsymbol{z}}}{\partial g}\right) - (\Theta^2 + u_b \Theta) \left(\boldsymbol{a} \times \frac{\partial \hat{\boldsymbol{z}}}{\partial g}\right) \end{aligned} \tag{6.30}$$

The final equation can be calculated by using the distributive law again, doing some simplifications and using the definition of Θ by inserting $\Theta^2 = u_b^2 - 4 \cdot u_a \cdot u_c$. The final equation is then given as

$$-2u_a^2 \Theta \frac{\partial \tau}{\partial g} = (\boldsymbol{a}(2u_a u_c - u_b \Theta - u_b^2) + \boldsymbol{b}(u_b u_a + u_a \Theta) - 2u_a^2 \boldsymbol{c}) \times \frac{\partial \hat{\boldsymbol{z}}}{\partial g}. \tag{6.31}$$

The last equations to calculate are the derivatives of the measurement in local coordinates with respect to the object state. The measurement in local coordinates is given by (6.2). The derivative with respect to the object center is given as

$$\frac{\partial \hat{\boldsymbol{z}}}{\partial \boldsymbol{m}} = \frac{\partial}{\partial \boldsymbol{m}}\left(R_\varphi^{-1} \cdot (\boldsymbol{z} - \boldsymbol{m})\right) = -R_\varphi^{-1} \tag{6.32}$$

The derivative with respect to the orientation is given as

$$\frac{\partial \hat{\boldsymbol{z}}}{\partial \varphi} = \frac{\partial}{\partial \varphi}\left(R_{\varphi}^{-1} \cdot (\boldsymbol{z} - \boldsymbol{m})\right) = \frac{\partial R_{\varphi}^{-1}}{\partial \varphi}(\boldsymbol{z} - \boldsymbol{m}) \tag{6.33}$$

and the derivative with respect to the extension is given as

$$\frac{\partial \hat{\boldsymbol{z}}}{\partial \boldsymbol{x}_{shape}} = \frac{\partial}{\partial \boldsymbol{x}_{shape}}\left(R_{\varphi}^{-1} \cdot (\boldsymbol{z} - \boldsymbol{m})\right) = 0\,. \tag{6.34}$$

Since (6.34) is zero, (6.23) simplifies to

$$\frac{\partial}{\partial \boldsymbol{x}_{shape}}\left(\boldsymbol{m} + R_{\varphi} \cdot S^{C} \cdot C(\tau)\right) = R_{\varphi} \cdot \mathrm{diag}\big(C(\tau)\big)\,. \tag{6.35}$$

Now all the calculations for the derivative of the predicted measurement are done. In the next section the presented spline measurement model is integrated in the EKF framework.

6.4 The spline EKF filter

When integrating the spline measurement model into an EKF framework, the predicted object state must be used to calculate the predicted measurements and their derivatives. In equation (6.2) the predicted object center $\boldsymbol{m}_{k|k-1}$ and the predicted orientation $\varphi_{k|k-1}$ have to be used to calculate the measurement in local coordinates. Those quantities also must be considered when calculating the derivatives in section 6.3. The scale factor used in those equations can be taken as updated scale factor from the previous time step $S^{C}_{k-1|k-1}$ or the predicted scale factor from the present time step $S^{C}_{k|k-1}$ if the extension is assumed to be constant. Using this assumption, the prediction step returns the updated scale factor from the previous time step. With a non-constant extension, the predicted scale factor must be used. In every time step now section 6.2 must be used to calculate the predicted measurement set $\boldsymbol{Y}_k$ with a predicted measurement for every measurement. Afterwards, the corresponding active basis functions and the walk parameters τ_i have to be used to calculate the set of derived predicted measurements $\frac{\partial \boldsymbol{Y}_k}{\partial \boldsymbol{x}_k} = \left\{\frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{x}_k}\middle| i = 1 \dots n_k\right\}$.

To integrate the measurement model in the EKF framework, the calculated quantities from the previous sections must be stacked [1]. The measurements and the predicted measurements must be used in global coordinates and stacked like

$$\boldsymbol{z}_k = (\boldsymbol{z}_{k,1}^T; \boldsymbol{z}_{k,2}^T; \dots; \boldsymbol{z}_{k,n_k}^T) \tag{6.36}$$

$$\boldsymbol{y}_k = (\boldsymbol{y}_{k,1}^T; \boldsymbol{y}_{k,2}^T; \dots; \boldsymbol{y}_{k,n_k}^T) \tag{6.37}$$

with the semicolon meaning to stack the quantities on top of each other forming two vectors $\boldsymbol{z}_k, \boldsymbol{y}_k \in \mathbb{R}^{2n_k}$. The derivatives of the predicted measurements must be stacked like

$$H_k = \left(\frac{\partial \boldsymbol{y}_{k,1}}{\partial \boldsymbol{x}_k}; \frac{\partial \boldsymbol{y}_{k,2}}{\partial \boldsymbol{x}_k}; \dots; \frac{\partial \boldsymbol{y}_{k,n_k}}{\partial \boldsymbol{x}_k}\right) \tag{6.38}$$

forming the measurement matrix $H_k \in \mathbb{R}^{2n_k \times 7}$. The last quantity needed is the measurement covariance matrix stacked like

$$R_k = \begin{pmatrix} R_{k,1} & 0 & \cdots & 0 \\ 0 & R_{k,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{k,n_k} \end{pmatrix} \tag{6.39}$$

forming a quadratic matrix $R_k \in \mathbb{R}^{2n_k \times 2n_k}$. Now those quantities can be used to form an EKF, using (2.20)-(2.22), (2.25) and (2.17)-(2.19). The spline EKF algorithm [1] is summarized within the pseudocode given in Table 1.

Table 1: The spline EKF algorithm

1: **function** Spline EKF
2: Input: $\hat{\boldsymbol{x}}_{k-1|k-1}, P_{k-1|k-1}, \boldsymbol{Z}_k$
3: **step 1: prediction step system state**
4: calculate transition matrix $F_k = \nabla_{\boldsymbol{x}^T} f(\boldsymbol{x})|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k-1|k-1}}$
5: predict $\hat{\boldsymbol{x}}_{k|k-1} = f\left(\hat{\boldsymbol{x}}_{k-1|k-1}\right)$
6: predict $P_{k|k-1} = F_k \cdot P_{k-1|k-1} \cdot F_k^T + Q_k$
7: **step 2: prediction step measurements**
8: **for** $i = 1, \dots, n_k$ **do**
9: measurement in local coordinates $\hat{\boldsymbol{z}}_i = R_{\varphi_{k|k-1}}^{-1} \cdot \left(\boldsymbol{z}_i - \boldsymbol{m}_{k|k-1}\right)$
10: calculate angle $\delta_i = \arctan\left(\frac{\hat{z}_{y,i}}{\hat{z}_{x,i}}\right)$
11: calculate scaled basis points $P^C = S_{k|k-1}^C \cdot P$
12: get active basis points $P_{\tau,i}$ and corresponding $k(\tau_i)$ using δ_i and P^C
13: calculate $(\boldsymbol{a}_i, \boldsymbol{b}_i, \boldsymbol{c}_i) = P_{\tau,i} \cdot M$
14: calculate $u_{q,i} = \det\left((\boldsymbol{q}_i, \hat{\boldsymbol{z}}_i)\right)$
15: **if** $u_{a,i} = 0$
16: $\tau_i' = -\frac{u_{c,i}}{u_{b,i}}$
17: **else**
18: $\tau_i' = \frac{-\sqrt{u_{b,i}^2 - 4 \cdot u_{a,i} \cdot u_{c,i}} - u_{b,i}}{2 \cdot u_{a,i}}$
19: **end if**
20: calculate walk parameter $\tau_i = \tau_i' + k(\tau_i)$
21: calculate predicted measurement $\boldsymbol{y}_i = \boldsymbol{m}_{k|k-1} + R_{\varphi_{k|k-1}} \cdot S_{k|k-1}^C \cdot C(\tau_i)$
22: **end for**
23: **step 3: predicted measurement derivation**
24: **for** $i = 1, \dots, n_k$ **do**
25: contour derivative $\frac{\partial C(\tau_i)}{\partial \tau} = (\boldsymbol{a}_i, \boldsymbol{b}_i, \boldsymbol{c}_i) \cdot (2\tau_i', 1, 0)^T$
26: measurement in local coordinates derivatives $\frac{\partial \hat{\boldsymbol{z}}_i}{\partial \boldsymbol{m}} = R_{\varphi_{k|k-1}}^{-1}$
27: $\frac{\partial \hat{\boldsymbol{z}}_i}{\partial \varphi} = \frac{\partial R_{\varphi_{k|k-1}}^{-1}}{\partial \varphi} \cdot (\boldsymbol{z}_i - \boldsymbol{m}_{k|k-1})$
28: **if** $u_{a,i} = 0$
29: $u_{b,i}^2 \cdot \frac{\partial \tau_i}{\partial g} = \left(\boldsymbol{b}_i \cdot u_{c,i} - \boldsymbol{c}_i \cdot u_{b,i}\right) \times \frac{\partial \hat{\boldsymbol{z}}_i}{\partial g}$
30: **else**
31: $-2u_{a,i}^2 \Theta_{\mathrm{i}} \frac{\partial \tau_i}{\partial g} = (\boldsymbol{a}_i\left(2u_{a,i}u_{c,i} - u_{b,i}\Theta_{\mathrm{i}} - u_{b,i}^2\right) + \boldsymbol{b}_i(u_{b,i}u_{a,i} + u_{a,i}\Theta_{\mathrm{i}}) - 2u_{a,i}^2 \boldsymbol{c}_i) \times \frac{\partial \hat{\boldsymbol{z}}_i}{\partial g}$
32: **end if**
33: derive predicted measurement $\frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{m}} = I_2 + R_{\varphi_{k|k-1}} \cdot S_{k|k-1}^C \cdot \frac{\partial C(\tau_i)}{\partial \tau} \cdot \frac{\partial \tau_i}{\partial \boldsymbol{m}}$

34: $\frac{\partial \boldsymbol{y}_i}{\partial \varphi} = \frac{\partial R_{\varphi_{k|k-1}}}{\partial \varphi} \cdot S^C_{k|k-1} \cdot C(\tau_i) + R_{\varphi_{k|k-1}} \cdot S^C_{k|k-1} \cdot \frac{\partial C(\tau_i)}{\partial \tau} \cdot \frac{\partial \tau_i}{\partial \varphi}$

35: $\frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{x}_{shape}} = R_{\varphi_{k|k-1}} \cdot \mathrm{diag}\big(C(\tau_i)\big)$

36: put derivatives together $\frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{x}_k} = \left(\frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{m}}, 0, \frac{\partial \boldsymbol{y}_i}{\partial \varphi}, 0, \frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{x}_{shape}}\right)$

37: **end for**

38: **step 4: get stacked quantities**

39: measurements $\boldsymbol{z}_k = (\boldsymbol{z}_{k,1}^T; \boldsymbol{z}_{k,2}^T; \dots; \boldsymbol{z}_{k,n_k}^T)$

40: predicted measurements $\boldsymbol{y}_k = (\boldsymbol{y}_{k,1}^T; \boldsymbol{y}_{k,2}^T; \dots; \boldsymbol{y}_{k,n_k}^T)$

41: measurement matrix $H_k = \left(\frac{\partial \boldsymbol{y}_{k,1}}{\partial \boldsymbol{x}_k}; \frac{\partial \boldsymbol{y}_{k,2}}{\partial \boldsymbol{x}_k}; \dots; \frac{\partial \boldsymbol{y}_{k,n_k}}{\partial \boldsymbol{x}_k}\right)$

42: measurement covariance matrix $R_k = \begin{pmatrix} R_{k,1} & 0 & \cdots & 0 \\ 0 & R_{k,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{k,n_k} \end{pmatrix}$

43: **step 5: EKF update**

44: innovation covariance matrix $S_k = H_k \cdot P_{k|k-1} \cdot H_k^T + R_k$

45: Kalman gain $K_k = P_{k|k-1} \cdot H_k \cdot S_k^{-1}$

46: update system state $\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + K_k \cdot (\boldsymbol{z}_k - \boldsymbol{y}_k)$

47: update covariance matrix $P_{k|k} = P_{k|k-1} - K_k \cdot H_k \cdot P_{k|k-1}$

48: Output: $\hat{\boldsymbol{x}}_{k|k}, P_{k|k}$

49: **end function**

The implementation and the performance of the spline EKF, as well as the comparison with the rectangular shape estimator is presented in the next chapter, while the last section in this chapter is about to integrate the spline measurement modal in the PHD filter framework.

6.5 The spline PHD filter

To integrate the spline measurement model [1] in the GM-PHD filter framework for extended objects [35], the equations of section 4.4 and 4.5 need to be used. Additionally, the stacked derivatives of (6.36)-(6.39) are considered as well. The prediction step of the spline PHD filter is the same as the prediction step of the PHD filter for point objects. Therefore, the modeled birth RFS is given using (4.23), while the prediction of the existing GM components is calculated using (4.26)-(4.29). As the spline measurement model is a measurement model for extended objects, (4.40)-(4.57) need to be used to calculate the updated GM components. As stacked derivatives of (4.44)-(4.48) the ones presented in (6.36)-(6.39) need to be used for every GM component in every cell in every partition. As the update step considers every cell in every partition to update every predicted GM component, the number of updated GM components, calculated in (4.50)-(4.52), can grow very fast. To reduce this amount a merging and pruning step [29] is used after the update step. In the pruning step every GM component with a weight smaller than a specific threshold T is ignored to cut off every component under a certain threshold. The merging step is used to merge the GM components with closely spaced means, with the distance calculated using the Mahalanobis distance. With a given set of pruned GM components $\left\{\omega_{k|k}^{(i)}, \hat{\boldsymbol{x}}_{k|k}^{(i)}, P_{k|k}^{(i)}\right\}_{i=1}^{J_T}$, the first step is to compute the component with the biggest weight $\omega_{k|k}^{(j)}$.

Afterwards, the corresponding mean $\hat{\boldsymbol{x}}_{k|k}^{(j)}$ is used to find the closely spaced components. Therefore, a second threshold U is introduced [29]. The indices of the closely spaced components are calculated using

$$\boldsymbol{L} \coloneqq \left\{ i \in \{1, \dots, J_T\} \middle| \left(\hat{\boldsymbol{x}}_{k|k}^{(i)} - \hat{\boldsymbol{x}}_{k|k}^{(j)}\right)^T {P_{k|k}^{(i)}}^{-1} \left(\hat{\boldsymbol{x}}_{k|k}^{(i)} - \hat{\boldsymbol{x}}_{k|k}^{(j)}\right) \leq U \right\}. \tag{6.40}$$

After computing the closely spaced components, those are merged using the following equations:

$$\widetilde{\omega}_{k|k}^{(l)} = \sum_{i \in \boldsymbol{L}} \omega_{k|k}^{(i)}, \tag{6.41}$$

$$\widetilde{\boldsymbol{x}}_{k|k}^{(l)} = \frac{1}{\widetilde{\omega}_{k|k}^{(l)}} \sum_{i \in \boldsymbol{L}} \omega_{k|k}^{(i)} \cdot \hat{\boldsymbol{x}}_{k|k}^{(i)}, \tag{6.42}$$

$$\tilde{P}_{k|k}^{(l)} = \frac{1}{\widetilde{\omega}_{k|k}^{(l)}} \sum_{i \in \boldsymbol{L}} \omega_{k|k}^{(i)} \left(P_{k|k}^{(i)} + \left(\widetilde{\boldsymbol{x}}_{k|k}^{(l)} - \hat{\boldsymbol{x}}_{k|k}^{(i)}\right) \cdot \left(\widetilde{\boldsymbol{x}}_{k|k}^{(l)} - \hat{\boldsymbol{x}}_{k|k}^{(i)}\right)^T\right). \tag{6.43}$$

Finally, a second pruning step can be performed by taking the $J_{\max}$ GM components with the largest merged weights. Thus, the limitations of used hardware can be considered. After the merging and pruning steps the state extraction can be done by simply taking the GM components with weights larger than 0.5 as states of the objects actually being present.

As LIDAR measurements enable the edge visibility of the measured object a DBSCAN algorithm [37] is a possible choice as cluster algorithm for the spline PHD filter. The algorithm clusters closely spaced measurements if the number of measurements is larger than a predefined minimum point threshold $\mathrm{minPoints}$. Remaining measurements are clustered in a noise set. Two points are closely spaced, if the distance between the points is smaller than the predefined parameter ε. Since the LIDAR measurements are close to each other if the object is close to the sensor and vice versa further apart if the object and sensor are further apart, a set of distance parameters $\boldsymbol{\varepsilon} = \{\varepsilon_1, \dots, \varepsilon_{J_P}\}$ has to be used to compute a reasonable subset of all possible partitions of the measurement set.

Computing the rate of the Poisson distributed random number modeling the amount of measurements generated from a specific object $\gamma\left(\hat{\boldsymbol{x}}_{k|k-1}^{(j)}\right)$, can be done by computing the intersections of the LIDAR sensors lines of sight and the objects' contour. Using the position $\boldsymbol{m}_{k|k-1}^{(j)}$, orientation $\varphi_{k|k-1}^{(j)}$ and extension provided by $s_{x,k|k-1}^{(j)}$ and $s_{y,k|k-1}^{(j)}$, the objects' contour can be computed. The LIDAR sensor can be modeled using lines of sight arranged with the given resolution of the sensor. A measurement can occur if a line of sight intersects with the contour of the object. For the sake of simplicity, the spline contour is approximated as rectangle. The edge detection used for the computation of the rate modeling the generated measurements is illustrated in Figure 6.3. The amount of edge detections can be taken as rate of object generated measurements $\gamma\left(\hat{\boldsymbol{x}}_{k|k-1}^{(j)}\right)$.

When computing the partition weights according to (4.56), the product of the cell weights can get that large a numerical overflow can occur. Therefore, the cell weights can be stored using the logarithm of the cell weights. An explanation is given in [38] and the equations are given in the all in all spline PHD filter algorithm.

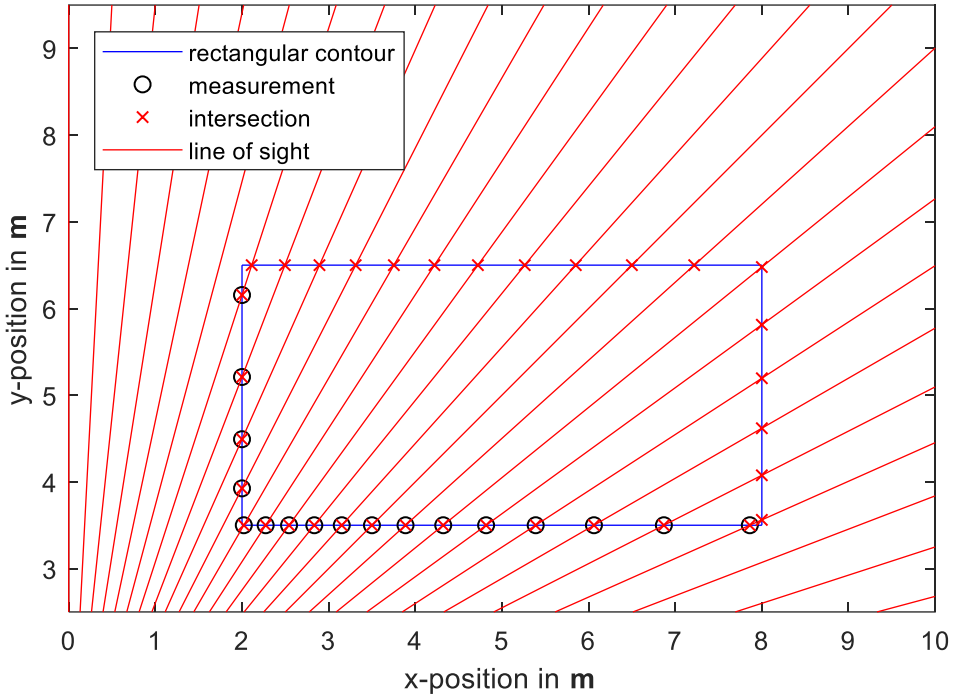

Figure 6.3: Illustration of the edge detection used for the spline PHD filter

The summary of the spline PHD filter, where the spline measurement model is integrated in a GM-PHD filter framework for extended objects, is as pseudocode of Table 2.

Table 2: The spline PHD filter algorithm

1: **function** Spline PHD
2: Input: GM components $\left\{\omega_{k-1\|k-1}^{(j)}, \hat{\boldsymbol{x}}_{k-1\|k-1}^{(j)}, P_{k-1\|k-1}^{(j)}\right\}_{j=1}^{J_{k-1\|k-1}}$, measurement set $\boldsymbol{Z}_k$, set of clustering parameters $\boldsymbol{\varepsilon} = \left\{\varepsilon_p\right\}_{p=1}^{J_P}$
3: **step 1: prediction step**
4: **step 2: compute measurement partitions**
5: **step 3: compute rate of target generated measurements**
6: **step 4: update step**
7: **step 5: merging and pruning**
8: **step 6: state extraction**
9: Output: GM components $\left\{\omega_{k\|k}^{(j)}, \hat{\boldsymbol{x}}_{k\|k}^{(j)}, P_{k\|k}^{(j)}\right\}_{j=1}^{J_{k\|k}}$, extracted object states $\hat{\boldsymbol{X}}_{k\|k}$
10: **end function**

In the following tables the individual steps, specified in Table 2, are given. The prediction step [29] is given in Table 3, the measurement partitioning [37] in Table 4, computing the amount of target generated measurements in Table 5, the update step [35] [1] in Table 6 and the merging and pruning step [29], as well as the state extraction [29], in Table 7 and Table 8 respectively.

Table 3: The spline PHD filter prediction step

1: **function** prediction of birth components

2: Input: GM components $\left\{\omega_{k-1|k-1}^{(j)}, \hat{\boldsymbol{x}}_{k-1|k-1}^{(j)}, P_{k-1|k-1}^{(j)}\right\}_{j=1}^{J_{k-1|k-1}}$

3: initialize prediction counter $i = 0$

4: **for** $j = 1, \ldots, J_{b,k}$ **do**

5: increase prediction counter $i = i + 1$

6: predicted components $\omega_{k|k-1}^{(i)} = \omega_{b,k}^{(j)}$, $\hat{\boldsymbol{x}}_{k|k-1}^{(i)} = \hat{\boldsymbol{x}}_{b,k}^{(j)}$, $P_{k|k-1}^{(i)} = P_{b,k}^{(j)}$

7: **end for**

8: **for** $j = 1, \ldots J_{k-1|k-1}$ **do**

9: increase prediction counter $i = i + 1$

10: predict weight $\omega_{k|k-1}^{(i)} = p_S \cdot \omega_{k-1|k-1}^{(j)}$

11: calculate Jacobi matrix $F_k^{(j)} = \nabla_{\boldsymbol{x}^T} f(\boldsymbol{x})|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k-1|k-1}^{(j)}}$

12: predict mean $\hat{\boldsymbol{x}}_{k|k-1}^{(i)} = F_k^{(j)} \cdot \hat{\boldsymbol{x}}_{k-1|k-1}^{(j)}$

13: predict covariance $P_{k|k-1}^{(i)} = F_k^{(j)} \cdot P_{k-1|k-1}^{(i)} \cdot {F_k^{(j)}}^T + Q_k$

14: **end for**

15: predict amount of GM components $J_{k|k-1} = i$

16: Output: predicted GM components $\left\{\omega_{k|k-1}^{(j)}, \hat{\boldsymbol{x}}_{k|k-1}^{(j)}, P_{k|k-1}^{(j)}\right\}_{j=1}^{J_{k|k-1}}$

17: **end function**

Table 4: Measurement partitioning for the spline PHD filter

1: **function** compute measurement partitions

2: Input: measurement set $\boldsymbol{Z}_k$, set of clustering parameters $\boldsymbol{\varepsilon} = \{\varepsilon_p\}_{p=1}^{J_P}$

3: **for** $p = 1, \ldots, J_P$

4: compute partition cells $\left\{\boldsymbol{Z}_k^{\boldsymbol{W}_n^p}\right\}_{n=1}^{|p_p|} = \mathrm{DBSCAN}(\boldsymbol{Z}_k, \varepsilon_p, \mathrm{minPoints})$

5: **end for**

6: Output: measurement partitions $\left\{\left\{\boldsymbol{Z}_k^{\boldsymbol{W}_n^p}\right\}_{n=1}^{|p_p|}\right\}_{p=1}^{J_P}$

7: **end function**

Table 5: Computing amount of target generated measurements for the spline PHD filter

1: **function** compute rate of target generated measurements

2: Input: predicted GM means $\left\{\hat{\boldsymbol{x}}_{k|k-1}^{(j)}\right\}_{j=1}^{J_{k|k-1}}$

3: **for** $j = 1, \ldots, J_{k|k-1}$ **do**

4: $\gamma_k^{(j)} = \gamma\left(\hat{\boldsymbol{x}}_{k|k-1}^{(j)}\right)$

5: **end for**

6: Output: rate of target generated measurements $\left\{\gamma_k^{(j)}\right\}_{j=1}^{J_{k|k-1}}$

7: **end function**

Table 6: Update step for the spline PHD filter

1: **function** update step

2: Input: predicted GM components $\left\{\omega_{k|k-1}^{(j)}, \hat{\boldsymbol{x}}_{k|k-1}^{(j)}, P_{k|k-1}^{(j)}\right\}_{j=1}^{J_{k|k-1}}$, rate of target generated measurements $\left\{\gamma_k^{(j)}\right\}_{j=1}^{J_{k|k-1}}$, measurement partitions $\left\{\boldsymbol{Z}_k^{\boldsymbol{W}_n^p}\right\}_{n=1}^{J_P}$

3: **for** $j = 1, \ldots, J_{k|k-1}$ **do**

4: update weight $\omega_{k|k}^{(j)} = \left(1 - \left(1 - e^{-\gamma_k^{(j)}}\right) p_D\right) \omega_{k|k-1}^{(j)}$

5: update mean $\hat{\boldsymbol{x}}_{k|k}^{(j)} = \hat{\boldsymbol{x}}_{k|k-1}^{(j)}$ and covariance $P_{k|k}^{(j)} = P_{k|k-1}^{(j)}$

6: **end for**

7: initialize cell counter $l = 0$

8: **for** $p = 1, \ldots J_P$ **do**

9: **for** $n = 1, \ldots, |p_p|$ **do**

10: increment cell counter $l = l + 1$

11: select measurement set $\boldsymbol{Z}_n = \boldsymbol{Z}_k^{\boldsymbol{W}_n^p}$

12: **for** $j = 1, \ldots, J_{k|k-1}$ **do**

13: **predict measurements**

14: **for** $m = 1, \ldots, |\boldsymbol{Z}_n|$ **do**

15: predict measurements according to (6.2)-(6.17) using $\boldsymbol{Z}_n$ and $\hat{\boldsymbol{x}}_{k|k-1}^{(j)}$.

16: **end for**

17: Output: $\left\{\tau_m^{(j)}, \boldsymbol{y}_m^{(j)}\right\}_{m=1}^{|\boldsymbol{Z}_n|}, \left\{\left(\boldsymbol{a}_m^{(j)}, \boldsymbol{b}_m^{(j)}, \boldsymbol{c}_m^{(j)}\right)\right\}_{m=1}^{|\boldsymbol{Z}_n|}, \left\{\hat{\boldsymbol{z}}_m^{(j)}\right\}_{m=1}^{|\boldsymbol{Z}_n|}$

18: **predicted measurement derivation**

19: **for** $m = 1, \ldots, |\boldsymbol{Z}_n|$ **do**

20: derive predicted measurements according to (6.18)-(6.35) using $\left\{\tau_m^{(j)}, \boldsymbol{y}_m^{(j)}\right\}_{m=1}^{|\boldsymbol{Z}_n|}, \left\{\left(\boldsymbol{a}_m^{(j)}, \boldsymbol{b}_m^{(j)}, \boldsymbol{c}_m^{(j)}\right)\right\}_{m=1}^{|\boldsymbol{Z}_n|}, \hat{\boldsymbol{x}}_{k|k-1}^{(j)}, \boldsymbol{Z}_n$ and $\left\{\hat{\boldsymbol{z}}_m^{(j)}\right\}_{m=1}^{|\boldsymbol{Z}_n|}$

21: **end for**

22: Output: $\left\{\frac{\partial \boldsymbol{y}_m^{(j)}}{\partial \boldsymbol{x}_k}\right\}_{m=1}^{|\boldsymbol{Z}_n|}$

23: **get stacked quantities**

24: measurements $\boldsymbol{z}_n = \left(\boldsymbol{z}_1^T; \boldsymbol{z}_2^T; \ldots; \boldsymbol{z}_{|\boldsymbol{Z}_n|}^T\right)$

25: predicted measurements $\boldsymbol{y}_n^{(j)} = \left(\boldsymbol{y}_1^{(j)^T}; \boldsymbol{y}_2^{(j)^T}; \ldots; \boldsymbol{y}_{|\boldsymbol{Z}_n|}^{(j)^T}\right)$

26: measurement matrix $H_n^{(j)} = \left(\frac{\partial \boldsymbol{y}_1^{(j)}}{\partial \boldsymbol{x}_k}; \frac{\partial \boldsymbol{y}_2^{(j)}}{\partial \boldsymbol{x}_k}; \ldots; \frac{\partial \boldsymbol{y}_{|\boldsymbol{Z}_n|}^{(j)}}{\partial \boldsymbol{x}_k}\right)$

27: covariance matrix $R_n^{(j)} = \begin{pmatrix} R_1 & 0 & \cdots & 0 \\ 0 & R_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{|\boldsymbol{Z}_n|} \end{pmatrix}$

28: **compute updated components**

29: innovation covariance $S_n^{(j)} = H_n^{(j)} \cdot P_{k|k-1}^{(j)} \cdot H_n^{(j)^T} + R_n^{(j)}$

30: Kalman gain $K_n^{(j)} = P_{k|k-1}^{(j)} \cdot H_n^{(j)^T} \cdot S_n^{(j)^{-1}}$

31: mean $\hat{\boldsymbol{x}}_{k|k}^{(j+J_{k|k-1}l)} = \hat{\boldsymbol{x}}_{k|k-1}^{(j)} + K_n^{(j)} \cdot \left(\boldsymbol{z}_n - \boldsymbol{y}_n^{(j)}\right)$

32: covariance matrix $P_{k|k}^{(j+J_{k|k-1}l)} = P_{k|k-1}^{(j)} - K_n^{(j)} \cdot H_n^{(j)} \cdot P_{k|k-1}^{(j)}$

33: $\Gamma_n^{(j)} = e^{-\gamma^{(j)}} \cdot \left(\gamma^{(j)}\right)^{|\boldsymbol{z}_n|}$

34: $\Phi_n^{(j)} = \mathcal{N}\left(\boldsymbol{z}_n; \boldsymbol{y}_n^{(j)}, S_n^{(j)}\right) \cdot \prod_{m=1}^{|\boldsymbol{z}_n|} \frac{1}{\kappa_k(\boldsymbol{z}_m)}$

35: update weight $\omega_{k|k}^{\left(j+J_{k|k-1}l\right)} = p_D \cdot \Gamma_n^{(j)} \cdot \Phi_n^{(j)} \cdot \omega_{k|k-1}^{(j)}$

36: **end for**

37: compute cell normalization $d_{\boldsymbol{W}_n^p} = \delta_{|\boldsymbol{Z}_n|,1} + \sum_{j=1}^{J_{k|k-1}} \omega_{k|k}^{\left(j+J_{k|k-1}l\right)}$

38: normalize weights $\omega_{k|k}^{\left(j+J_{k|k-1}l\right)} = \frac{\omega_{k|k}^{\left(j+J_{k|k-1}l\right)}}{d_{\boldsymbol{W}_n^p}}$ for $j = 1, \dots, J_{k|k-1}$

39: **end for**

40: compute log partition normalization numerator $\widetilde{\omega}_{p_p} = \sum_{n=1}^{|p_p|} \log\left(d_{\boldsymbol{W}_n^p}\right)$

41: **end for**

42: update amount of GM components $J_{k|k} = J_{k|k-1} \cdot (l+1)$

43: log partition normalization $\widetilde{\omega}_{p_p} = \widetilde{\omega}_{p_p} - \left(\widetilde{\omega}_{p_1} + \log\left(1 + \sum_{p'=2}^{J_P} e^{\widetilde{\omega}_{p_{p'}} - \widetilde{\omega}_{p_1}}\right)\right)$ $p = 1, \dots, J_P$

44: partition normalization factor $\omega_{p_p} = e^{\widetilde{\omega}_{p_p}}$ for $p = 1, \dots, J_P$

45: $J_{\text{aux}} = J_{k|k-1}$

46: **for** $p = 1, \dots, J_P$ **do**

47: weight normalization $\omega_{k|k}^{(j+J_{\text{aux}})} = \omega_{k|k}^{(j+J_{\text{aux}})} \cdot \omega_{p_p}$ for $j = 1, \dots, J_{k|k-1}|p_p|$

48: $J_{\text{aux}} = J_{\text{aux}} + J_{k|k-1} \cdot |p_p|$

49: **end for**

50: Output: updated GM components $\left\{\omega_{k|k}^{(j)}, \hat{\boldsymbol{x}}_{k|k}^{(j)}, P_{k|k}^{(j)}\right\}_{j=1}^{J_{k|k}}$

51: **end function**

Table 7: Merging and pruning step for the spline PHD filter

1: **function** merging and pruning

2: Input: updated GM components $\left\{\omega_{k|k}^{(j)}, \hat{\boldsymbol{x}}_{k|k}^{(j)}, P_{k|k}^{(j)}\right\}_{j=1}^{J_{k|k}}$, thresholds $T, U, J_{\max}$

3: prune small weights $\boldsymbol{I} = \left\{i \in \{1, \dots, J_{k|k}\} \middle| \omega_{k|k}^{(i)} > T\right\}$

4: initialize merged components counter $l = 0$

5: **repeat**

6: increment merged components counter $l = l + 1$

7: find maximum weight $j = \arg\max_{i \in \boldsymbol{I}} \omega_{k|k}^{(i)}$

8: closely spaced components $\boldsymbol{L} = \left\{i \in \boldsymbol{I} \middle| \left(\hat{\boldsymbol{x}}_{k|k}^{(i)} - \hat{\boldsymbol{x}}_{k|k}^{(j)}\right)^T {P_{k|k}^{(i)}}^{-1} \left(\hat{\boldsymbol{x}}_{k|k}^{(i)} - \hat{\boldsymbol{x}}_{k|k}^{(j)}\right) \leq U\right\}$

9: merge weights $\widetilde{\omega}_{k|k}^{(l)} = \sum_{i \in \boldsymbol{L}} \omega_{k|k}^{(i)}$

10: merge means $\widetilde{\boldsymbol{x}}_{k|k}^{(l)} = \frac{1}{\widetilde{\omega}_{k|k}^{(l)}} \sum_{i \in \boldsymbol{L}} \omega_{k|k}^{(i)} \cdot \hat{\boldsymbol{x}}_{k|k}^{(i)}$

11: merge covariance's $\tilde{P}_{k|k}^{(l)} = \frac{1}{\widetilde{\omega}_{k|k}^{(l)}} \sum_{i \in \boldsymbol{L}} \omega_{k|k}^{(i)} \left(P_{k|k}^{(i)} + \left(\widetilde{\boldsymbol{x}}_{k|k}^{(l)} - \hat{\boldsymbol{x}}_{k|k}^{(i)}\right)\left(\widetilde{\boldsymbol{x}}_{k|k}^{(l)} - \hat{\boldsymbol{x}}_{k|k}^{(i)}\right)^T\right)$

12: set derivative $\boldsymbol{I} = \boldsymbol{I} \backslash \boldsymbol{L}$

13: **until** $I = \emptyset$

14: **if** $l > J_{\max}$ **do**

15: replace GM components $\left\{\widetilde{\omega}_{k|k}^{(i)}, \widetilde{\boldsymbol{x}}_{k|k}^{(i)}, \tilde{P}_{k|k}^{(i)}\right\}_{i=1}^{l}$ with the components with $J_{\max}$ largest weights and $l = J_{\max}$

16: **end if**
17: set $\left\{\omega_{k|k}^{(j)}, \hat{\boldsymbol{x}}_{k|k}^{(j)}, P_{k|k}^{(j)}\right\}_{j=1}^{J_{k|k}} = \left\{\tilde{\omega}_{k|k}^{(i)}, \tilde{\boldsymbol{x}}_{k|k}^{(i)}, \tilde{P}_{k|k}^{(i)}\right\}_{i=1}^{l}$
18: Output: merged and pruned updated GM components $\left\{\omega_{k|k}^{(j)}, \hat{\boldsymbol{x}}_{k|k}^{(j)}, P_{k|k}^{(j)}\right\}_{j=1}^{J_{k|k}}$
19: **end function**

Table 8: state extraction for the spline PHD filter

1: **function** state extraction
2: Input: merged and pruned updated GM components $\left\{\omega_{k|k}^{(j)}, \hat{\boldsymbol{x}}_{k|k}^{(j)}, P_{k|k}^{(j)}\right\}_{j=1}^{J_{k|k}}$
3: initialize empty state set $\hat{\boldsymbol{X}}_{k|k} = \emptyset$
4: **for** $j = 1, \dots, J_{k|k}$ **do**
5: **if** $\omega_{k|k}^{(j)} > 0.5$ **do**
6: update state set $\hat{\boldsymbol{X}}_{k|k} = \left(\hat{\boldsymbol{X}}_{k|k}, \hat{\boldsymbol{x}}_{k|k}^{(j)}\right)$
7: **end if**
8: **end for**
9: Output: state set $\hat{\boldsymbol{X}}_{k|k}$
10: **end function**

The parameters used for the implementation as well as the simulation results are given in the next chapter.

7 Performance evaluation

In order to evaluate the performance of the spline extension model within the EKF and the GM-PHD Filter, a self-implemented version of both filters is tested. Since a decoupled error analysis, using root mean square errors for every component of the object state, does not show the impact of one component on the total error, a metric for the performance evaluation of extended objects is introduced in section 7.1. Afterwards, the performance evaluation for tracking one extended object using a spline EKF is illustrated in section 7.2. Within this section the performance is analyzed using simulated measurements. The first subsection 7.2.1 is therefore about the implemented simulation environment to generate the measurements in every time step. In subsection 7.2.2 the results using the spline EKF to track the simulated measurement data are presented. The results for one simulation run and a Monte Carlo simulation with 1000 runs are shown. In subsection 7.2.3 the performance of the spline EKF is compared with the rectangular shape estimator presented in section 3.3. Therefore, also the results for one simulation run and a Monte Carlo simulation with 1000 runs are shown. The performance of the spline PHD filter is analyzed in section 7.4. The performance of this filter is also evaluated with simulated measurements. Therefore, the simulation environment for generating measurements of several extended objects in a cluttered environment is presented in subsection 7.4.1. Finally, the results of the simulation are illustrated in subsection 7.4.2.

7.1 A distance measure for extended objects

The metric used for the performance evaluation in this thesis is the optimal subpattern assignment (OSPA) metric introduced in [39]. The paper describes a consistent metric for the performance evaluation of multi object point tracking problems, where the performance is described in a single number. In contrast to a decoupled analysis using root mean square

errors, the OSPA provides an actual evaluation of the tracker's performance. The metric can be adapted to analyze the performance of a single extended object tracker presented in [40].

In order to calculate the OSPA distance of two extended objects n equidistant points must be chosen from the reference contour $\boldsymbol{x}$ and the estimated contour $\hat{\boldsymbol{x}}$ to approximate a uniform distribution on the boundary.

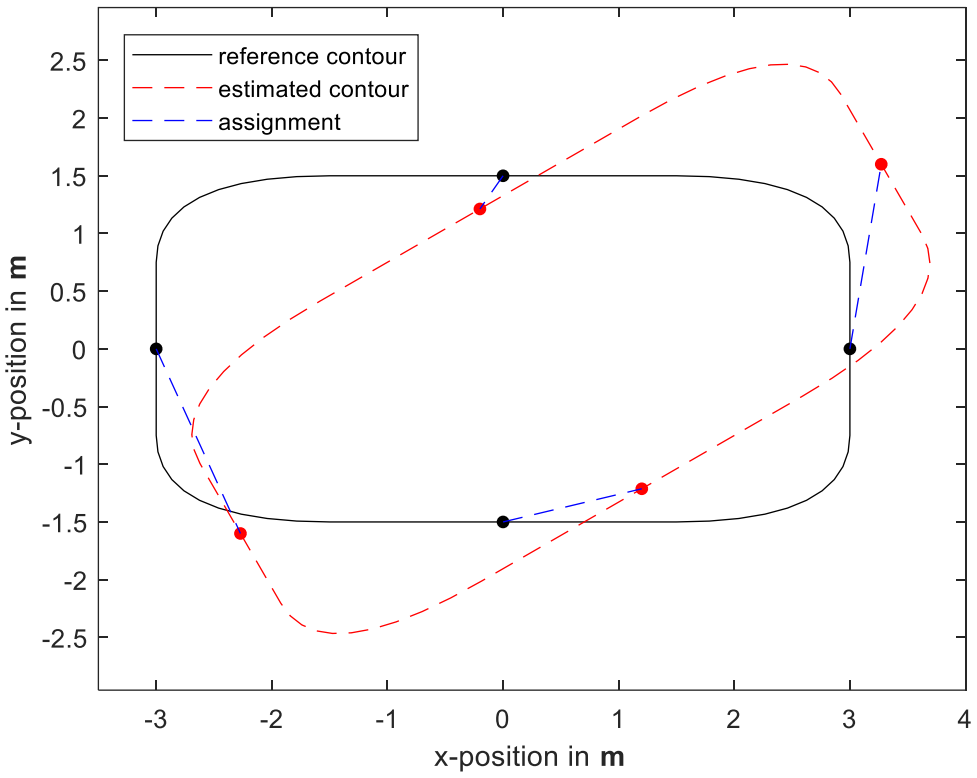

Figure 7.1: Visualization of the OSPA distance for extended objects

The points are denoted as $p_{\boldsymbol{x}} = \{p_{\boldsymbol{x}}^1, p_{\boldsymbol{x}}^2, \dots, p_{\boldsymbol{x}}^n\}$ for the reference contour and $p_{\hat{\boldsymbol{x}}} = \{p_{\hat{\boldsymbol{x}}}^1, p_{\hat{\boldsymbol{x}}}^2, \dots, p_{\hat{\boldsymbol{x}}}^n\}$ for the estimated contour. The OSPA distance for extended objects is then defined as

$$d_{OSPA,n}(p_{\boldsymbol{x}}, p_{\hat{\boldsymbol{x}}})^p = \min_{\pi \in \Pi} \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left\| p_{\boldsymbol{x}}^i - p_{\hat{\boldsymbol{x}}}^{\pi(i)} \right\|^p} \tag{7.1}$$

with Π as the set of all permutations of $\{1,2,\dots,n\}$ and π as one permutation in the set of permutations. The OSPA metric therefore searches for the minimal sum of distances where each point on the reference contour is assigned to one point on the estimated contour. The OSPA distance is illustrated in Figure 7.1 with $n = 4$ and the optimal assignment of the two sets of points. Finding the minimum costs for the assignment problem is a common problem in combinatorial optimization and can be solved using the Hungarian method, also called Kuhn-Munkres algorithm [41], or the auction algorithm [42]. An implementation of the Kuhn-Munkres algorithm of [43] is used for the OSPA calculation in this thesis.

7.2 Performance of the spline EKF

This section is about to evaluate the performance of the spline EKF tracker presented in Table 1. In order to provide a well-defined object state in every time step, (6.1) is used to define the system state vector. As system transition model a coordinated turn model with polar velocity [8] combined with an assumed constant extension is used. The transition function is then given as

$$f(\boldsymbol{x}_k) = \begin{pmatrix} x_k + \frac{2v_k}{\omega_k}\sin\left(\frac{\omega_k t}{2}\right)\cos\left(\varphi_k + \frac{\omega_k t}{2}\right) \\ y_k + \frac{2v_k}{\omega_k}\sin\left(\frac{\omega_k t}{2}\right)\sin\left(\varphi_k + \frac{\omega_k t}{2}\right) \\ v_k \\ \varphi_k + \omega_k t \\ \omega_k \\ s_{x,k} \\ s_{y,k} \end{pmatrix} \tag{7.2}$$

with the periodic time t of the sensor. The system covariance matrix is calculated as

$$Q = G \cdot \mathrm{diag}\left(\left(\sigma_v^2, \sigma_\omega^2, \sigma_{s_x}^2, \sigma_{s_y}^2\right)\right) \cdot G^T \tag{7.3}$$

with G given as

$$G = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & t & 0 & 0 \\ 0 & 0 & t & 0 \\ 0 & 0 & 0 & t \end{pmatrix}. \tag{7.4}$$

The system standard deviations are specified as $\sigma_v = 2.2m/s$, $\sigma_\omega = 1.5°/s$, $\sigma_{s_x} = 0.1m$ and $\sigma_{s_y} = 0.1m$. One single measurement covariance matrix used for the combination to the measurement covariance matrix specified in (6.39) is given as

$$R = \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\vartheta^2 \end{pmatrix} \tag{7.5}$$

with the measurement standard deviations specified as $\sigma_r = 0.1m$ and $\sigma_\vartheta = 0.5°$. The measurement set is therefore given as

$$\boldsymbol{Z}_k = \begin{pmatrix} r_1 & r_2 & \dots & r_{n_k} \\ \vartheta_1 & \vartheta_2 & \dots & \vartheta_{n_k} \end{pmatrix} \tag{7.6}$$

with the range r and the azimuth ϑ for every measurement in polar coordinates.

7.2.1 Simulation environment

Within the simulation environment to evaluate the performance of the spline EKF, the measurements for one moving object need to be generated. The LIDAR sensor is modeled using several lines of sight going through the point where the sensor is located. Those lines of

sight are generated using a specific angle as resolution of the sensor which can be adjusted within the simulation environment.

In order to generate the shape measurements, the first step is to create a random trajectory using (7.2) without the extension components and adding Gaussian random noise terms using σ_v and σ_ω. Given the position and the orientation of the trajectory at every time step those quantities can be used to create a rectangle within the surveillance area.

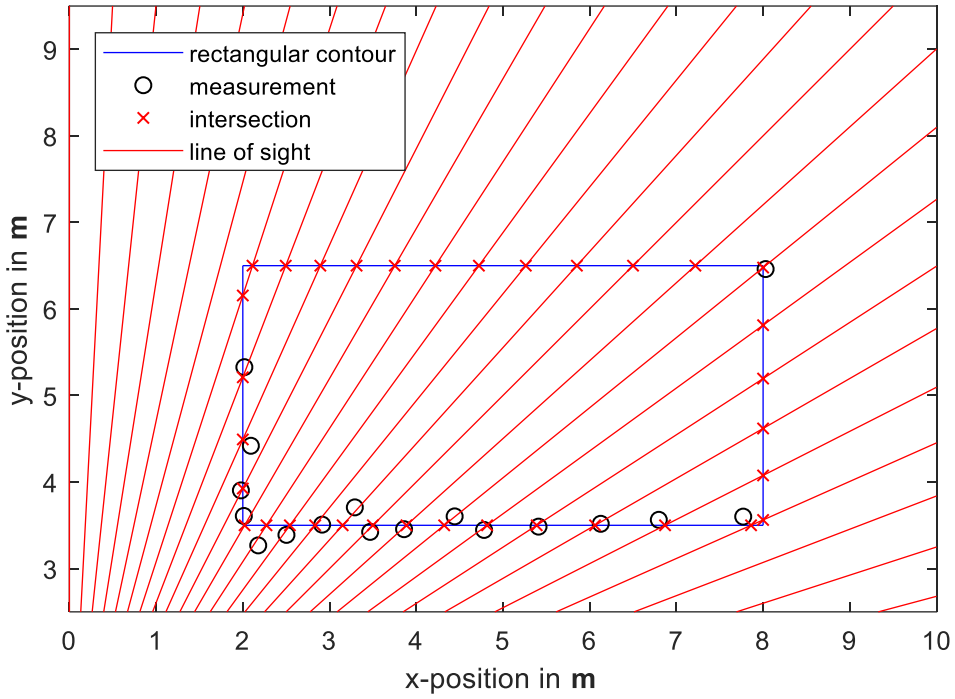

Figure 7.2: Illustration of the edge detection used for the EKF simulation environment

With a specific model length and width, the corners of the rectangle can be calculated. The analytical representation of the lines of sight and the edges of the model allows to calculate their intersections. Those intersections are illustrated in Figure 7.2 and are now used to generate the shape measurements. Since the sensor can only see the front facing it and one line of sight has two intersections with the rectangle, it is necessary to find out which intersection is located closer to the sensor. The closer located intersection is added to a data set if a uniform distributed random number is smaller than the predefined probability of detection p_D. The more distant intersection is added to a data set if a uniform distributed random number is smaller than the predefined probability of multipath detection p_{MD}. This data set now contains noiseless shape measurements in Cartesian coordinates. In order to calculate the final measurement set, those data points need to be given in polar coordinates. The last step is to add a random Gaussian noise term to the range using σ_r and the azimuth using σ_ϑ of each data point.

7.2.2 Simulation results

The simulations are done with a sensor resolution of one degree and a detection probability of $p_D = 0.95$. The first results presented are the root mean square errors and the OSPA distance for a random single run. Within this simulation the probability of multipath detection

is set to $p_{MD} = 0$ and the periodic time of the sensor is set to $t = 0.1s$. The OSPA distance is calculated using $n = 50$ samples both of the contour of the estimation and the reference. In Figure 7.3 the root mean square errors for the position in x and y dimension, the orientation and the scale factors in x and y dimension for one run are shown. The figure shows the accuracy of the filter for every component separately.

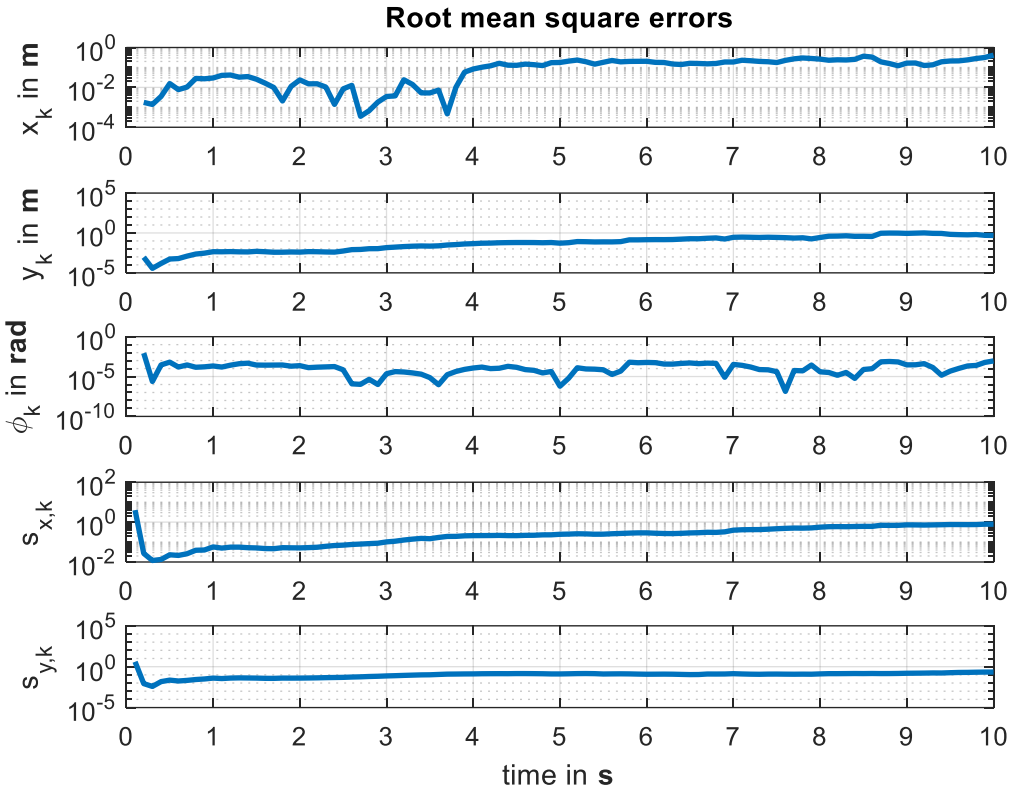

Figure 7.3: Root mean square errors for one simulation run

While the orientation error stays nearly constant, all the other errors increase over time. The reason for that can be seen in Figure 7.4, where the OSPA distance as well as the number of measurements are illustrated for the same simulation run. The number of measurements first increases at the beginning of the simulation and then decreases rapidly at the ending. The trajectory therefore starts near the sensor, then leads past the sensor, where the maximum number of measurements is, and then diverges from the sensor, where the number of measurements decreases. The deterioration of the performance can also be detected in the OSPA distance, which also increases over time. However, the deterioration in performance is not linearly related to the number of measurements, which is a positive property. The value itself of the OSPA distance is difficult to interpret since the position, the orientation and the extension are included in the distance. Since the RMSE values of the simulation run show an accurate performance of the filter, the OSPA distance seems to be in an acceptable range too. However, to interpret the value of the OSPA distance the best it needs to be compared to either another simulation run of the spline EKF or the simulation of another filter. Both comparisons will be shown within this chapter.

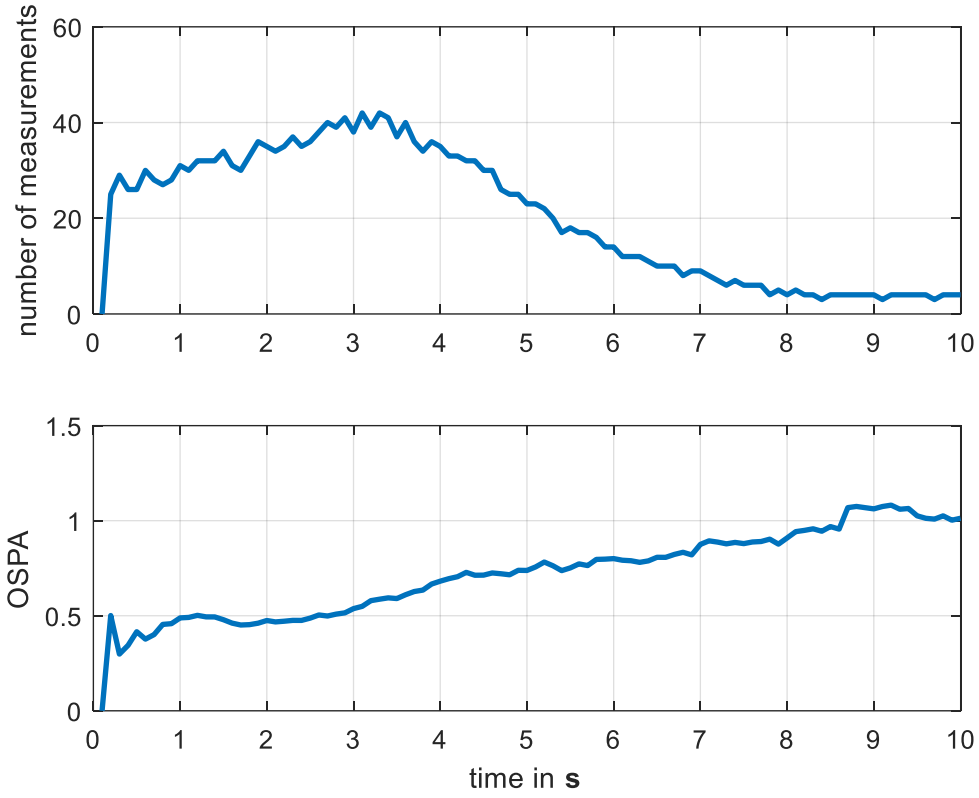

Figure 7.4: OSPA distance and number of measurements for one simulation run

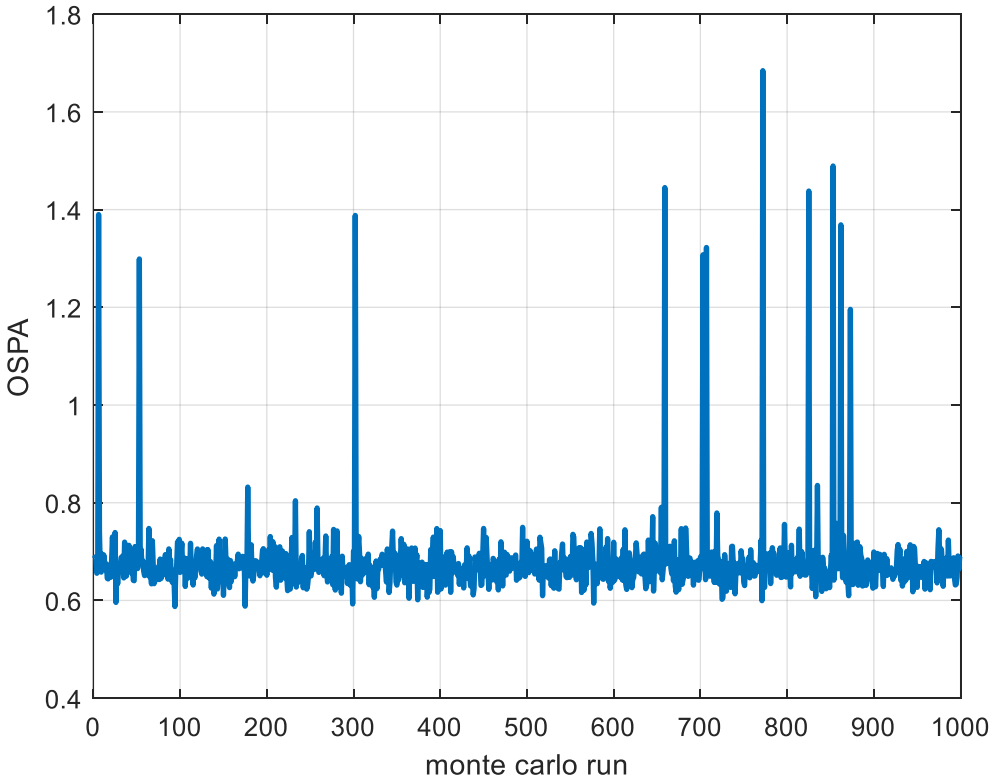

Figure 7.5: OSPA distance of the first Monte Carlo simulation of the spline EKF

In order to show a long-term performance of the spline EKF the results of a Monte Carlo simulation with 1000 runs is shown. In Figure 7.5 the mean OSPA distance of every simulation run is shown, while Figure 7.6 shows the mean root mean square errors of every run. The simulation is done using the same uncertainties, probabilities, simulation time and OSPA samples as stated above.

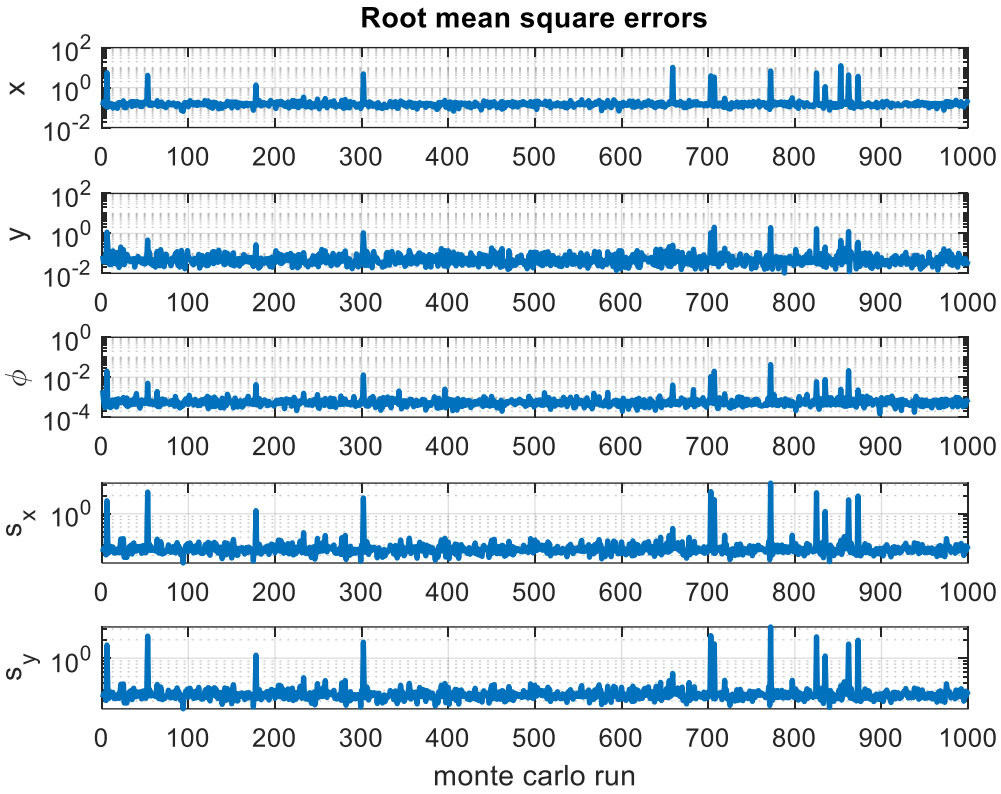

Figure 7.6: RMSE values of the first Monte Carlo simulation of the spline EKF

The figures show a constant performance for most of the runs. However, several simulations show a worse result than most of the others. The reason for that is up to the initialization of the spline EKF in the run. In the case where the measurements are assigned to the wrong side of the car within the initialization step the covariance matrix P of the filter gets too small, to correct the wrong assignment throughout the simulation. To correct this only a new initialization can help. The remaining runs show a stable performance of the spline EKF tracker. Most of the mean values of the OSPA are between 0.6 and 0.8 were also the mean OSPA distance of Figure 7.4 is located.

In order to improve the performance of the spline EKF the detection and multipath detection probabilities are set to $p_D = 0.95$ and $p_{MD} = 0.05$ in a second Monte Carlo simulation. Also, in this simulation 1000 runs are done. The results are shown in the mean values of the OSPA distance and the RMSE values in Figure 7.7 and Figure 7.8 respectively. The figures show a much better performance than the results in the previous Monte Carlo simulation. Most of the mean OSPA values are lower than before and the RMSE values show a better performance as well. However, these results must be treated with caution, since there is no guarantee that there is a single multipath detection in a specific scenario. In summary the results show a very good performance of the spline EKF filter apart from the runs, where the initialization fails. In those cases, only a new initialization leads to a better performance.

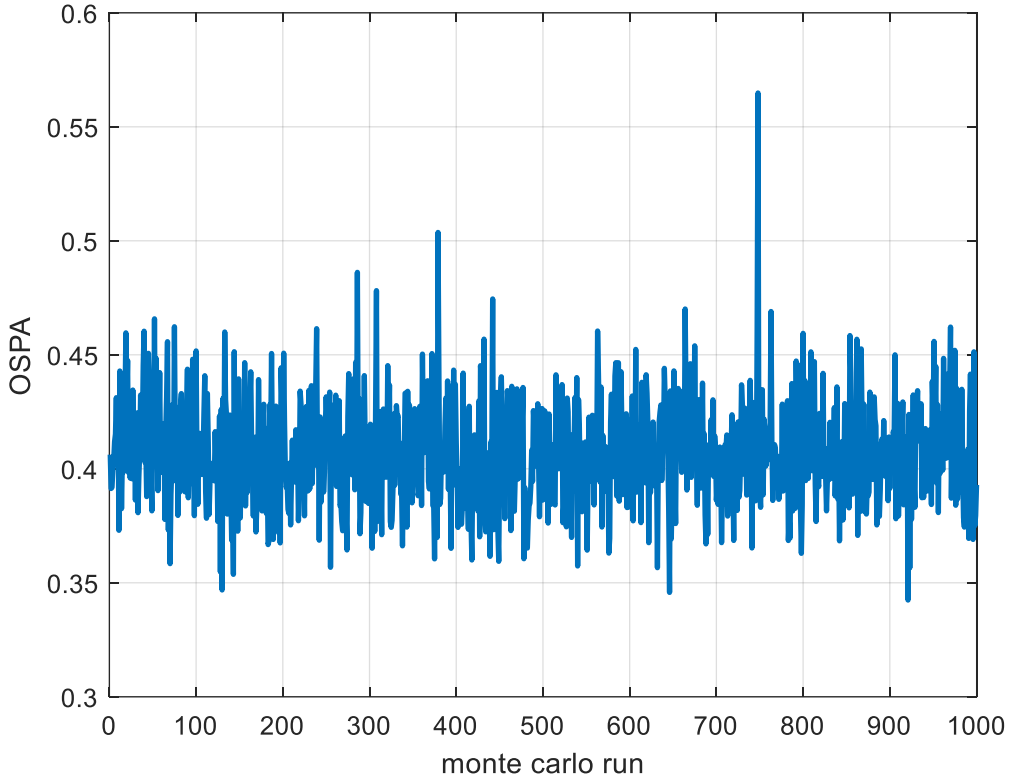

Figure 7.7: OSPA distance of the second Monte Carlo simulation of the spline EKF

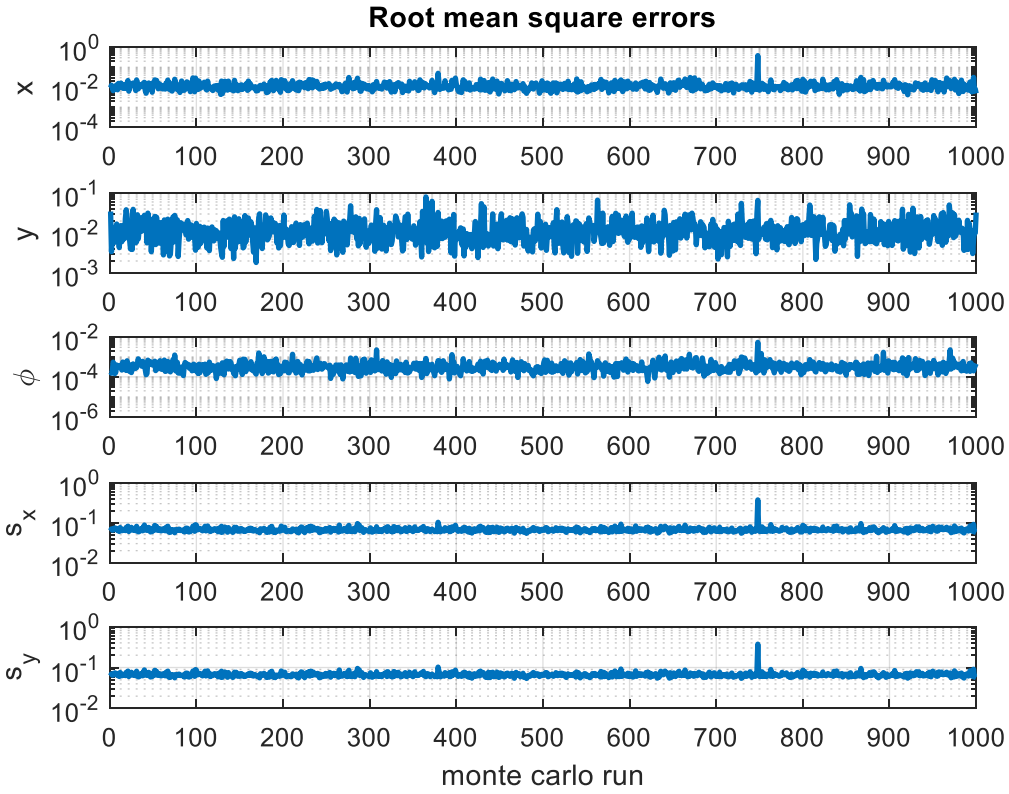

Figure 7.8: RMSE values of the second Monte Carlo simulation of the spline EKF

7.2.3 Performance comparison

Within this subsection the spline EKF tracker is compared to the rectangular shape estimator presented in section 3.3. Therefore, the same simulation environment presented in subsection 7.2.1 is used. First the simulation results of a single run are illustrated in Figure 7.9 and Figure 7.10 with the mean root mean square errors and the OSPA distances for every time step respectively. The second figure also shows the number of measurements per time step.

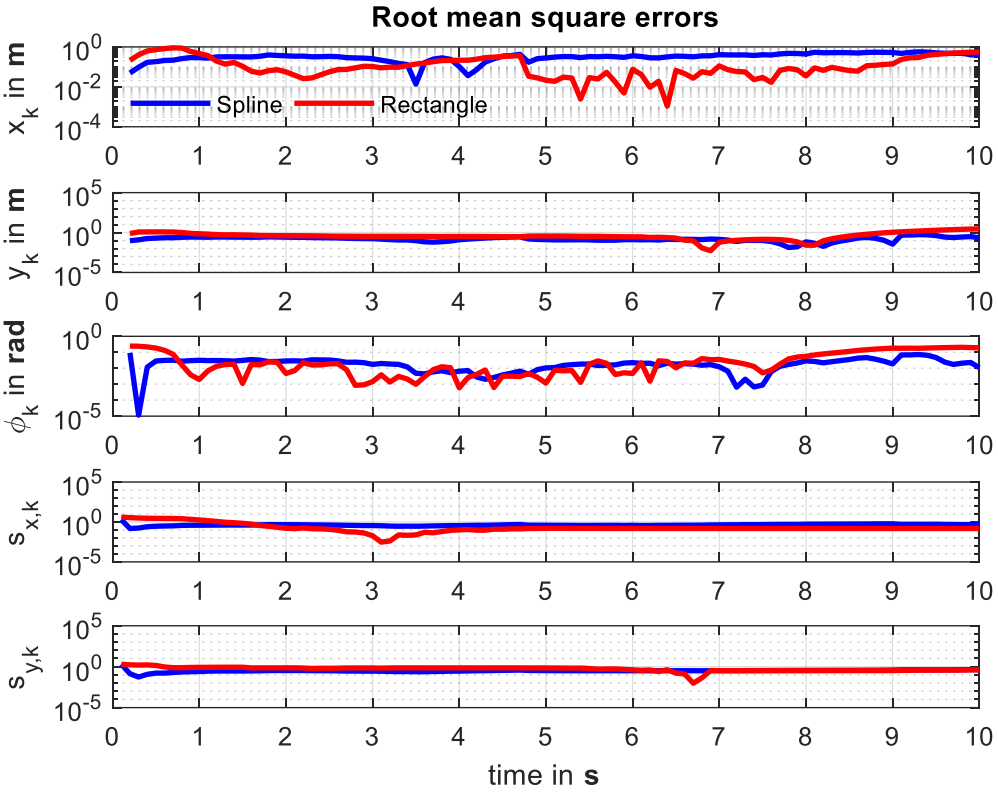

Figure 7.9: Compared RMSE values of a single run

The simulation is done with the same uncertainties, simulation time and OSPA samples as stated above. The probability of detection and multipath detection probability are set to $p_D = 1$ and $p_{MD} = 0$. The figure showing the root mean square errors is hard to interpret, since it shows a decoupled error consideration. The tendency is that the spline EKF is more stable than the rectangular shape estimator. At some points the spline EKF and at other points the rectangular shape estimator shows a better performance in a specific component of the system state, but the overall performance of the filters cannot be picked out easily. Therefore, the second illustration showing the OSPA distance with the number of measurements is more suitable. The scenario simulated is a drive-by of the car like in the subsection before, so the number of measurements increase in the beginning until the maximum number is reached and decreases as the vehicle moves away. The OSPA distances show the overall performance of the filters. In the beginning the rectangular estimator is much worse than the spline EKF, since the rectangular estimator needs quite some time steps in order to estimate the vehicles extension accurately. Afterwards, the rectangular estimator can get better than the spline EKF but is again getting worse as the number of measurements decreases. The results of a single run show the stability of the spline EKF tracker in contrast to the rectangular shape estimator. This can be better in some situations but is much worse in the first time steps and can completely loose the track with a decreasing number of measurements in the worst case.

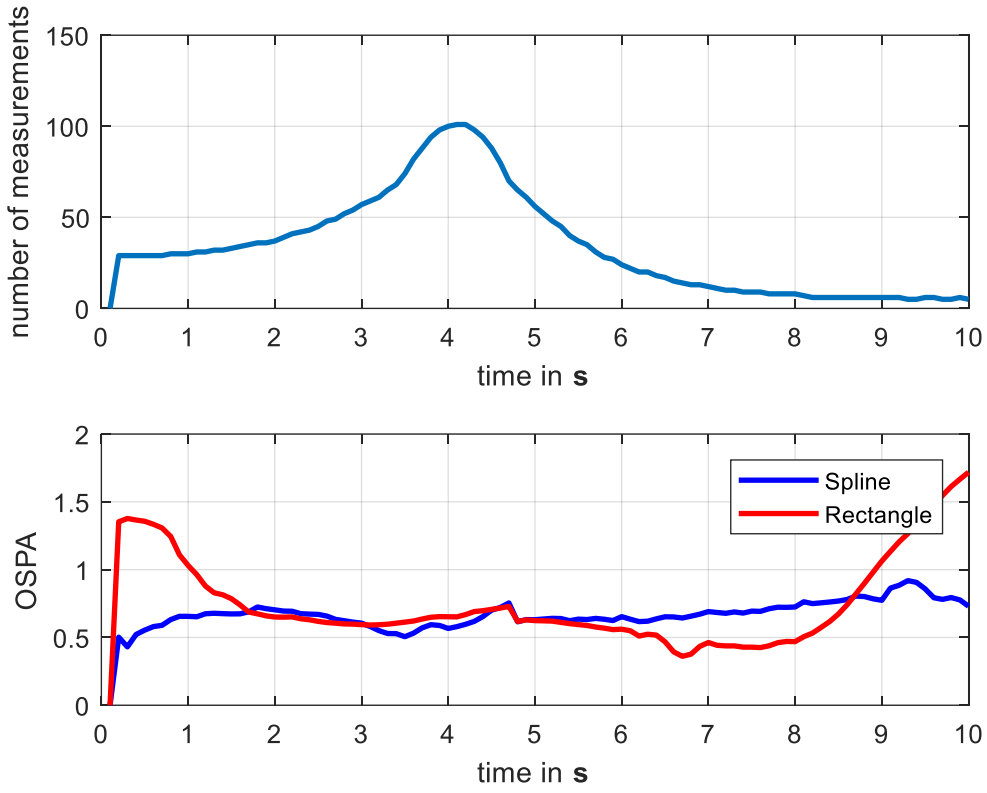

Figure 7.10: Compared OSPA distance of a single run

In order to verify the first impression, the results of two Monte Carlo simulations are illustrated in the following. Therefore, the same scenario as above with the same settings except the resolution of the sensor and the two probabilities is used for every run. The probability of detection is set to $p_D = 0.95$ and the multipath detection probability is set to $p_{MD} = 0.05$. In the first simulation a resolution of two degrees is used. In Figure 7.11 the mean OSPA distance, the total simulation time and the mean number of measurements for every time step are shown. The results of this Monte Carlo simulation show the outperformance of the spline EKF tracker towards the rectangular shape estimator in this scenario. The OSPA distance shows a much better and more constant performance of the spline EKF. Also, the total simulation time is mostly shorter with the implementations used. The mean number of measurements is under 20 measurements per time step and the spline EKF tracker is still showing a good performance. In order to achieve a similar performance between the spline EKF tracker and the rectangular estimator, the resolution of the sensor needs to be set to 0.5 degrees. The results of this simulation are not shown here.

To outperform the spline EKF the resolution needs to be set to 0.1 degrees. The results of this Monte Carlo simulation are shown in Figure 7.12. The OSPA distance of both trackers get much more constant and the rectangular estimator shows a better performance and gets faster as the spline EKF, which also still shows a good performance. The mean number of measurements grows up to 250 to 500 measurements per time step. A remarkable result of those simulations is the constancy of the spline EKF tracker, which nearly shows the same results with around 20 measurements per time step as up to several hundreds of measurements per time step. On the other side the rectangular estimator gets better and finally outperforms the spline EKF with an increasing number of measurements.

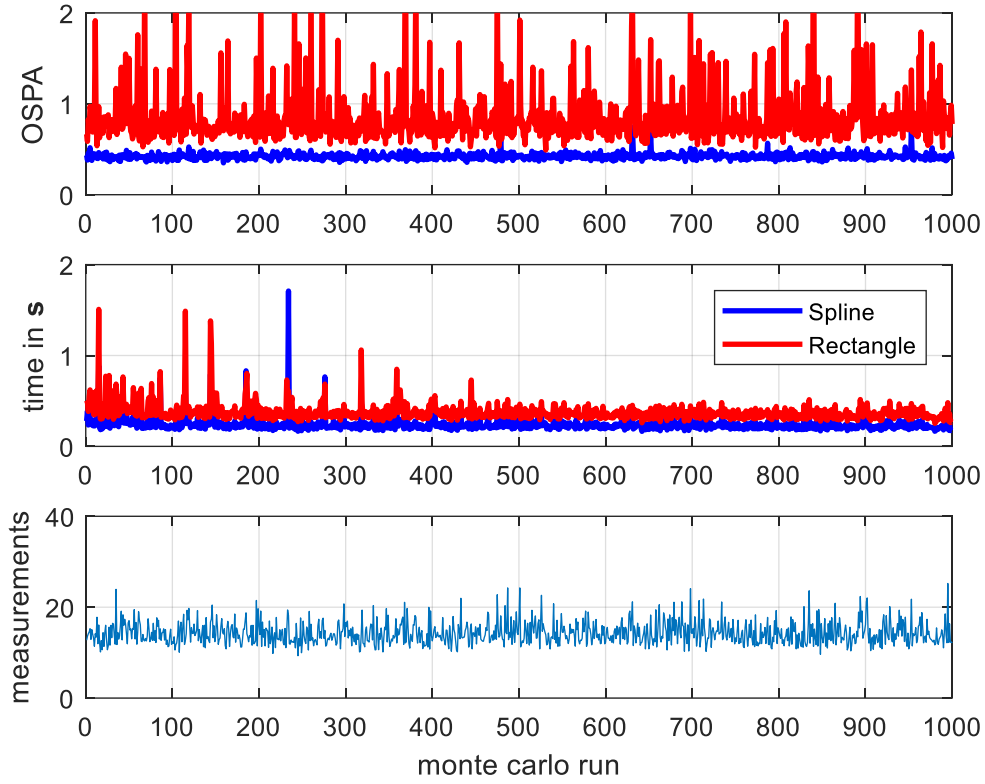

Figure 7.11: First Monte Carlo simulation for the comparison

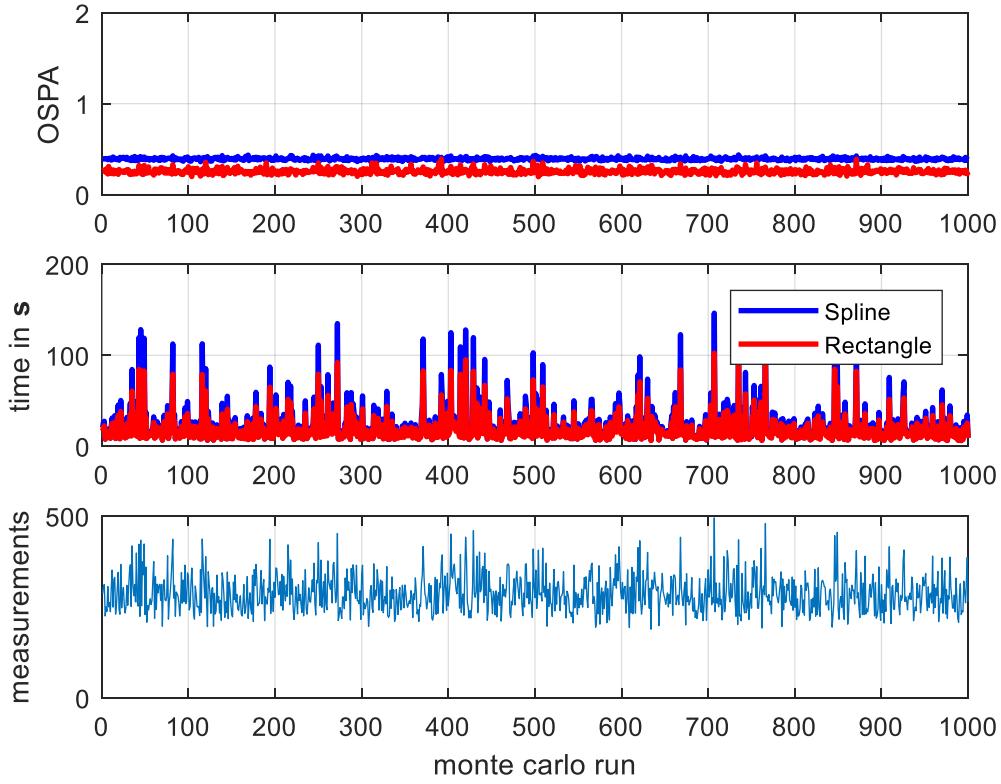

Figure 7.12: Second Monte Carlo simulation for the comparison

7.3 Evaluation of multi object trackers

When evaluating a multi object tracker, the distance of the estimation to the ground truth needs to be measured. In the case of a single object the Euclidean distance can be taken into account. When tracking multiple objects in a cluttered environment the mean distance of an estimation to the ground truth could be considered. However, a labeling of the estimations would have to be available for this evaluation, which is not the case for a general PHD filter. But more importantly, the number of estimations and ground truths is not always the same as misdetections, misestimating the cardinality and false associations can occur. The goal is a multi object metric that incorporates all these outcomes of the filter. One solution is provided by the OSPA metric proposed in [39]. Given a set of references $\boldsymbol{X}_k = \left\{\boldsymbol{x}_k^{(i)}\right\}_{i=1}^{m_k} \in \mathcal{F}(\boldsymbol{W})$ and a set of estimations $\widehat{\boldsymbol{X}}_k = \left\{\widehat{\boldsymbol{x}}_k^{(i)}\right\}_{i=1}^{n_k} \in \mathcal{F}(\boldsymbol{W})$ as subsets of the same underlying set $\boldsymbol{W}$, the OSPA metric is defined as

$$\bar{d}_p^{(c)}\left(\boldsymbol{X}_k, \widehat{\boldsymbol{X}}_k\right) := \left(\frac{1}{n_k}\left(\min_{\pi\in\Pi_{n_k}} \sum_{i=1}^{m_k} d^{(c)}\left(\boldsymbol{x}_k^{(i)}, \widehat{\boldsymbol{x}}_k^{\pi(i)}\right)^p + c^p \cdot (n_k - m_k)\right)\right)^{\frac{1}{p}} \tag{7.7}$$

if $m_k \leq n_k$. If $m_k > n_k$ the OSPA metric is simply defined as $\bar{d}_p^{(c)}\left(\boldsymbol{X}_k, \widehat{\boldsymbol{X}}_k\right) := \bar{d}_p^{(c)}\left(\widehat{\boldsymbol{X}}_k, \boldsymbol{X}_k\right)$. The parameter p defines the p-norm used to calculate the OSPA. The parameter c is given as the cut off distance if $n_k \neq m_k$ and an assignment of an estimation to a reference is not possible in every case. The minimal sum of distances is calculated by considering every permutation $\pi \in \Pi_{n_k}$ of the estimation set as assignment specification. The distance $d^{(c)}(\boldsymbol{x}, \boldsymbol{y}) := \min\left(c, d(\boldsymbol{x}, \boldsymbol{y})\right)$ denotes an arbitrary metric with values in $[0, c]$. The minimization problem can be solved using either the Kuhn-Munkres algorithm [41] or the auction algorithm [42]. To evaluate the performance of an extended multi object tracker, the distance measure of (7.1) is used. The adapted OSPA metric for extended objects is then calculated as

$$\hat{d}_p^{(c)}\left(\boldsymbol{X}_k, \widehat{\boldsymbol{X}}_k\right) := \left(\frac{1}{n_k}\left(\min_{\pi\in\Pi_{n_k}} \sum_{i=1}^{m_k} d_{OSPA,n}^{(c)}\left(p_{\boldsymbol{x}_k^{(i)}}, p_{\widehat{\boldsymbol{x}}_k^{\pi(i)}}\right)^p + c^p \cdot (n_k - m_k)\right)\right)^{\frac{1}{p}}. \tag{7.8}$$

Using the adapted OSPA measure for extended objects as distance for the OSPA metric, the position, orientation and extension errors for each estimation are calculated in one number and combined to a performance evaluation of the multi extended object tracker. Another evaluation tool for multi extended object trackers is the cardinality comparison. Thus, the true number of objects and the estimated number of objects are compared. The estimated number of objects can either be computed by taking the number of extracted objects or by the sum of updated, merged and pruned weights as

$$\text{cardinality} = \sum_{j=1}^{J_{k|k}} \omega_{k|k}^{(j)}. \tag{7.9}$$

The cardinality is computed as sum of weights in this thesis.

7.4 Performance of the spline PHD filter

This section is about to evaluate the performance of the spline PHD filter presented in Table 2. Equally to the spline EKF filter, the object state vector is used according to (6.1). As system state transition model also a coordinated turn model with polar velocity [8] is combined with an assumed constant extension according to (7.2). The system standard deviations are specified as $\sigma_v = 2m/s$, $\sigma_\omega = 3°/s$, $\sigma_{s_x} = 0.1m$ and $\sigma_{s_y} = 0.1m$ to form the system covariance matrix given in (7.3) and (7.4). A single measurement covariance matrix is computed using (7.5) with the measurement standard deviations of $\sigma_r = 0.1m$ and $\sigma_\vartheta = 0.5°$. For clustering the measurement set $\boldsymbol{Z}_k$ a DBSCAN algorithm [37] with a set of parameters given as $\boldsymbol{\varepsilon} = (1, 1.2, \ldots, 5)$, to create 21 different partitions in each time step, is used. The implementation of the DBSCAN algorithm is taken from [44]. The thresholds for merging and pruning are taken as $U = 4$ and $T = 10^{-5}$ while the maximum number of PHD components is specified as $J_{\max} = 100$. Two scenarios, introduced in the next subsection, are investigated using the spline PHD filter. The probability of survival $p_S = 0.99$ is equal in both scenarios. As the probability of detection differs in both scenarios it is given in the next subsection.

7.4.1 Simulation environment

The measurement creation is similar to the spline EKF simulation environment explained in subsection 7.2.1. Unlike the measurement creation for a single object, the occlusion of an object has to be considered for the measurement creation of several objects. Also, the fact if the measurements are inside of the surveillance area has to be checked. The first scenario investigated is illustrated in Figure 7.13.

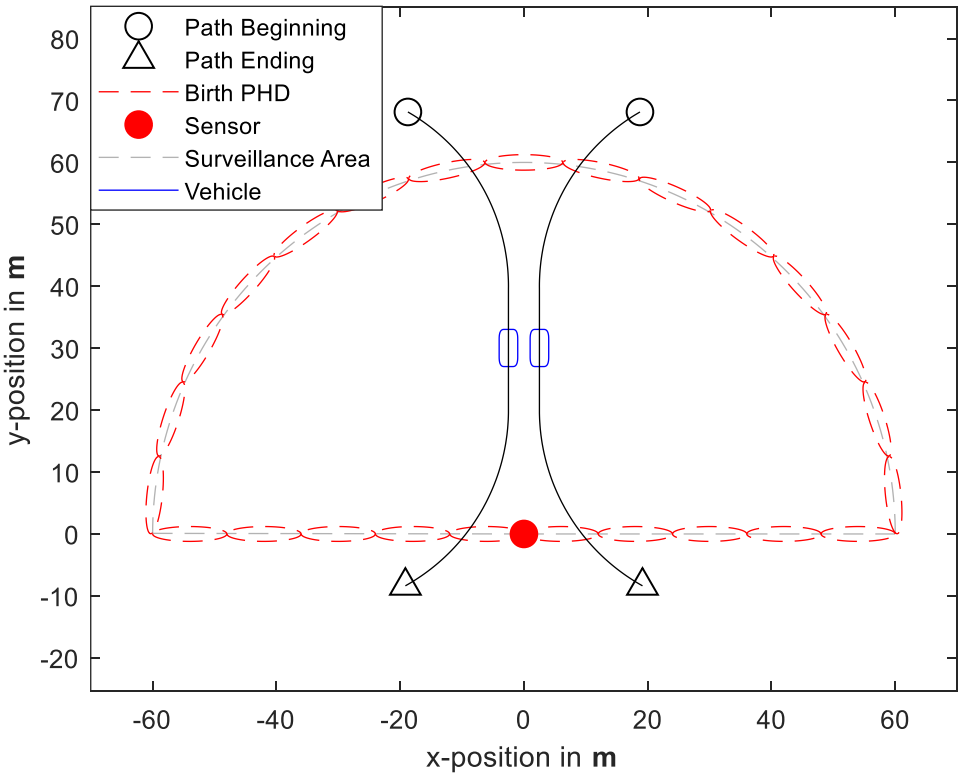

Figure 7.13: First investigated scenario with the PHD filter

In this scenario two objects enter the surveillance area with the edge of the contours moving up to two meters towards each other. Following, they separate again and exit the surveillance

area. By investigating this scenario, the birth process and tracking of closely spaced objects can be examined. Especially the performance of the DBSCAN algorithm needs to be appropriate for closely spaced objects. The probability of detection is taken as $p_D = 0.99$ as the objects are assumed to be detected in every time step. The second investigated scenario is illustrated in Figure 7.14.

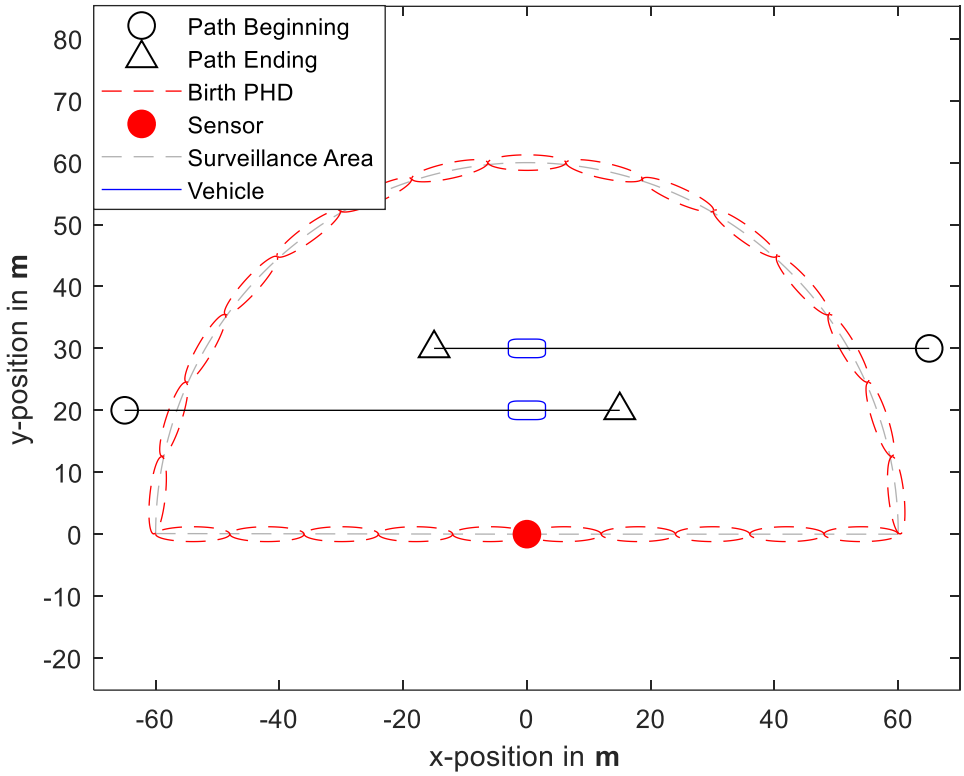

Figure 7.14: Second investigated scenario with the PHD filter

In this scenario two objects enter surveillance area and drive past each other. At the level of the sensor one object is occluded by the other and does not generate a single measurement for a few time steps. In order for the GM component representing the occluded object to survive, the probability of detection is taken as $p_D = 0.9$ in this scenario.

The number of birth PHDs positioned as illustrated in Figure 7.13 and Figure 7.14 is $J_b = 25$ using weights of $\omega_b^{(j)} = \frac{0.1}{J_b} = \frac{0.1}{25}$ with $j = 1, \dots, J_b$ in every time step. The positions of the birth PHDs are chosen like that because the spontaneous births are assumed to happen at the edge of the surveillance area, as an object enters it, not inside of the surveillance area. The opening angle of the sensor is $180°$ with a resolution of $1°$ and a range of $60m$ forming a semicircle. The objects entering the surveillance area are simulated as rectangles with a length of $6m$ and a width of $3m$. The spatial clutter distribution is modeled as uniform distribution over the surveillance area, while the number of clutter measurements is modeled as Poisson distributed random number with a rate of $\lambda = 10$ clutter measurements per time step.

7.4.2 Simulation results

In this section the simulation results of the spline PHD filter adapted to both scenarios are illustrated and discussed. The parameters and settings are taken as mentioned before in both scenarios. Following, the results of a single run and a Monte Carlo simulation with 1000 runs

for each scenario are given. The tracker is investigated by considering the cardinality as sum of weights and the OSPA distance as specified in section 7.3. For the OSPA metric the cut off distance $c = 30m$ is used. The adapted OSPA distance, to measure the distance of an estimated single extended object to its ground truth, is used with a number of $n = 50$ samples on the contour as distance measure in the OSPA distance. Both minimization problems, for the OSPA and adapted OSPA distance, are solved using the Kuhn-Munkres algorithm. The results of a single run on the first scenario are illustrated in Figure 7.15.

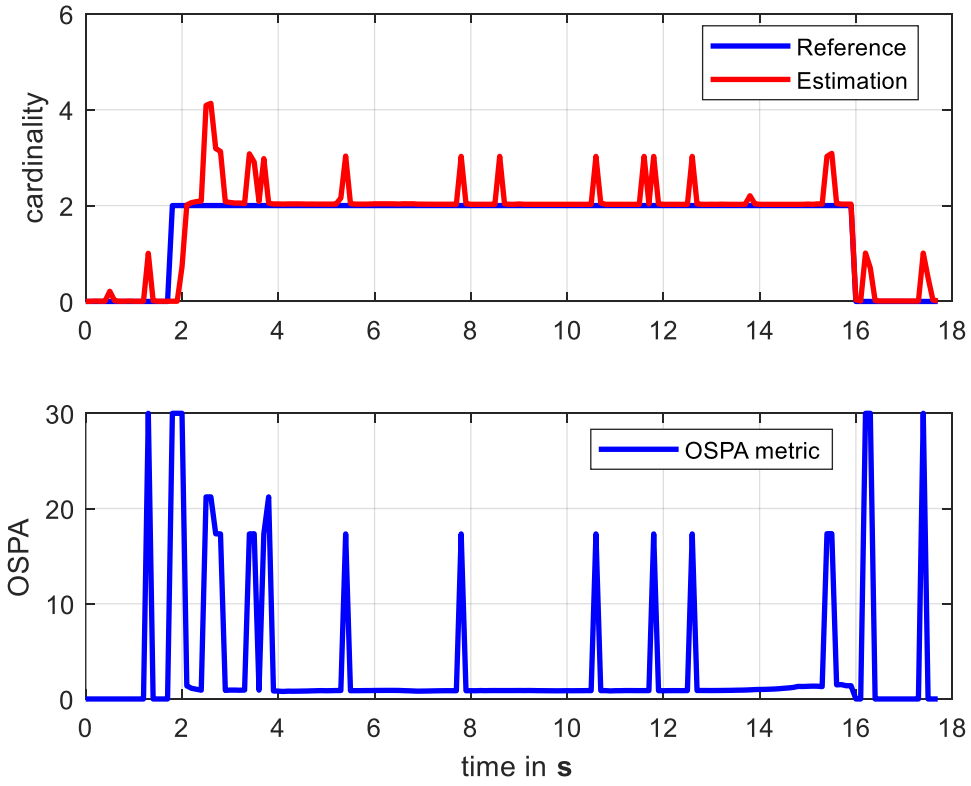

Figure 7.15: Single run result on first scenario

The cardinality plot shows a working birth process as the estimated cardinality rises when the objects enter the surveillance area. However, the birth process is an error-prone process. During the birth of the objects the OSPA distance is very high in most of the time steps, also the cardinality is at its level of four estimated objects. This issue first arises from the fact, that the objects entering the surveillance area is a process of a few time steps. The sensor does not detect the whole contour of the object from the first moment on. Thus, the correct estimation of the position, orientation and extension fails. Furthermore, the spline measurement model struggles with an imprecise state prediction. With the objects entering the surveillance area a few meters away from the mean of the birth PHD, the measurement prediction in the update step also fails. However, the birth process still works. Another abnormality in Figure 7.15 are the peaks in the cardinality and OSPA plot. The peaks in the cardinality occur if some clutter measurements are close to a birth PHD and assumed as new object. Subsequently the same peaks occur in the OSPA plot. If a peak in the cardinality plot does not engender a peak in the OSPA plot, still the true number of objects is extracted in the last step of the spline PHD filter. As the state extraction only considers GM components with weights higher than 0.5 as true objects, two or more GM components must have weights lower than this threshold in this case. For the calculation of the OSPA distance only the GM

components after the state extraction are considered. The DBSCAN clustering algorithm can separate the measurement sets occurring from different closely spaced objects as the estimated cardinality is always two or higher. The spline measurement model still performs pretty good within the PHD filter since the OSPA distance is very low in the time steps of a good performance. The results of a single run on the second scenario are illustrated in Figure 7.16.

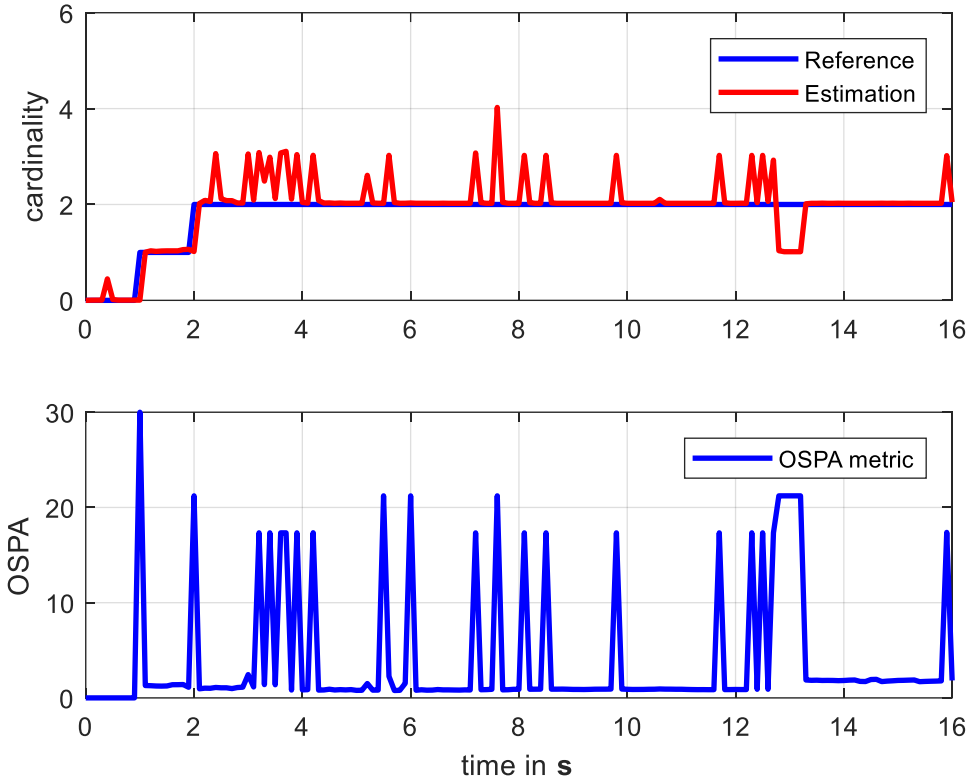

Figure 7.16: Single run result on first scenario

Those results also show a working birth process as the cardinality rises when the objects enter the surveillance area. The cardinality and OSPA plots show the same peaks like in Figure 7.15. An additional abnormality are the peaks in the OSPA distance without a peak in the cardinality plot at the same time. In those cases, a true object is not detected of the filter, while a wrong object containing only clutter measurements is taken as true object. Thus, the cardinality estimate is close to the reference, while the OSPA distance shows a peak. Another result of investigating the second scenario concerns the occlusion of one object. The occlusion takes place after about 13 seconds. For a few time steps the estimated cardinality shrinks to one but rises up to two again as the object is detected again. So the occluded object is tracked again as it is detected again. The results of the single runs on both scenarios already show a working multi extended object filter but still many problems that need to be solved. The birth process is not that accurate because of the initialization problem of the spline measurement model that needs to be improved. Also, an improvement of the state extraction could yield in better performance results by considering not only the actual state extraction but also the previous ones. As the results of a single run do not provide a long-term conclusion, the following illustrations show the results of a Monte Carlo simulation with 1000 runs on each scenario. The results of the Monte Carlo simulation on the first scenario are illustrated in Figure 7.17.

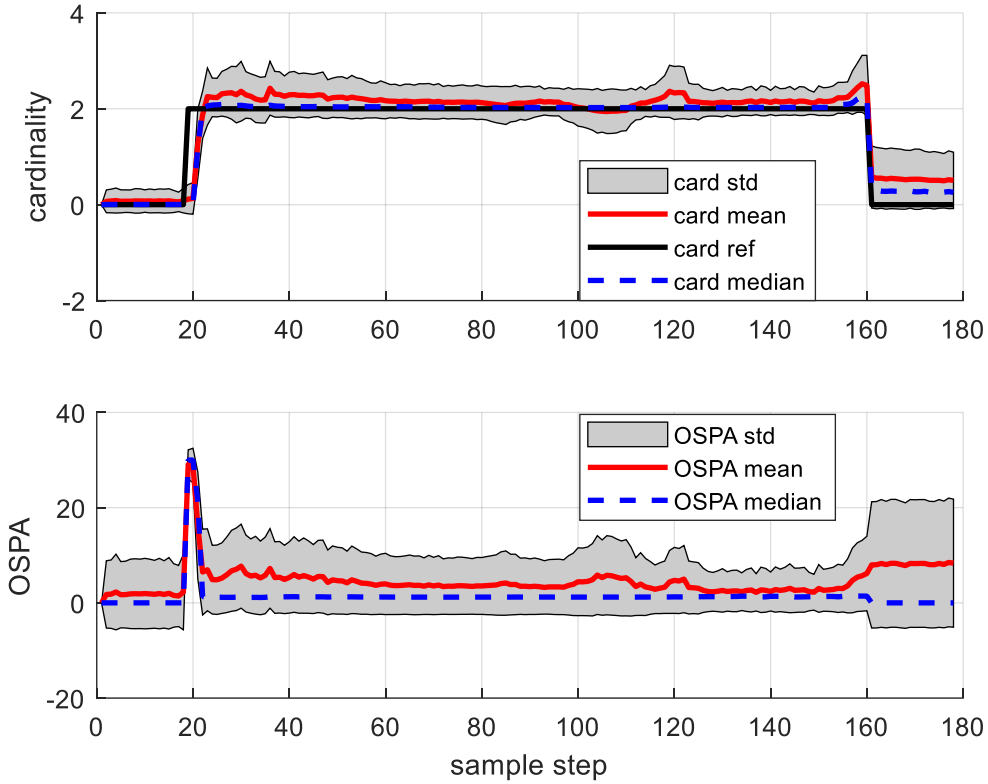

Figure 7.17: Results of Monte Carlo simulation on first scenario

The OSPA and cardinality plot show the means, medians and standard deviations considering the 1000 results in each sample step. The same trajectories are used in every Monte Carlo run, but the number and position of the clutter measurements, as well as the noise and the number of the object generated measurements differs in every run. The results of this long-term simulation clarify the assumptions made after the single run on the first scenario. The birth process still works after considering 1000 runs. However, the OSPA distance shows a peak at the moment of the objects entering the surveillance area, that clarifies the bad performance of the birth process. The median of both the estimated cardinality and the OSPA distance shows a nearly perfect performance of the filter in this scenario. However, the mean once more illustrates the aforementioned problems. The cardinality is overestimated in most of the time steps leading to a higher mean OSPA distance than the median. The state extraction as well as the birth process need to be improved for a better performance. In summary, the functionality of the filter is proven considering the results of the first Monte Carlo simulation. The results of the Monte Carlo simulation on the second scenario are illustrated in Figure 7.18. The cardinality as well as the OSPA plot show a worse performance of the spline PHD filter on the second scenario compared to the first scenario. The overestimation of the cardinality is higher in both the mean and the median leading to a larger OSPA distance compared to the first scenario. The functionality of the filter however is still proven considering the results of Figure 7.18. The occlusion process taking place after about 13 seconds also works in a long-term simulation. The occluded object is tracked again as it is detected again. This can be seen as the estimated cardinality rises and the OSPA distance shrinks as the object is detected again.

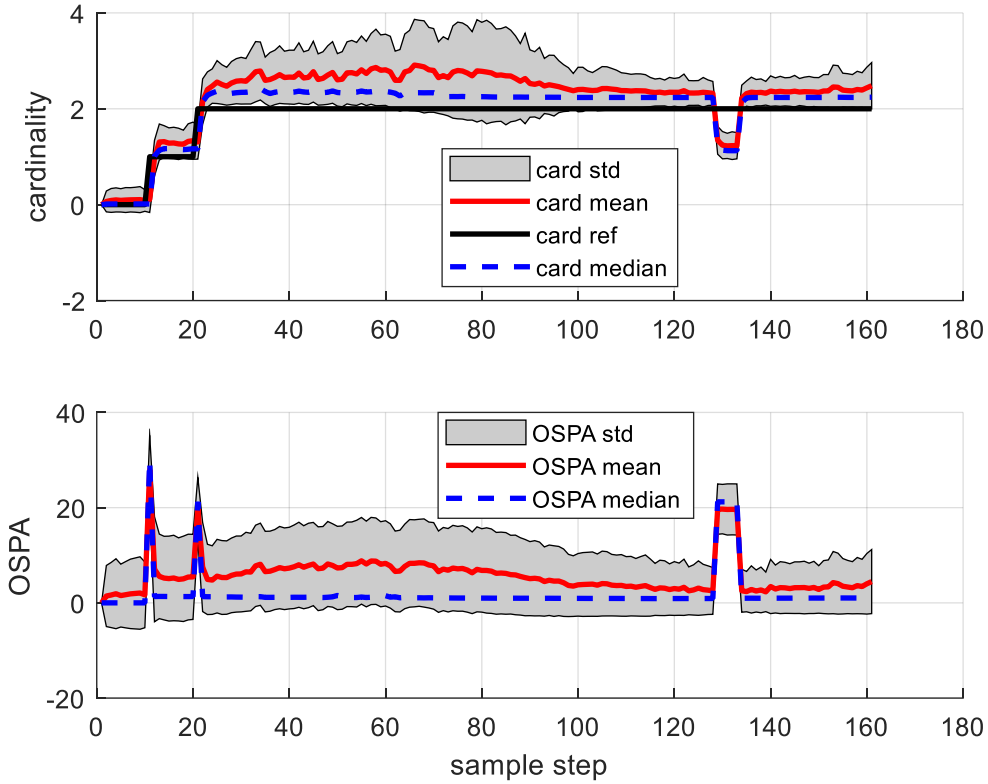

Figure 7.18: Results of Monte Carlo simulation on second scenario

All the investigations addressed in this chapter show a good performance of the spline measurement model although some problems still remain. In the next chapter the conclusions summarizing the deliberations and results of this thesis and possible ideas for future work are addressed.

8 Conclusions

In this thesis the spline measurement model [1] for tracking extended objects with LIDAR measurements was investigated. The shape of the object is modeled using quadratic periodic uniform B-spline functions. To establish the basis of the resulting filter algorithms, the theories of single object tracking, extended object tracking and multiple object tracking were briefly addressed. Also, the theory of B-spline functions was recapped. To use a B-spline represented object contour in a tracking algorithm a measurement prediction represented by a measurement source on the contour is needed. Additionally, those measurement sources need to be derived with respect to the object state. The derivation of those equations was presented in detail. Also, the use of a spline contour for other objects than vehicles was briefly addressed. To use the spline measurement model in a tracking scenario it was first integrated in an EKF framework. The full algorithm is given in pseudocode. The implementation was investigated using Monte Carlo simulations. Furthermore, the spline measurement model was compared to the rectangular shape estimator [22] also using Monte Carlo simulations. In order to track several extended objects in a cluttered environment, the spline measurement model was integrated in the GM-PHD filter for extended objects [35]. The equations, as well as the full algorithm in pseudocode is given. The investigation of the spline PHD filter using Monte Carlo simulations shows the successful integration of the spline measurement model in the GM PHD filter for extended objects, as well as the problems of this filter. Further investigations

could deal with the initialization process of the spline measurement model, as well as with improving the birth process of the PHD filter for extended objects as explained in section 7.4. The improvement of those processes could lead to a better performance of the spline PHD filter considering the overestimation of the cardinality and the poor birth process.

9 References

[1] H. Kaulbersch, J. Honer and M. Baum, "A Cartesian B-Spline Vehicle Model for Extended Object Tracking," in *2018 21st International Conference on Information Fusion (FUSION)*, 2018.

[2] S. Challa, M. R. Morelande, D. Mušicki and R. J. Evans, Fundamentals of Object Tracking, Cambridge University Press, 2011.

[3] K. Granström, "Extended target tracking using PHD filters," 2012.

[4] B. T. Vo, "Random Finite Sets in Multi-Object Filtering," 2008.

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

[6] Y. Bar-Shalom, T. Kirubarajan and X.-R. Li, Estimation with Applications to Tracking and Navigation, New York, NY, USA: John Wiley & Sons, Inc., 2002.

[7] S. S. Blackman and R. Popoli, Design and Analysis of Modern Tracking Systems, Artech House, 1999.

[8] M. Roth, G. Hendeby and F. Gustafsson, "EKF/UKF maneuvering target tracking using coordinated turn models with polar/Cartesian velocity," in *17th International Conference on Information Fusion (FUSION)*, 2014.

[9] P. A Bromiley, "Products and Convolutions of Gaussian Distributions," 1 2003.

[10] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering,* vol. 82, pp. 35-45, 1960.

[11] Y. Bar-Shalom, Tracking and Data Association, San Diego, CA, USA: Academic Press Professional, Inc., 1987.

[12] S. J. Julier and J. K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *Proc. of AeroSense: The 11th Int. Symp. on Aerospace/Defense Sensing, Simulations and Controls*, 1997.

[13] O. Cappe, S. J. Godsill and E. Moulines, "An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo," *Proceedings of the IEEE,* vol. 95, pp. 899-924, 5 2007.

[14] A. Doucet, N. Freitas and N. Gordon, "An Introduction to Sequential Monte Carlo Methods," in *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. Freitas and N. Gordon, Eds., New, York: Springer New York, 2001, pp. 3-14.

[15] K. Granström, M. Baum and S. Reuter, "Extended Object Tracking: Introduction, Overview, and Applications," *Journal of Advances in Information Fusion,* vol. 12, 12 2017.

[16] K. Gilholm, S. Godsill, S. Maskell and D. Salmond, "Poisson models for extended target and group tracking," in *Optics & Photonics 2005*, 2005.

[17] W. Koch, "Bayesian Approach to Extended Object and Cluster Tracking using Random Matrices," *IEEE Transactions on Aerospace and Electronic Ssystems,* vol. 44, pp. 1042-1059, 7 2008.

[18] M. Baum and U. D. Hanebeck, "Random Hypersurface Models for extended object tracking," in *2009 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 2009.

[19] K. Granstrom and U. Orguner, "A phd Filter for Tracking Multiple Extended Targets Using Random Matrices," *IEEE Transactions on Signal Processing,* vol. 60, pp. 5657-5671, 11 2012.

[20] M. Baum, M. Feldmann, D. Fränken, U. D. Hanebeck and W. Koch, "Extended Object and Group Tracking: A Comparison of Random Matrices and Random Hypersurface Models," in *Proceedings of the IEEE ISIF Workshop on Sensor Data Fusion: Trends, Solutions, Applications (SDF 2010), October 2010, Leipzig*, 2010.

[21] M. Baum and U. D. Hanebeck, "Shape tracking of extended objects and group targets with star-convex RHMs," in *14th International Conference on Information Fusion*, 2011.

[22] P. Broßeit, M. Rapp, N. Appenrodt and J. Dickmann, "Probabilistic rectangular-shape estimation for extended object tracking," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, 2016.

[23] B.-N. Vo, M. Mallick, Y. Bar-shalom, S. Coraluppi, R. Osborne III, R. Mahler and B.-T. Vo, "Multitarget Tracking," in *Wiley Encyclopedia of Electrical and Electronics Engineering*, American Cancer Society, 2015, pp. 1-15.

[24] D. Musicki and R. Evans, "Joint Integrated Probabilistic Data Association: JIPDA," *Aerospace and Electronic Systems, IEEE Transactions on,* vol. 40, pp. 1093-1099, 8 2004.

[25] R. P. S. Mahler, ""Statistics 101" for multisensor, multitarget data fusion," *IEEE Aerospace and Electronic Systems Magazine,* vol. 19, pp. 53-64, 1 2004.

[26] R. P. S. Mahler, Statistical Multisource-Multitarget Information Fusion, Norwood, MA: Artech House, Inc., 2007.

[27] B. Ristic, B. Vo, B. Vo and A. Farina, "A Tutorial on Bernoulli Filters: Theory, Implementation and Applications," *IEEE Transactions on Signal Processing,* vol. 61, pp. 3406-3430, 7 2013.

[28] R. P. S. Mahler, "Multitarget Bayes filtering via first-order multitarget moments," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 39, pp. 1152-1178, 10 2003.

[29] B. N. Vo and W. K. Ma, "The Gaussian Mixture Probability Hypothesis Density Filter," *IEEE Transactions on Signal Processing,* vol. 54, pp. 4091-4104, 11 2006.

[30] B. N. Vo, S. Singh and A. Doucet, "Sequential Monte Carlo methods for multitarget filtering with random finite sets," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 41, pp. 1224-1245, 10 2005.

[31] R. Mahler, "PHD filters of higher order in target number," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 43, pp. 1523-1543, 10 2007.

[32] B. Vo, B. Vo and A. Cantoni, "The Cardinalized Probability Hypothesis Density Filter for Linear Gaussian Multi-Target Models," in *2006 40th Annual Conference on Information Sciences and Systems*, 2006.

[33] R. Mahler, "PHD filters for nonstandard targets, I: Extended targets," in *2009 12th International Conference on Information Fusion*, 2009.

[34] G.-C. Rota, "The Number of Partitions of a Set," *The American Mathematical Monthly,* vol. 71, pp. 498-504, 1964.

[35] K. Granström, C. Lundquist and O. Orguner, "Extended Target Tracking using a Gaussian-Mixture PHD Filter," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 48, pp. 3268-3286, 10 2012.

[36] A. Blake and M. Isard, Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion, 1st ed., Springer Publishing Company, Incorporated, 2012, pp. 41-63.

[37] M. Ester, H.-P. Kriegel, J. Sander and X. Xu, "A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, 1996.

[38] K. Granström and U. Orguner, Implementation of the GIW-PHD filter, Linköping University Electronic Press, 2012.

[39] D. Schuhmacher, B. Vo and B. Vo, "A Consistent Metric for Performance Evaluation of Multi-Object Filters," *IEEE Transactions on Signal Processing,* vol. 56, pp. 3447-3457, 8 2008.

[40] S. Yang, M. Baum and K. Granström, "Metrics for performance evaluation of elliptic extended object tracking methods," in *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2016.

[41] H. W. Kuhn and B. Yaw, "The Hungarian method for the assignment problem," *Naval Res. Logist. Quart,* pp. 83-97, 1955.

[42] D. P. Bertsekas, "Auction Algorithms," in *Encyclopedia of Optimization, Second Edition*, 2009, pp. 128-132.

[43] Y. Cao, "Hungarian Algorithm for Linear Assignment Problems (V2.3)," 2011. [Online]. Available: https://de.mathworks.com/matlabcentral/fileexchange/20652-hungarian-algorithm-for-linear-assignment-problems-v2-3.

[44] P. Kovesi, "Basic implementation of DBSCAN clustering," 2013. [Online]. Available: https://www.peterkovesi.com/matlabfns/.