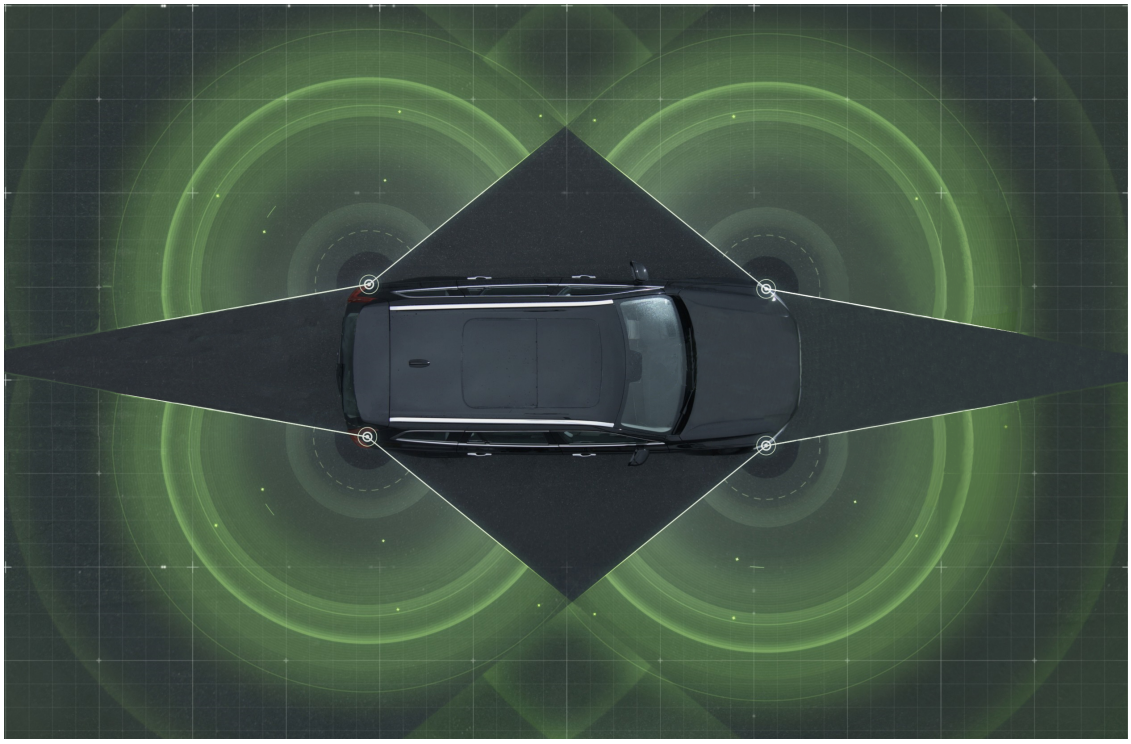


Radar-based target tracking for 360-degree environmental perception

Master's thesis in Systems, Control and Mechatronics

Erik Henriksson
Viktor Kardell

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Master's thesis EX025/2016

Radar-based target tracking for 360-degree environmental perception

Erik Henriksson
Viktor Kardell

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals and System
Chalmers University of Technology
Gothenburg, Sweden 2016

Radar-based target tracking for 360-degree environmental perception
Erik Henriksson
Viktor Kardell

Supervisors: Malin Lundgren & Daniel Svensson, Volvo Car Group
Examiner: Lennart Svensson, Signals and Systems, Chalmers University of Technology

Master's Thesis EX025/2016
Department of Signals and System
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The sensor placement and fields-of-view of the four sensors used in this thesis.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2016

Radar-based target tracking for 360-degree environmental perception
Erik Henriksson, Viktor Kardell
Department of Signals and Systems
Chalmers University of Technology

Abstract

This thesis focuses on extended target tracking of dynamic objects using automotive radar sensors. The tracking is based on data from a 360-degree environmental perception system comprising four radar sensors with overlapping fields-of-view. Two solutions are proposed to track the states of the objects, which include position, velocity, heading and size. The first algorithm forms clusters based on detections and creates rectangles that are used in the update step of an extended target tracker. The second algorithm uses a Gaussian Mixture Probability Hypothesis Density (GM-PHD) filter, clusters components of that filter and creates a rectangle around the filtered components. Evaluation on logged data shows good results for both solutions in terms of position and velocity accuracy. However, the detection-based tracking solution shows a slightly more stable result than the PHD-based solution. When it comes to estimation of the heading and the physical extension of objects, the proposed solutions differ somewhat, but both produce rather poor estimates. Especially at long ranges, the heading and size estimates are unstable.

Keywords: Target tracking, Extended targets, Radar, Kalman filter, Clustering methods, Automotive applications.

Acknowledgements

First, we want to thank Volvo Cars for giving us the opportunity to do our thesis at a great company. Also, we want to thank the Sensor fusion department for letting us be a part of the group. A special thanks to our highly dedicated and helpful supervisors Malin Lundgren and Daniel Svensson for their support throughout the entire thesis. This thesis wouldn't have been carried out the way that it did without them. We would also like to thank our examiner Lennart Svensson for his appreciated support. Together with these people we also would like to take the opportunity to thank all other persons that have helped us both during this spring, but also throughout our university studies.

At last, we would like to thank our respective families for their support and encouragement throughout the entire journey.

Erik Henriksson and Viktor Kardell, Gothenburg, June 2016

Contents

List of Figures

List of Tables

1
Introduction

When you buy a new car today it comes with many safety features designed to assist the driver. These systems, jointly called active safety systems, are intended to help the driver realising potential risks in the traffic and even avoid them. There are four levels of active safety functions. The first level is to warn the driver of potential hazards, and includes systems for forward collision warning and blind spot detection. The second level involves active intervention of the vehicle in specific safety-critical situations. An example of such a system is collision mitigation by braking. The third level is semi-autonomous driving, where the vehicle drives itself in limited environments, but where the driver is responsible at all times. The final and highest level is autonomous driving, where the vehicle drives without a driver in control [1].

Today, no fully autonomous cars are on the roads, but several car manufacturers are aiming at delivering autonomous cars in a near future. By achieving this, the amount of inattentive drivers can be decreased, and thus the number of traffic incidents caused by a driver can potentially be reduced. Another benefit, mainly for the driver, is that while the car is responsible to drive in a safe way, the driver can focus on tasks he or she find more important. Additionally, the fuel consumption could potentially be decreased with cars driving in a fuel conservative way, which is a step towards a sustainable future [2].

Both autonomous vehicles and active safety systems require a description of the surrounding environment in order to make accurate decisions. It is important to be able to detect all kinds of dangers that potentially can cause an accident. It can be to detect a pot hole in the road ahead and calculate a route to avoid the danger, but it can also be to detect other road users that have a predicted trajectory that collides with the trajectory of the ego vehicle. Hence, the first step in having autonomous cars is to sense and describe the surroundings including both static and dynamic objects, and make this information accessible for the car.

In order to create a description of the surrounding environment, vehicles must be equipped with a combination of different sensors. The type of sensors are determined by the application that they are used for. Common sensor types for vehicles are camera, laser and radar. For active safety systems, it is beneficial to use several sensors because of their different strengths. Typically, cameras are good for classifying objects but lacks the possibility of accurately estimating speed of objects. This is a strength of the laser and is therefore used just for this reason. The drawback of the laser is that it is very sensitive to weather such as rain or snow and mud [3]. The radar is, in difference from the camera and laser, robust to weather and is called a all-weather sensor [4] but does not give as accurate angular resolution as the

laser or as good object classification as the camera. This is why the combination of sensors is necessary in an automotive application. In this thesis project, the focus is on using radars solely.

For any type of surveillance system where dynamic targets are involved, target tracking is an important component. The target tracking objective is to collect data from sensors and filter data from the same object over time into a so-called track. From these tracks, estimated characteristics such as velocities, accelerations and future positions of tracked objects can be obtained. One important key of tracking multiple targets is to associate data to the right tracks. This is referred to as data association and is a necessary part of target tracking [4].

A complete 360-degree environmental perception requires information from multiple radars which in turn requires fusion in some way. There are at least two alternatives of fusing the information; tracking in each sensor and combining the information or tracking using all detections in central tracking. The last approach is used in this thesis project.

Using radars in an automotive setting may give rise to more than one detection per object and scan. The majority of the target tracking literature focuses on point targets, i.e., targets that give rise to at most one detection. An object that may give rise to several detections is referred to as an extended object. Hence, the area of tracking such objects is denoted extended target tracking.

1.1 Purpose

This thesis will evaluate the the capabilities of radar-based target tracking and investigate how to handle extended targets in an automotive setting. By developing a target tracker based on radar measurements, the information provided may then be fused with the other sensors. This is to ultimately achieve a system that can describe the environment in a robust and accurate way.

1.2 Objective

The objective of the thesis project is to detect and track objects in the surroundings of the vehicle using radar sensors. The tracked objects are extended, and shall be represented by the following information: position, velocity, heading, length and width. The aim is to have a robust solution, which can be tuned to the requirements.

The focus of the work is to have a tracking algorithm for logged data in an offline environment. Although the tracking algorithm will be built for offline use, a long term goal is to implement the algorithm in a real-time system. Therefore ease for such an implementation shall be considered during the development phase.

1.3 Thesis Outline

The thesis gives in Chapter 2 a description of the system used for developing the tracking algorithms, such as vehicle, sensors and logged data together with the

problem formulation. Chapter 3 covers the underlying theory to the methods used in this report. Chapter 4 provides discussion of existing methods and two proposals for solving the problem. Chapter 5 explains the methods and implementations that is common for both proposals. Chapter 6 and 7 present the two implemented solutions respectively and Chapter 8 displays the results. Lastly, Chapter 9 presents discussion and a conclusion of the thesis project.

2
Problem description

This chapter provides a description of the studied problem and of the physical system. First, a system overview is presented, containing technical specifications of the sensors and their mounting location on the vehicle, followed by a description of the data acquired from the sensors. Finally, the problem is formulated and limitations to the problem is formed.

2.1 System overview

The system used in this thesis is a vehicle mounted with a set of sensors. We do not have access to the vehicle, but are granted with logged data from test runs. The vehicle is a Volvo XC90 equipped with four radar sensors and one internal measurement unit (IMU) presented in this section, and these are the sensors of interest in this thesis.

Radar has an all-weather capability, and is accurate in measuring the range to and range-rate of an object [4]. These characteristics make it suitable for the object-tracking task in a traffic environment. A radar transmits a pulse of energy, and scans the response to check whether the pulse have reflected against a surface. The range to a reflective surface is related to the calculated round-trip time of the pulse. The radars also have the capability to measure the range rate of the reflected pulse. If the reflected surface moves along the vector of the transmitted pulse, the radar will be able to measure the velocity along this vector using the Doppler effect. By analyzing the phase difference of a returning wavefront at the different elements of an antenna, the radar is also capable of measuring the angle of arrival, i.e., the bearing to an object. The bearing measurements are typically less accurate than the range and range-range measurements.

The radars used in this thesis project is called side object detection sensors (SODs) and are mounted at the corners of the vehicle, as shown in Figure 2.1. The sensor placement, relative the ego vehicle coordinate system is given in Table 2.1 and the sensor specifications are given in Table 2.2. The SOD radars only measure the surroundings in a plane around the vehicle i.e. they only give information in two spatial dimensions. At each time k, we receive a collection of m measurements, $\mathbf{Z}_k^n = \{\mathbf{z}_k^{1,n}, \mathbf{z}_k^{2,n}, ..., \mathbf{z}_k^{m,n}\}$, from each radar sensor n. The measurements contain object-generated measurements, false alarms and clutter. Each measurement can be described by

$$\mathbf{z}_k^{i,n} = \begin{bmatrix} r_k^i, & \alpha_k^i, & \dot{r}_k^i \end{bmatrix}^T \tag{2.1}$$

where r_k^i is the range of the detection, α_k^i is the detection angle and $\dot{r}_k^i$ is the range-rate. Each SOD sensor provides 64 detections per scan, at a frequency of 20 Hz.

Table 2.1: Sensor placement.

Sensor (n)	Placement (x_s^n,y_s^n)	Angle (γ_s^n)
1	front left	58°
2	rear left	120°
3	rear right	−120°
4	front right	−58°

Table 2.2: Sensor specification.

Measurements	Coverage	Accuracy
Range [m]	0.3 to 85	$\pm 0.25\% + 0.1$
Angle [°]	−75 to 75	±1
Range rate [m/s]	−55 to 15	±0.07

The sensor placement is described by the offset relative to the ego vehicle reference frame and is given by

$$S^n = \{x_s^n, y_s^n, \gamma_s^n\} \tag{2.2}$$

where S^n is the specification for sensor n.

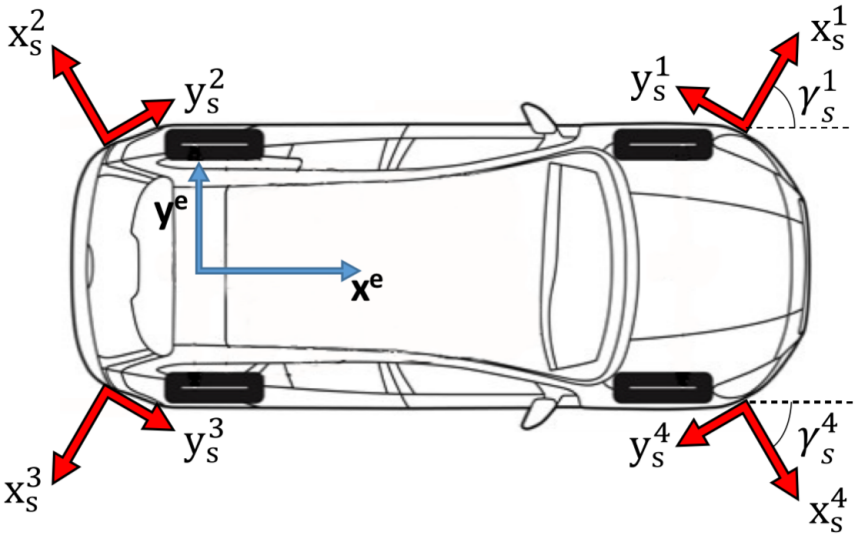


Figure 2.1: The coordinate systems of the vehicle. (x^e, y^e) represents the ego vehicle coordinate system. The coordinate systems at the corners of the ego vehicle represent the individual coordinate system of each sensor. The angular offset relative to the ego vehicle reference frame is described by γ_s^n

Since the ego vehicle is moving, the coordinate frame in Figure 2.1 is moving. To measure the translation and rotation, we use an internal measurement unit

(IMU). An IMU is an electronic device that measures the angular rate and the body specific force using a number of accelerometers and gyroscopes. In a vehicle, it is used to calculate the velocity, acceleration and yaw. From the IMU in the ego vehicle, the states describing the ego vehicle motion is given by

$$\mathbf{x}_k^e = \begin{bmatrix} \Delta x_k^e, & \Delta y_k^e, & \dot{x}_k^e, & \dot{y}_k^e, & \Delta\theta_k^e \end{bmatrix}^T \tag{2.3}$$

where Δx_k^e and Δy_k^e are changes in position from last time step, estimated from longitudinal velocity $\dot{x}_k^e$ and lateral velocity $\dot{y}_k^e$ respectively. $\Delta\theta_k^e$ is given by the change in heading from last time step. The IMU is providing measurements at 20 Hz. The specifications of the IMU is unknown but the uncertainties is assumed to be much lower than the uncertainties of the radars.

2.2 Radar data

The available radar data is logged at a test track and on public roads. The data is stored as MATLAB formatted data and built in structure arrays. From the radar, detections are output. Each detection includes information of range, range rate and detection angle to the reflecting point. Next, we describe the data in more detail. For confidentiality reasons, we are only allowed to show simulated data to illustrate the radar detections.

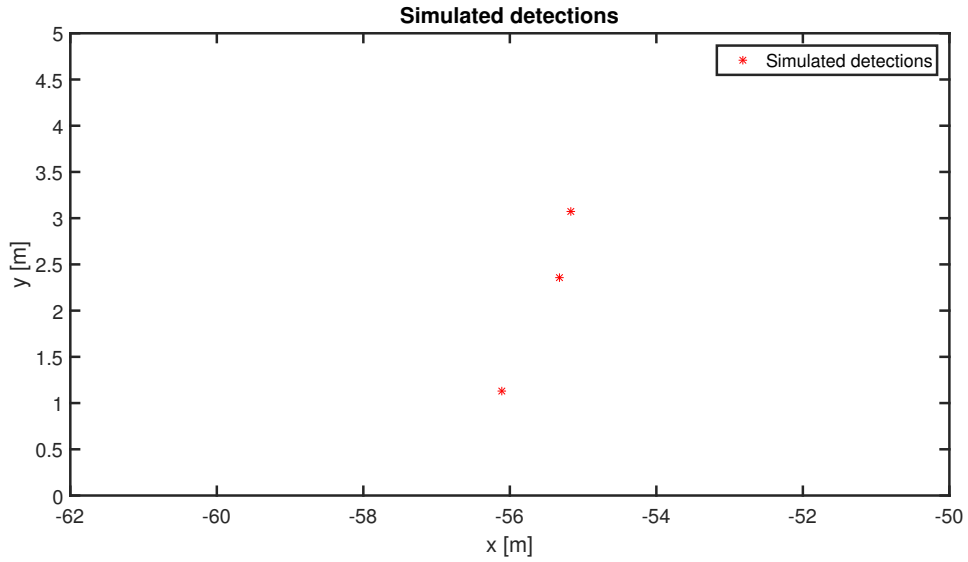


Figure 2.2: Samples of how the detections from another vehicle might appear at far distances. The expected number of detections at this distance is between one to three. Note that the ego vehicle is at the origin, i.e., at the far right.

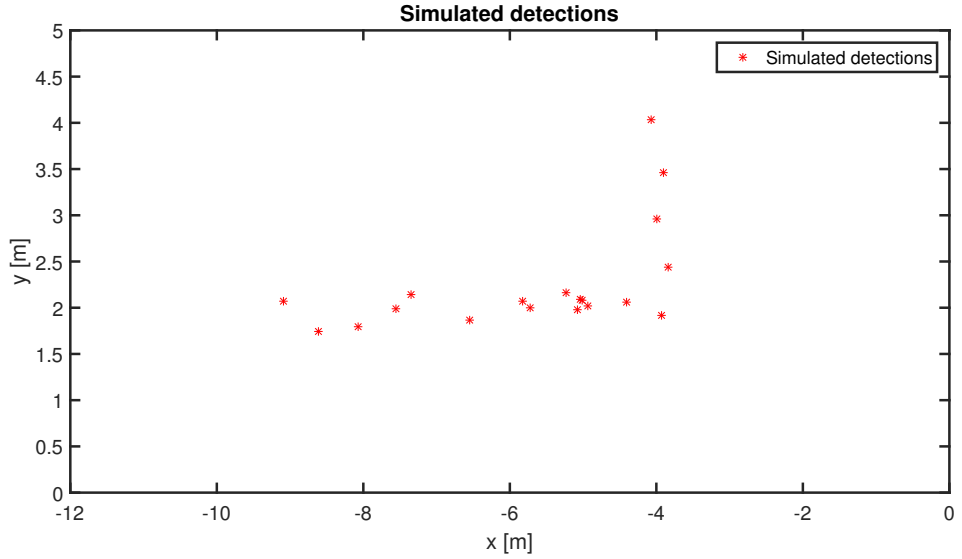


Figure 2.3: Samples of how the detections from another vehicle might appear at close distances. Here is a example when detections give a distinct description of the vehicle.

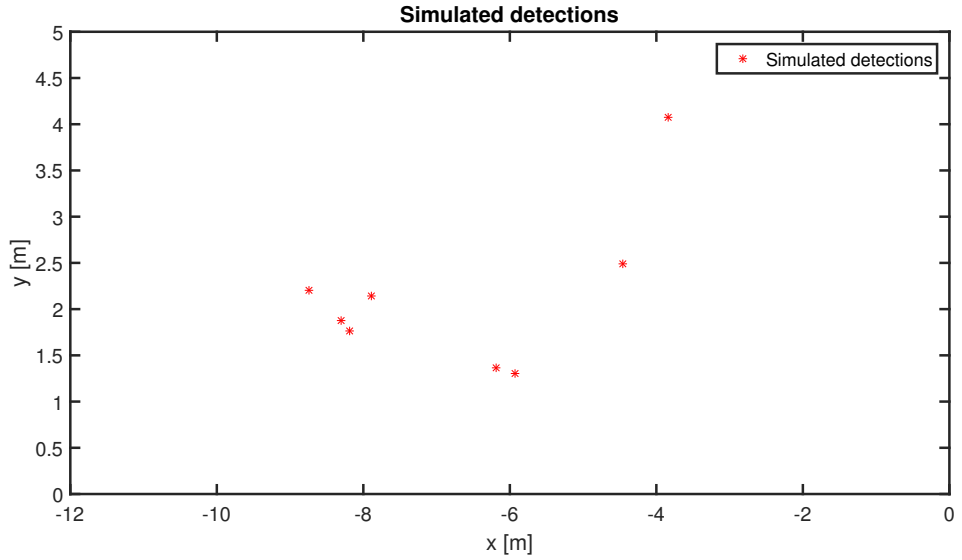


Figure 2.4: Samples of how the detections from another vehicle might appear at close distances. This figure shows an example of how several detections give a vague description of the vehicle.

Figure 2.2, 2.3 and 2.4 show three examples of how an object might appear in the view of the SODs. On a far distance, one to three measurements appear on the same object whereas if the object is about 5 meters from the ego vehicle, it generates a larger number of detections. In addition to detections from moving objects, the radar detects stationary structures such as guardrails and traffic signs.

2.3 Problem formulation

In order to have an autonomous car, the environment around the car needs to be surveyed and provide information to the control system. In this thesis, the studied problem is to detect and track dynamic objects in the surrounding environment. At each time k, the collection of measurements, $\mathbf{Z}_k^n = \{\mathbf{z}_k^{1,n}, \mathbf{z}_k^{2,n}, ..., \mathbf{z}_k^{m,n}\}$, from the four radar sensors contains detections of range, angle and range rate given by

$$\mathbf{z}_k^{i,n} = \begin{bmatrix} r_k^i, & \alpha_k^i, & \dot{r}_k^i \end{bmatrix}^T. \tag{2.4}$$

By using information from the radar sensors and information of the ego state,

$$\mathbf{x}_k^e = \begin{bmatrix} \Delta x_k^e, & \Delta y_k^e, & \dot{x}_k^e, & \dot{y}_k^e, & \Delta\theta_k^e \end{bmatrix}^T, \tag{2.5}$$

the aim is to estimate the object state, which include position, velocity, size and heading at the discrete time instants $k = 1, ..., K$. The state of an object is given by

$$\mathbf{x}_k = \begin{bmatrix} x_k, & y_k, & \dot{x}_k, & \dot{y}_k, & L_k, & W_k, & \theta_k \end{bmatrix}^T, \tag{2.6}$$

where (x_k, y_k) is the position relative to the ego vehicles coordinate system, $(\dot{x}_k, \dot{y}_k)$ is the velocity relative the ground, L_k and W_k is the length and width respectively and θ_k is the heading relative to the ego vehicles coordinate system according to Figure 2.1.

Determining the position of an object is not such a trivial task as it first might seem, first a valid representation of the position is necessary. Since handling extended targets, the detections will originate from different locations on the target and thus, a way to represent the position from the detections is needed.

As seen in Figure 2.2-2.4, the measurements appear very differently in different scenarios and time steps. All of these different cases must be handled in order to solve the problem. That includes combining all information gathered by the four sensors and create a 360-view of objects in the surrounding. Figure 2.5 illustrates that all four sensors will be used in a target tracker which tracks the objects.

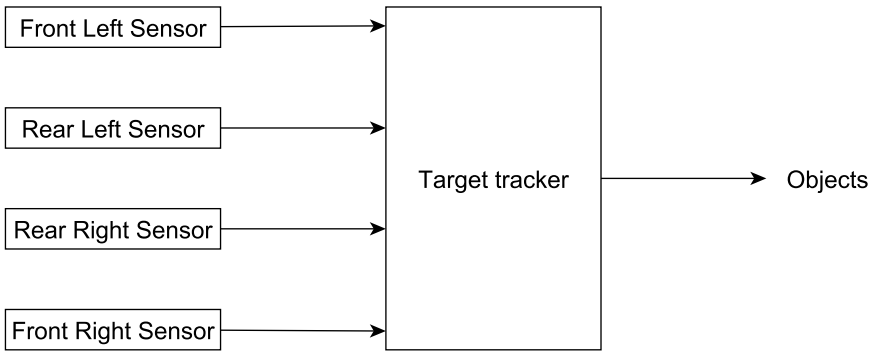


Figure 2.5: Description of the tracking system.

2.3.1 Limitations

To narrow the scope of the thesis, we have decided on a set of limitations to the tracking problem. Since the SOD radars only measure the surroundings in a plane, the tracking system is limited to operate in a 2D Cartesian coordinate system. The tracking system will use logged data, hence implementation in a real vehicle is not within the scope of this thesis. The main focus is on tracking moving passenger cars. Therefore, this thesis will focus on tracking objects of the certain size, shape and movements that are typical for a passenger car. At last, the scenarios considered is when all vehicles travels at the same road as the ego vehicle, i.e. no crossing traffic.

The thesis focuses on extended target tracking and how to approach the extended target tracking problem. Attributes such as tracking multiple targets will not be considered, and the solutions proposed in this thesis will be developed for a scenario with one target vehicle.

3
Theory

This section presents the underlying theory used in this report. It first covers the tracking theory in general and the background of Bayesian filtering used in tracking. Secondly, the approach of tracking using random finite sets is presented together with related work of tracking extended targets. To find detections that belong to the same physical object or to apply the point target assumption on extended targets, clustering is needed and it is covered in this section as well.

3.1 Tracking theory overview

The main objective of target tracking is to recursively estimate the states of the objects detected by one or several sensors. As illustrated in Figure 3.1, a tracking algorithm consists of four main parts, namely prediction, data association, measurement update and track handling. The prediction and measurement update is in this thesis referred as filtering.

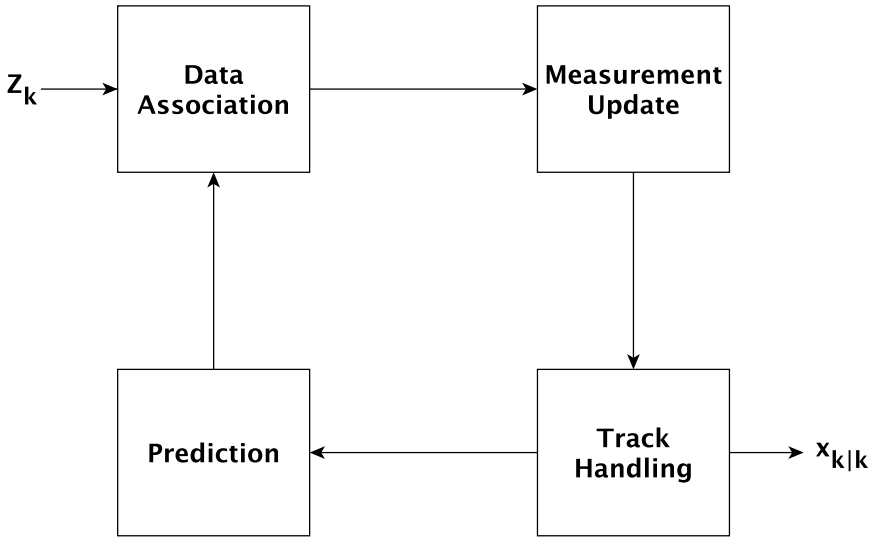


Figure 3.1: Target tracking in general where $\mathbf{Z}_k = \{\mathbf{z}_k^1, \mathbf{z}_k^2, ..., \mathbf{z}_k^m\}$ is measurements and $\mathbf{x}_{k|k}$ is the estimated state.

Given the posterior state estimate of objects, $\mathbf{x}_{k-1|k-1}$, and their corresponding covariances, $\mathbf{P}_{k-1|k-1}$, the state estimates are predicted using the motion model that describes the dynamics of the objects. The measurements, $\mathbf{Z}_k$, are associated to

tracks to determine which measurements that stem from which objects represented by tracks. Using the associated measurements, each is updated with new information. The track handling module initializes new tracks from the non-associated measurements and deletes tracks that have not been updated for a number of time steps. After the track handling, the target tracker outputs the current state estimates $\mathbf{x}_{k|k}$, and the covariances $\mathbf{P}_{k|k}$. This process is repeated at each time step in order to detect and track objects in the observed scene. In the next sections we present some background on filtering, data association and track handling.

3.2 Filtering

Filtering is about recursively estimating parameters of interest using data from a sensor. In the target tracking framework, filtering is about describing the sequence of states of an object, given sensor data about that object.

3.2.1 Bayesian statistics

A popular filtering framework stems from Bayesian statistics [5]. This filtering method is called Recursively Bayesian estimation. The distribution of interest is the posterior distribution

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) = p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{z}_{1:k-1}) \tag{3.1}$$

of a state $\mathbf{x}_k$ at time k, given a set of measurements, $\mathbf{z}_{1:k} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_k\}$. Bayes' theorem shows how the conditional probability can be derived as

$$P(\mathrm{A}|\mathrm{B}) = \frac{P(\mathrm{B}|\mathrm{A})P(\mathrm{A})}{P(\mathrm{B})} \tag{3.2}$$

where A and B are events and P(A) and P(B) are probabilities for the respective event to occur. This theorem can describe the distribution of interest as

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{z}_{1:k-1})p(\mathbf{x}_k|\mathbf{z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})} \propto p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{z}_{1:k-1})p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) \tag{3.3}$$

where $p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{z}_{1:k-1})$ can be simplified assuming that sensor noise is independent over time such that $\mathbf{z}_{1:k-1}$ can be disregarded in the expression. The remaining expression $p(\mathbf{z}_k|\mathbf{x}_k)$ describes the likelihood function, or in filtering terms the sensor model. The $p(\mathbf{x}_k|\mathbf{z}_{1:k-1})$ describes the prior distribution of $\mathbf{x}_k$ and can be obtained by marginalising over previous time step $\mathbf{x}_{k-1}$ according to

$$p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})d\mathbf{x}_{k-1}. \tag{3.4}$$

A common assumption is that the object dynamics follow a Markov model, where knowledge of the previous state $\mathbf{x}_{k-1}$ is sufficient to describe the future state $\mathbf{x}_k$, i.e., $p(\mathbf{x}_k|\mathbf{x}_{1:k-1}) = p(\mathbf{x}_k|\mathbf{x}_{k-1})$. This assumption gives that the simplifications in

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{x}_{k-2}, ..., \mathbf{x}_1, \mathbf{x}_0) = p(\mathbf{x}_k|\mathbf{x}_{k-1}) \tag{3.5}$$

are valid. The resulting distribution $p(\mathbf{x}_k|\mathbf{x}_{k-1})$, are in filtering terms recognized as the transition model or motion model.

The posterior distribution of $\mathbf{x}_{k|k}$ is proportional to the likelihood multiplied with the prior distribution as shown in Equation (3.3). These equations is fundamental for popular filtering methods, such as Kalman filter.

3.2.2 The Kalman filter

Based on equations presented in Section 3.2.1, the Kalman filter equations are derived under the assumption that the transition and measurement models are linear and Gaussian. The filter estimates a mean and a variance of the Gaussian distribution, represented as state vector $\hat{\mathbf{x}}_{k|k}$, and covariance matrix $\mathbf{P}_{k|k}$ at time k.

Derived from Equation (3.4), the prediction step is formed as

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{A}_{k-1}\hat{\mathbf{x}}_{k-1|k-1} \tag{3.6}$$

$$\mathbf{P}_{k|k-1} = \mathbf{A}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{A}_{k-1}^T + \mathbf{Q}_{k-1} \tag{3.7}$$

where the $\mathbf{A}_{k-1}$ is the linear transition matrix which captures the dynamics of the estimated states. $\mathbf{Q}_{k-1}$ is covariance of the process noise which captures the event of model miss match between the real system and the transition matrix by increasing the covariance $\mathbf{P}_{k|k-1}$.

The update step of the filter includes

$$\mathbf{v}_k = \mathbf{z}_k - \mathbf{H}_k\mathbf{x}_{k|k-1} \tag{3.8}$$

$$\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k \tag{3.9}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^T\mathbf{S}_k^{-1}. \tag{3.10}$$

Here, $\hat{\mathbf{z}}_k$ is the innovation i.e. the error in the predicted state with respect to the measurement. $\mathbf{S}_k$, is the innovation covariance, and $\mathbf{K}_k$ is the optimal Kalman gain. The $\mathbf{H}_k$ is the sensor model of the system. The update results in a posterior according to

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\mathbf{v}_k \tag{3.11}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}. \tag{3.12}$$

This filter is running iteratively and updates at discrete time steps, whenever new measurements can be used to update the state.

3.2.3 Non-linear Kalman filter

As describe above, the Kalman filter is used when we have linear and Gaussian models. However, it is often the case that non-linear models are necessary. The non linearity can occur if either or both the motion and the measurement model are nonlinear. A non-linear system can be described as

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}) + v_k \tag{3.13}$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + w_k \tag{3.14}$$

where $\mathbf{x}_k$ represents the state and v_k and w_k are process- and measurement noises. The noise assumed to be additive white Gaussian, with a zero mean and $\mathbf{Q}_{k-1}$ and $\mathbf{R}_k$ covariances. Here, f is the nonlinear prediction function and h describes the transition from state space to measurement state.

When filtering with non-linear models, the normal Kalman filter can no longer be used. A way to still be able to use Kalman-like algorithms is to perform linearization. This can be done either analytically, giving the extended Kalman filter (EKF) [6], or statistically, giving the Unscented Kalman filter (UKF) [7].

The Extended Kalman Filter

Extended Kalman filter (EKF) solves the nonlinear filtering problem, by differentiating the nonlinear functions. In order to predict the covariance matrix, the prediction function is linearized according to

$$\bar{\mathbf{F}}_{k-1} = \left.\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k-1|k-1}}. \tag{3.15}$$

Using the linearized prediction function $\bar{\mathbf{F}}_{k-1}$, the prediction step of the EKF is governed by

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}) \tag{3.16}$$

$$\mathbf{P}_{k|k-1} = \bar{\mathbf{F}}_{k-1}\mathbf{P}_{k-1|k-1}\bar{\mathbf{F}}_{k-1}^T + \mathbf{Q}_k. \tag{3.17}$$

Note that the prediction of state vector is equal to the normal Kalman prediction, with the exception that transition function is non-linear. For the update step, the measurement model must be linearized in order to update the covariance prediction. The update step is

$$\mathbf{v}_k = \mathbf{z}_k - h(\hat{\mathbf{x}}_{k|k-1}) \tag{3.18}$$

$$\mathbf{S}_k = \bar{\mathbf{H}}_k\mathbf{P}_{k|k-1}\bar{\mathbf{H}}_k^T + \mathbf{R}_k \tag{3.19}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\bar{\mathbf{H}}_k^T\mathbf{S}_k^{-1} \tag{3.20}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\mathbf{v}_k \tag{3.21}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\bar{\mathbf{H}}_k)\mathbf{P}_{k|k-1} \tag{3.22}$$

where $\hat{\mathbf{z}}_k$ is the innovation and $\mathbf{S}_k$ is the approximate covariance of the innovation. The $\mathbf{K}_k$ is the Kalman gain, which because of the linearization is no longer the optimal gain such as in the linear case. The sensor model is differentiated in same way as the prediction function, i.e.

$$\bar{\mathbf{H}}_k = \left.\frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k|k-1}}. \tag{3.23}$$

Unscented Kalman Filter

The unscented Kalman filter (UKF) uses the unscented transform in order to approximate linear Gaussian distribution from a nonlinear distribution. The unscented

transform approximates the mean and variance of a stochastic variable propagated through a non-linear function [8]. A common use of the unscented transform is to extend Kalman filters to non-linear function.

The transformation is done as follows. First a number of points from the probability distribution are deterministically chosen. The number of points and their location depend on the dimension of the state space and on the covariance matrix, respectively. These points are then transformed through the non-linear function. Finally, based on the transformed points, the mean and covariance matrix of the transformed random variable can be extracted.

The deterministic points are referred to as sigma points, and are given by the following equations

$$\mathcal{X}^{(0)} = \hat{\mathbf{x}}_k \tag{3.24}$$

$$\mathcal{X}^{(i)} = \hat{\mathbf{x}}_k + \sqrt{\frac{n}{1-W_0}}\mathbf{P}_i^{1/2} \tag{3.25}$$

$$\mathcal{X}^{(i+n)} = \hat{\mathbf{x}}_k - \sqrt{\frac{n}{1-W_0}}\mathbf{P}_i^{1/2}. \tag{3.26}$$

Here n is the dimension of the state space and W_0 is a design parameter. A suitable value for the W_0 is $1 - n/3$ for Gaussian densities.

The UKF uses the sigma points, Equations (3.24) - (3.26), with corresponding weights to create the prediction equations,

$$\hat{\mathbf{x}}_{k|k-1} \approx \sum_{i=0}^{2n} f(\mathcal{X}_{k-1}^i)W_i \tag{3.27}$$

$$\mathbf{P}_{k|k-1} \approx \mathbf{Q}_{k-1} + \sum_{i=0}^{2n}(f(\mathcal{X}_{k-1}^i) - \hat{\mathbf{x}}_{k|k-1})(\mathcal{X}_{k-1}^i) - \hat{\mathbf{x}}_{k|k-1})^T W_i \tag{3.28}$$

and the update equations

$$\hat{\mathbf{z}}_{k|k-1} \approx \sum_{i=0}^{2n} h(\mathcal{X}_k^i)W_i \tag{3.29}$$

$$\mathbf{P}_{xz} \approx \sum_{i=0}^{2n}(\mathcal{X}_k^i - \hat{\mathbf{x}}_{k|k-1})(h(\mathcal{X}_k^i) - \hat{\mathbf{z}}_{k|k-1})^T W_i \tag{3.30}$$

$$\mathbf{S}_k \approx \mathbf{R}_k + \sum_{i=0}^{2n}(h(\chi_k^i) - \hat{\mathbf{z}}_{k|k-1})(h(\chi_k^i) - \hat{\mathbf{z}}_{k|k-1})^T W_i \tag{3.31}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{P}_{xz}\mathbf{S}_k^{-1}(\mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1}) \tag{3.32}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{P}_{xz}S_k^{-1}\mathbf{P}_{xz}^T \tag{3.33}$$

where $\mathbf{z}_k$ is the measurement. The covariance matrix $\mathbf{P}_{xz}$, is the approximated state-measurement cross covariance.

3.3 Data association

Data association considers the determining of which measurements that originate from which objects. The association problem applies not only when tracking multiple objects, but also when tracking a single object because of the occurrences of true measurements together with false alarms (clutter). Here, we will describe methods for data association. The problem of data association is to associate a set of detections with one or several existing tracks. In order to decrease complexity, gating is typically used, where some of the measurements are excluded from the association to a certain track, based on them being unreasonably far away from the prediction.

Association methods

Data association methods in tracking applications are mainly used to associate detections to tracks. Given a set of measurements $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, ...\mathbf{z}_n\}$ and tracks $T = \{T_1, T_2, ..., T_m\}$, the data association methods forms hypotheses of all association combinations that can be formed. By combining a track T_i to every measurements, n hypotheses are formed. Additionally, one hypothesis is added to represent that the track has not been associated, i.e. in total $n + 1$ hypotheses are formed. The association is then performed using either hard or soft decisions. Hard association means that only one detection can be associated to one target, while soft association means that a weighted combination of detections can be associated to an object. Nearest neighbour (NN), global nearest neighbour (GNN) and multiple hypothesis tracking (MHT) are methods of hard decisions. Further, commonly used soft association methods are probabilistic data association (PDA) and joint PDA (JPDA). There is also versions of MHT that merges hypotheses, which makes it a soft algorithm as well.

The MHT does not make a decision instantaneously. Instead, more data at subsequent scans are collected, before a decision is made. It is hence a deferred decision logic, which helps to resolve measurement to track ambiguities. Since many association hypotheses are kept, it is a multiple hypothesis approach.

Both NN and GNN are sometimes referred to as single hypothesis tracking. They are thus special cases of an MHT. For every data set, the goal is to sequentially identify the most likely assignment of measurements to tracks. The difference of NN and GNN is that the NN is a local algorithm, where each track can select its closest detection, regardless of the other tracks. The GNN, on the other hand, find the globally nearest neighbours, without conflict. The flaw of NN is hence that one detection can be associated to two different tracks, which is not possible in GNN.

The PDA is a statistical solution to the association problem, where the PDA finds the detection with highest probability to be associated to the track. JPDA is a modified version of the PDA and is performing better when there are multiple targets present [4]. In these filters, multiple hypotheses are formed after each scan of data. The hypotheses are then combined before processing the new scan of data. Both PDA and JPDA has the property of merging tracks that are close together, which is referred to as the track coalescence problem. One solution to this is given by the so-called set JPDA filter [9].

Gating

A gate is used to eliminate detections that are unlikely to stem from a track. All observations that falls within the gate, satisfies the gating relationship. The gate is formed around the predicted measurement, and all measurements that satisfy the relationship are candidates for the track update. The gating may be done in several ways but rectangular and ellipsoidal gates are two solutions [4].

A third way to gate is by using the statistical distance that is the distance between two statistical objects $\mathbf{x}_1$ and $\mathbf{x}_2$. By using Mahalanobis' distance [10], the distance is calculated as

$$d_M(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T \mathbf{S}^{-1} (\mathbf{x}_1 - \mathbf{x}_2)} \tag{3.34}$$

where $\mathbf{x}_1$, $\mathbf{x}_2 \in \mathbb{R}^{N \times 1}$, $\mathbf{S} \in \mathbb{R}^{N \times N}$. The distance, d_M, is comparable to the cumulative chi squared distribution, which enables the possibility of finding a threshold value that is based on the probability that the two compared components are the same.

3.4 Track handling

Track handling is used to initiate, maintain and delete tracks. To do initiation of a track, data must fulfill some suiting criteria, e.g. some binary tests, and then an initial state for the track is formed. The track maintenance includes the target tracking parts such as filtering and association. Another common part of track handling is splitting and merging tracks, where a split occurs when one object spawns from another object. The third function, deletion, is basically that the system keeps the track a defined time after the last measurement, associated to the track, occurred. After that it can be deleted [11].

M/N-logic for initialization and deletion

A criterion for initializing new tracks is M/N-logic. Using three values, $\{N_1, N_2, M\}$, the M/N-logic initializes a track when the criterion N_1/N_1 and N_2/M is fulfilled. It means that the track is initialized if N_1 out of N_1 and then N_2 out of M following measurements are inside the gate of a predicted track. The track deletion is similar to the initiation but consists of that it must be N out of M following measurements inside the gate. The N and M parameters do not necessary need to be the same in the initialization as in the deletion [11].

Sequential probability ratio test

Instead of using a binary test when deciding if the data from sensors are correct, using sequential probability ratio test (SPRT) [12] can be useful. SPRT, given two hypotheses, calculates the posterior probability ratio and compare to two thresholds given by

$$A = \frac{1-\beta}{\alpha} \tag{3.35}$$

$$B = \frac{\beta}{1-\alpha} \tag{3.36}$$

where α is the false track confirmation probability and β is the true track deletion probability. After that it decides whether it can accept one of the hypotheses or it needs more data to be able to make a decision with high enough confidence.

3.5 Finite Set Statistics

Conventional target tracking methods are based on calculating the probability density function of a state vector, which includes states of one or several objects. The position in the joint state vector gives each tracked object an identity. Another approach to the problem is to collect all possible target state vectors into a random set. This is the basis for random finite sets (RFSs) [13], from which several target tracking methods have been derived. The goal, then, is to calculate the probability density function of the random finite set. However, due to the large computational complexity, approximations are necessary. One such approximation is to represent the probability density function with its first-order moment, which is referred to as the probability hypothesis density (PHD). The PHD describes the intensity of targets in space. That is, for each point in space, the PHD describes the intensity of targets at that point. Instead of associating and ranking each object to one measurement according to a list of possible associations, PHD associates all measurements with all objects simultaneously.

3.5.1 Gaussian Mixture Probability Hypothesis Density filter

Under the assumption that the targets dynamics and birth processes are Gaussian, a Gaussian mixture PHD (GM-PHD) filter is shown to be applicable in [14]. The Gaussian mixture is used to approximate the RFS in order to make it possible to implement as a solution to the tracking problem. The tracked objects of the PHD filter have no identity. A detailed description of the GM-PHD for both linear as well as nonlinear cases is covered in [15]. A short summary follows in this section.

The Gaussian mixture describes the posterior intensity function by a set of Gaussian components. These components consists of the state vector of the tracked objects and are described by the mean, $\mathbf{m}_k$, of the Gaussian component. The corresponding uncertainties are described by covariances, $\mathbf{P}_k$. The Gaussian mixture components also have weights w_k that describe the number of objects each of them represents, such that if the weight of one component is 1, the component describes one object. The Gaussian mixture PHD filter described in this section based on on the implementation found in [15], and a simplification of that algorithm is presented in Algorithm 1. The GM-PHD presented in this chapter is a point target tracker

Algorithm 1: Gaussian Mixture PHD filter

Data: Gaussian components described by $\mathrm{w}_{k-1|k-1}$, $\mathbf{m}_{k-1|k-1}$, $\mathbf{P}_{k-1|k-1}$ and set of measurements $\mathbf{Z}_k$

1: Add birth components ($\mathbf{m}_{k,\gamma}$,$\mathbf{P}_{k,\gamma}$ and $\mathrm{w}_{k,\gamma}$) to the set of Gaussian components $\mathbf{m}_{k-1}$.

2: Predict existing components from previous time $k-1$, add to the set of Gaussian components

$$\begin{aligned}\mathbf{m}_{k|k-1} &= \mathbf{A}_{k-1}\mathbf{m}_{k-1|k-1},\\ \mathbf{P}_{k|k-1} &= \mathbf{A}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{A}_{k-1}^{T} + \mathbf{Q}_{k-1},\\ \mathrm{w}_{k|k-1} &= p_S \mathrm{w}_{k-1|k-1}.\end{aligned}$$

3: Construction of PHD update components $\mathbf{K}_k$, $\mathbf{S}_k$ and the Gaussian component covariance $\mathbf{P}_{k|k}$.

4: Missed detection update, $\mathbf{m}_{k|k} = \mathbf{m}_{k|k-1}$, $\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1}$ and $\mathrm{w}_{k|k} = p_D \mathrm{w}_{k|k-1}$.

5: For each $\mathbf{z} \in \mathbf{Z}_k$, measurement update of all components

$$\begin{aligned}\mathrm{w}_{k|k} &= p_D \mathrm{w}_{k|k-1}\mathcal{N}(\mathbf{z}_k; \mathbf{H}_k\mathbf{m}_{k|k-1}, \mathbf{S}_k),\\ \mathbf{m}_{k|k} &= \mathbf{m}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_{k-1}\mathbf{x}_{k|k-1}).\end{aligned}$$

6: Normalization of weights

algorithm. The initial step of the GM-PHD filter is creating birth components described by $\mathbf{m}_{k,\gamma}$,$\mathbf{P}_{k,\gamma}$ and $\mathrm{w}_{k,\gamma}$. These birth components are supposed to handle the event of identifying new dynamic object in the set of measurements apart from the ones already identified. Components from previous time steps are predicted with a transition model, which can be linear or non-linear, and their weights is multiplied with survival probability p_S. Further, the set of predicted components are duplicated, where one half is multiplied with $(1 - p_D)$ to describe that they are not detected, hence the p_D describes the probability for object to be detected. The other half is multiplied with p_D to describe that they are detected. In the measurement update step, all predicted components are associated to all measurements. The corresponding weight of a component associated to a measurement is multiplied with the likelihood of that measurement, which is given by the innovation. The update step results in exponential growth of Gaussian components. To mitigate this together with reducing the computational complexity of the filter, gating for the PHD update is introduced in [16], which reduces the number of components yielded in the update step. The PHD does only update components and measurement if the distance between them falls below a certain threshold. The updated components are normalized such that the sum of all components represents an estimate of the number of objects in the posterior distribution. The combination of all Gaussian components represents the posterior density.

Merging and Pruning

As mentioned, gating only mitigates the exponential growth of Gaussian components. To keep the amount of components to a more reasonable number, a method is presented in [15] that merges components that lie within a certain distance between them for the entire set of Gaussian components. The merging step is followed by a step where components with a weight below a threshold is removed.

3.6 Extended target tracking

An extended target is an object that may give rise to more than one measurement per time step. In situations with high resolution sensors or/and objects within close range to the sensors, extended targets must be considered. In this section, we present two solutions that are popular in the literature for solving extended target tracking.

3.6.1 Probabilistic Multi-Hypotesis Tracker

The probabilistic multi-hypotesis tracker(PMHT) is a multi-target tracker [17]. The PMHT approach for target tracking gives an probabilistic solution to the measurement-to-track problem by associating all measurements to one track. Each measurement is weighted with an estimate of how likely it is that the measurement stem from the track. Though the PMHT filter handles the data association problem, the filter does not handle the track initiation and deletion of tracks and relies on having accurate knowledge of the number of object present in the set of measurement. In [18] the general idea is to separate the kinematic state and extension states of an object. Then, the expectation-maximization (EM) is used to iteratively optimize ellipsoids, representing the state of an object, and the kinematic states by applying a Kalman filter from the PMHT framework. The filter assumes that the kinematics of a point, the centroid of the object, and an ellipse is an valid description for the entire object.

3.6.2 PHD-filter applications for extended targets

An alternative to track extended targets are presented in [14] and [19] which are using the theory of PHD, presented in Section 3.5. There, *extended target tracking* with Gaussian Mixture Probability Hypothesis Density filter (GMPHD) and *extended target tracking* with Cardinalized Probability Hypothesis Density filter (ETT-CPHD) are described. The first paper shows how one can narrow the partition set of measurements without changing the quality of the estimations. The second paper is using CPHD which introduces the possibility to have a probability mass function of how many objects that is tracked. Notable is that both of these last mentioned approaches use laser measurements to track objects.

3.7 Clustering

If the extension of targets is not handled by the algorithm it self, clustering of data points that seem to originate from the same object is necessary. This is to avoid

having more tracks than the number of objects. The clustering of data points in a point target tracker can be done directly on detections and provide clusters of detections to the tracker. Or it can be done after tracking point targets and cluster tracks that seem to originate from the same target.

3.7.1 Hierarchical clustering

Hierarchical clustering is a way to build a hierarchy of clusters. There are two ways of performing hierarchical clustering, bottom-up or top-down. In a bottom-up algorithm, the clustering is initialized with each data points being its own cluster and then merge clusters while moving up in the hierarchy. In contrast, top-down is starting with all data points in one cluster that is split while moving down in the hierarchy. In [20], this theory is presented among the theory of single, average and complete linkage.

Single linkage is a type of bottom-up hierarchical clustering. At the bottom of the hierarchy, all data points are individual clusters. By combining the closest pair of clusters sequentially until all data points are in the same cluster. This creates a tree diagram, called dendrogram, that shows how the data points are clustered. One example of a dendrogram for 30 data points is shown in Figure 3.2. By choosing a maximum distance criterion, the big cluster are split into multiple clusters. In single linkage, the criterion is defined as the maximum allowed distance between the closest points in two clusters.

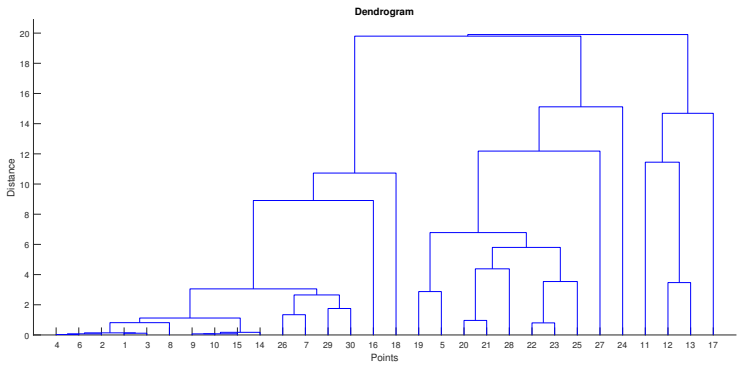


Figure 3.2: Example of a dendrogram.

There are other ways of choosing the criterion that splits the big cluster into smaller ones. By using the average point of each cluster and measure the distance between clusters, average linkage is used. If instead comparing the most distant points between clusters, complete linkage is used. These two are also types of hierarchical clustering.

4
Introduction to proposed solutions

In the literature, there are a number of algorithms developed for the extended target tracking problem. The algorithms have different characteristics which makes them more or less suitable as a solution to the problem in this thesis. In this chapter, we discuss some of the existing methods and introduce the algorithms developed in this thesis.

4.1 Drawbacks of existing methods

Extended target tracking is a area of interest by many researchers, not only by automotive industry. The existing methods developed for tracking extended targets have in many cases shown promising results but they have not always been tested in a practical situation. They have not necessarily focused on object tracking in an automotive setting either. This is important to consider when evaluating these approaches to see if they may suit for tracking in the environment considered in this thesis.

4.1.1 Ellipse-based methods

A common way to model extended objects is to use an ellipse that describes the distribution from an object. Some of the methods discussed in Section 3.5 and Section 3.6 are using ellipses to describe an object. As shown in Figure 2.3 and 2.4, the measurements are distributed along the sides of the car and not around the center. Creating an ellipse from these measurements with the center point as the mean of the measurements, will be a rough approach to represent a car. In Figure 4.1 it is illustrated that the center point of the ellipse is close to one of the sides and not in the middle. This creates ellipses that either not encloses all the measurements or encloses all but represents the object in a non accurate way.

4.1.2 Rectangle-based methods

There are methods that, instead of ellipses, model extended objects as rectangles. One such example is [21] where a PHD-based solution on rectangle-based models is presented. This approach uses accurate laser measurements and may not suit an application with radar measurements. As illustrated in Figure 4.2, the measurements do not appear along the border of a rectangle. If the radar would have produced detections along straight lines representing one or two sides of the object,

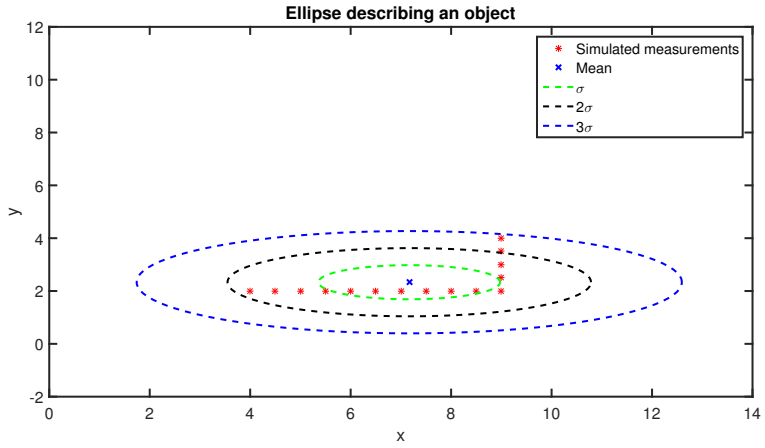


Figure 4.1: 3 ellipses created with 1-, 2- and 3-σ around a set of simulated measurements. The center point is the mean of the measurement and the ellipse is created from the covariance of the measurements.

the methods in Section 3.5 and 3.6 that models objects as rectangles would have been more appropriate to use. Instead we need to handle the multiple cases because the detections does not appear in distinct pattern.

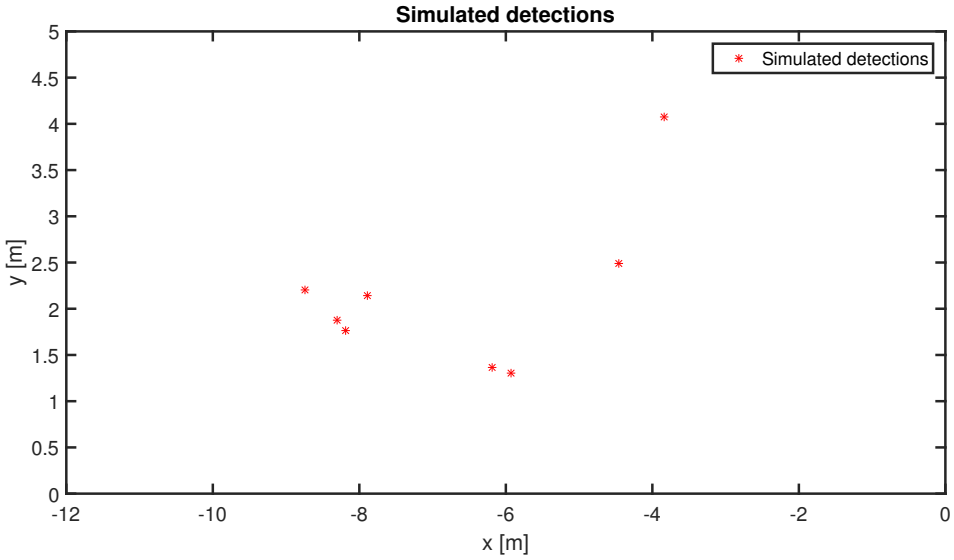


Figure 4.2: Simulated measurements of how the measurements might appear.

4.1.3 Complexity

In the automotive industry, there are often limitations on computational power and time for calculations. The process of PMHT is iterative in real time and it is not certain that the result converges in time if used in production cars. The extended target PHD and CPHD algorithms require partitioning of data and test multiple partitions of measurements to all tracks. This results in a large number of hypotheses to be handled by the algorithm and is not suitable for a production

system.

4.2 Proposed algorithms

As discussed in the previous sections, there are several practical aspects to consider when developing an extended target tracker in an automotive setting. First, when tracking vehicles, we would like to describe them as rectangles. Second, the proposed algorithms must be computationally efficient. In this thesis we present two solutions for tracking cars using radar. The first algorithm clusters the detections to represent one object per cluster and uses extended target tracking on the different clusters. The second algorithm uses target tracking on the detections, clusters the output of the tracking and then estimates the state of the object. The reason for developing two different algorithms is because after implementing a point target tracker, two potential solutions were identified. One were the detections were clustered and tracked in an extended target tracker and one were the detections were handled as point targets and then clustered. Next, we provide an brief overview of the two methods.

4.2.1 Detection-based solution

The detection-based solution starts by merging the detection from all four sensors into clusters. The clusters are associated to objects which are updated by an extended target tracking module assuming rectangular shaped objects. The non-associated measurements are used for initialization of new objects. The algorithm makes use of hard clustering and association methods in order to avoid a large number of cluster and association hypotheses. It also uses rectangular objects to represent the cars.

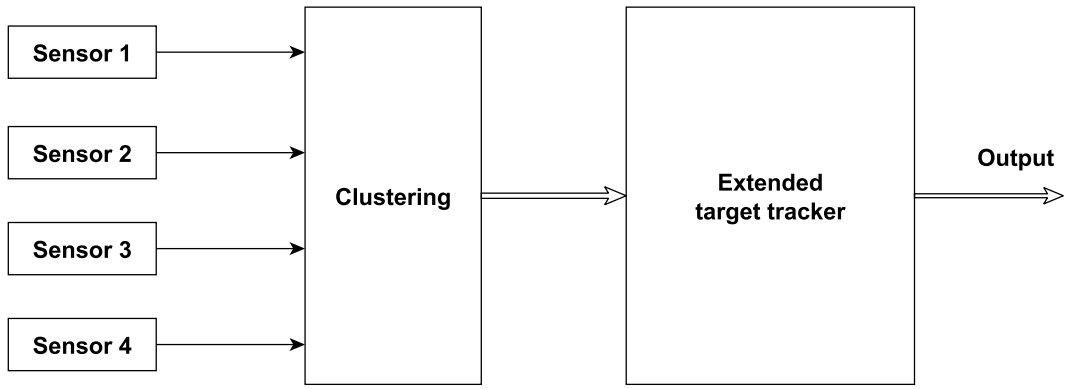


Figure 4.3: Rough block diagram of the detection-based solution.

4.2.2 PHD-based solution

The idea of having a PHD-based solution is that, even if the targets are extended, the point targets still contains information of the object. Using the estimated states

from all single point targets that originates from the extended object, an estimation of the extended target states could be extracted. The PHD-based solution runs a Gaussian mixture PHD-filter on all the measurements in order to find and track reflection points in the observed scene. The sensors may detect several reflection points per object and each detection are handled as a point target in a Gaussian Mixture PHD filter. These point targets may be found in the Gaussian mixture and are clustered to represent an object per cluster. These clusters are further used to estimate rectangular objects. The advantages of this approach is that it uses a proven point target tracking that handles targets that appear and disappear. It also uses hard decisions when clustering to avoid multiple hypotheses and a large amount of data. At last, it uses rectangular models to represent the cars.

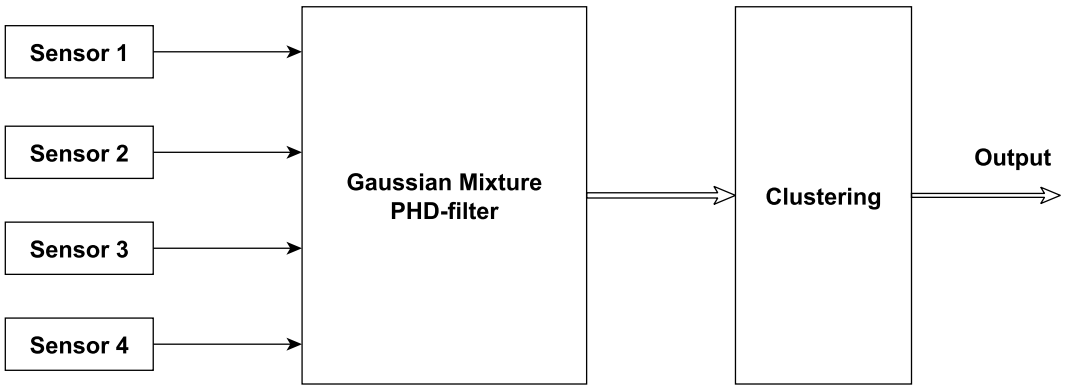


Figure 4.4: Rough block diagram of the PHD-based solution.

4.2.3 Common parts

Even though the two proposed solutions differ in general, some methods and models are applicable for both algorithms. The most central common parts are the handling of the object reference point, the sensor model, the ego motion compensation and the data preprocessing. In the next chapter, these subjects are described in detail. This to enhance the reader's knowledge of the implementations that are similar for both solutions before we discuss the individual algorithms further.

5
Modelling and data preprocessing

In both of the proposed algorithms there is a need for a sensor model and compensation of object states due to ego vehicle motion. In addition, both methods assume that the data has been processed in order to remove detections from stationary objects. In this chapter, these common parts are described.

5.1 Reference point

Since the objective is to track extended objects, finding a reference point on the tracked object is crucial in order to find e.g. position of that object. Choosing the geometrical middle point would have been an obvious choice, but since the measurements originate from the surface of the closest sides of the object, a good knowledge of the size would have been necessary to determine the middle point. The object is instead described by one of the closest sides by having eight reference points to represent the four corners and four sides of the objects as illustrated in Figure 5.1.

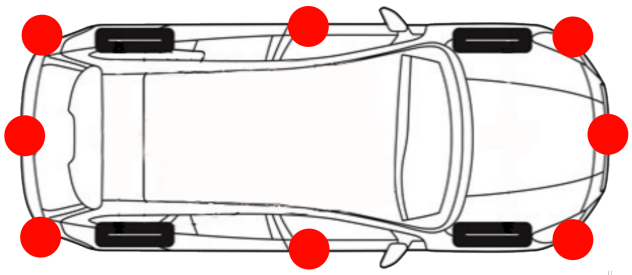


Figure 5.1: The eight reference points is illustrated by the red dots in the figure.

Figure 5.2 illustrates how the reference point is determined depending on the placement in the ego coordinate system and the heading of the tracked object. If the tracked object is heading towards or from the ego vehicle, with an deviation of

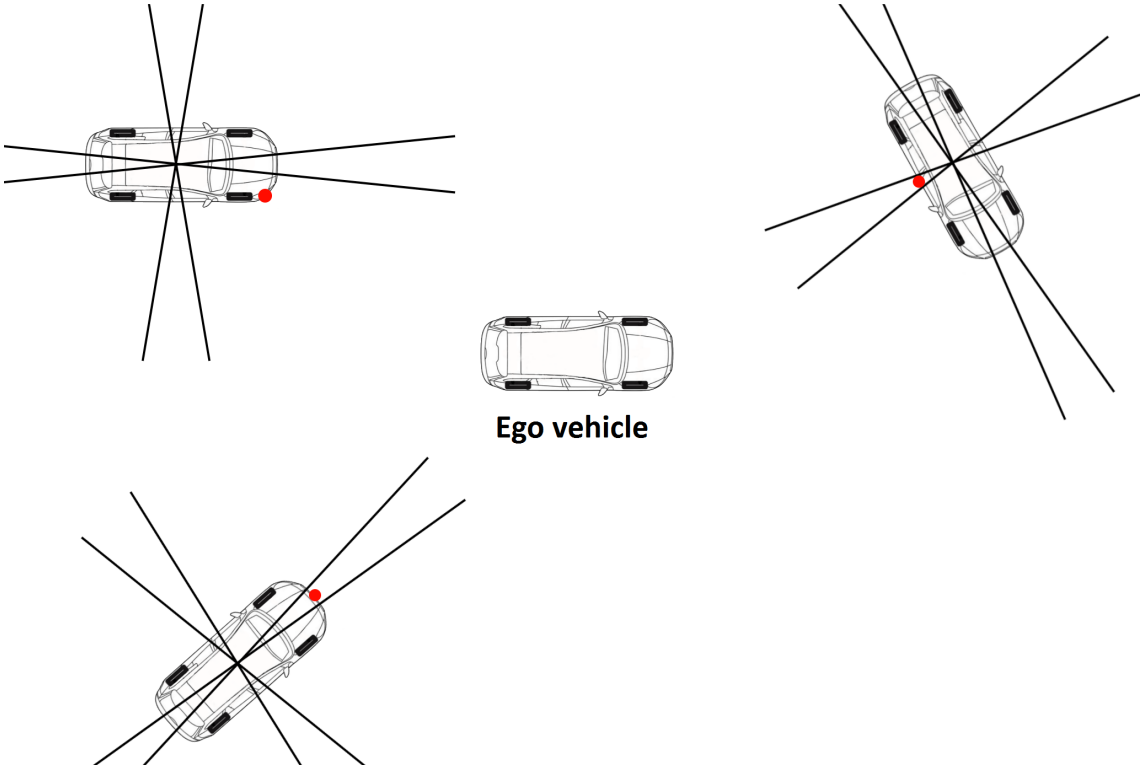


Figure 5.2: Three examples to illustrate how reference points on the tracked object are decided. All objects have velocity in the heading direction. The segments that encloses the origin of ego vehicle coordinate system, is determining the reference point.

0.1 radians, the reference point is the front or the back respectively. If the tracked objects heading is perpendicular relative the ego vehicle, with an deviation of 0.1 radians, is the reference point one the side pointing at the ego vehicle. In between, the closest corner is used as reference point. Three examples of which points that are used as reference points are shown in Figure 5.2. By doing this, one of the closest points is used as reference point and the deviation is used so that the reference point is not changed too often. The measurements often originates from the closest sides of the vehicle. Using the closest point on the target as reference point will indicate what sides of the object that the measurements originates from. With this knowledge it makes it possible to estimate the position. By using the information of length and width of the object, the reference point can be changed if another point should be used to track the object.

5.2 Ego motion compensation

Since the ego vehicle is moving, the tracking coordinate system is moving. Then, a compensation for the ego motion is required to describe the state vector and its corresponding covariance of the tracked object in the moving coordinate system. When tracking a point target, given the change from time $k-1$ to k in ego position (Δx_k^e, Δy_k^e) and heading ($\Delta\theta_k^e$), the mapping of the new state vector is done according to

$$\begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{inv}(\Delta\theta_k^e) & \mathbf{0}_{2x2} \\ \mathbf{0}_{2x2} & \mathbf{R}_{inv}(\Delta\theta_k^e) \end{bmatrix} \begin{bmatrix} x_k - \Delta x_k^e \\ y_k - \Delta y_k^e \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} \tag{5.1}$$

where the rotation is calculated by the inverse rotation matrix given by

$$\mathbf{R}_{inv}(\varphi) = \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix}. \tag{5.2}$$

When tracking an extended target, using the state vector including size and heading, the mapping is instead done according to

$$\begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \\ L_k \\ W_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{inv}(\Delta\theta_k^e) & \mathbf{0}_{2x2} & \mathbf{0}_{2x3} \\ \mathbf{0}_{2x2} & \mathbf{R}_{inv}(\Delta\theta_k^e) & \mathbf{0}_{2x3} \\ \mathbf{0}_{3x2} & \mathbf{0}_{3x2} & \mathbf{I}_{3x3} \end{bmatrix} \begin{bmatrix} x_k - \Delta x_k^e \\ y_k - \Delta y_k^e \\ \dot{x}_k \\ \dot{y}_k \\ L_k \\ W_k \\ \theta_k - \Delta\theta_k^e \end{bmatrix}. \tag{5.3}$$

Sigma points are created from the non-rotated state vector, individually rotated by Equation (5.1) and are then used to calculate a new covariance matrix from Equation (3.30). This covariance matrix is the estimated covariance of the transformed state vector.

5.3 Sensor Model

The sensor generates detections from the environment within its field-of-view. To include the sensor measurements in a tracking framework, a sensor model is required. That model describes the relationship between the state of an object and a measurement generated by the object. The sensor model h, is given by

$$\mathbf{z}_k = h(\mathbf{x}_k^D, \mathbf{x}_k^e, S^n). \tag{5.4}$$

Here, $\mathbf{x}_k^D = [x_k,\ y_k,\ \dot{x}_k,\ \dot{y}_k]^T$ and is a subset of the entire state vector, $\mathbf{x}_k$, describing the object with one point, $\mathbf{x}_k^e = [\Delta x_k^e,\ \Delta y_k^e,\ \dot{x}_k^e, \dot{y}_k^e,\ \Delta\theta_k^e]^T$ is the ego vehicle state and $S^n = \{x_s^n,\ y_s^n,\ \gamma_s^n\}$ describes the offset in position and rotation of sensor n relative to the ego vehicle reference frame.

In Figure 5.3 an example is shown of how a point target, described by the state vector $\mathbf{x}_k^D$, can be transformed to the sensor reference frame. The sensor of interest is in this illustration the front left sensor, and the Cartesian coordinate system for this sensor is described in the figure (blue in figure). The position (x_k, y_k) and the velocity ($\dot{x}_k$, $\dot{y}_k$) of the point target is relative to the ego vehicle reference frame (red in figure). Moreover, the sensor is mounted with an offset to the ego reference frame denoted as (x_s^n, y_s^n) with relative rotation described by γ_s^n.

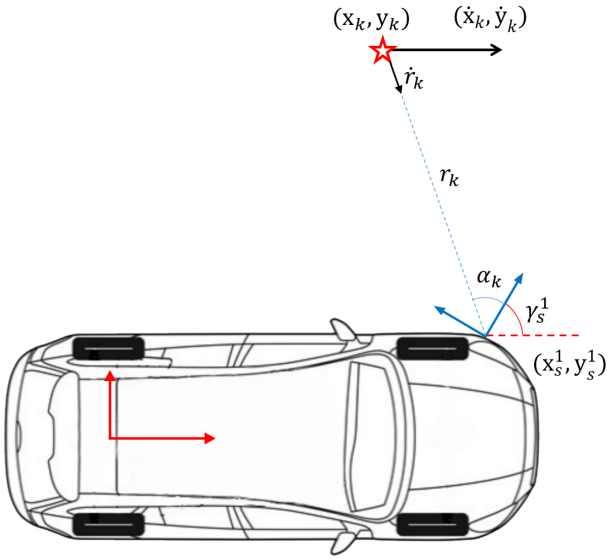


Figure 5.3: The red star represents an estimated point target, the figure shows how the relation between the measurement and the estimated states of the point target. In this particular example the sensor used is front left but the relation can be described in a similar way for all sensors.

First, we must describe the point target in the sensor Cartesian coordinate system. The positions, $(x_{k,r},\, y_{k,r})$, and velocities, $(\dot{x}_{k,r},\, \dot{y}_{k,r})$ relative to the sensor frame are described by

$$x_{k,r} = (x_k - x_s^n)\cos\gamma_s^n - (y_k - y_s^n)\sin\gamma_s^n \tag{5.5}$$

$$y_{k,r} = (x_k - x_s^n)\sin\gamma_s^n + (y_k - y_s^n)\cos\gamma_s^n \tag{5.6}$$

$$\dot{x}_{k,r} = \dot{x}_k\cos\gamma_s^n - \dot{y}_k\sin\gamma_s^n \tag{5.7}$$

$$\dot{y}_{k,r} = \dot{x}_k\sin\gamma_s^n + \dot{y}_k\cos\gamma_s^n. \tag{5.8}$$

The states given in the sensor frame are transformed from the sensor Cartesian coordinate system to the polar coordinate system. They are represented by range, angle and range rate given by

$$r_k = \sqrt{x_{k,r}^2 + y_{k,r}^2} \tag{5.9}$$

$$\alpha_k = \arctan\left(\frac{y_{k,r}}{x_{k,r}}\right) \tag{5.10}$$

$$\dot{r}_k = \frac{x_{k,r}\dot{x}_{k,r} + y_{k,r}\dot{y}_{k,r}}{r} - \left(\dot{x}_k^e\cos\left(\gamma_s^n + \alpha_k\right) + \dot{y}_k^e\sin\left(\gamma_s^n + \alpha_k\right)\right) \tag{5.11}$$

where $(\dot{x}_k^e,\, \dot{y}_k^e)$ is the ego vehicle speed relative the ground. Using Equation (5.9) - (5.11), the sensor model is formulated as

$$h(\mathbf{x}_k, \mathbf{x}_k^e, S^n) = \begin{bmatrix} r_k \\ \alpha_k \\ \dot{r}_k \end{bmatrix}. \tag{5.12}$$

Since the sensor does not capture the state of the objects perfectly, we need to take this in consideration. The radar sensors have an specified uncertainty for every measurement dimension which is modeled by multivariate measurement noise denoted w_k. The noise is described by

$$w_k = \mathcal{N}\left(\mathbf{0}_{3\times 1}, \begin{bmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\alpha & 0 \\ 0 & 0 & \sigma_{\dot{r}} \end{bmatrix}\right) \tag{5.13}$$

where σ_r, σ_α and $\sigma_{\dot{r}}$ is the specified uncertainty of range, bearing and range-rate measurements respectively. Hence, a covariance matrix $\mathbf{R}$, with the stated specification presented in Table 2.2, can be described as

$$\mathbf{R} = \begin{bmatrix} (0.1 + 0.0025r_k)^2 & 0 & 0 \\ 0 & 1^2 & 0 \\ 0 & 0 & 0.07^2 \end{bmatrix} \tag{5.14}$$

where r is the range measurement.

5.4 Data preprocessing

Since the tracking algorithms are going to track dynamic objects, we only want consider measurements that has a velocity relative to the ground, i.e. remove measurement that are assumed to originate from a static object. An estimate of velocity can be obtained by looking at range-rate of every measurements. The velocity is estimated according to

$$\hat{v}_k = \dot{r}_k + \dot{x}_k^e \cos(\gamma_s^n + \alpha_k). \tag{5.15}$$

where $\dot{r}_k$ range rate, α_k is bearing of the measurement detected by sensor n and $\dot{x}_k^e$ is ego velocity. Here, γ_s^n is the mounting angle of sensor s relative to the ego coordinate frame. If the velocity in Equation (5.15) is below 1 m/s, the detection is considered stationary and is removed from the set of measurement. The reason for removing detections up to such high velocity as $1m/s$ is that we want to make sure that we remove all static objects, including noisy detections. However, it is worth noting that a velocity close to zero also can imply that the object is traveling perpendicular relative the sensor. That is, with this approach we might remove detections not originating from stationary objects. The assumption used is that objects travelling perpendicular relative the ego vehicle will give rise to a detection by another sensor or from the same sensor but another location on the object, especially up close.

6
Detection-based solution

The idea of the detection-based solution is to use measurements from all four sensors in a central target tracking module. The tracker has three main components. First, a measurement grouping is performed by a clustering block. Second, created measurement clusters are associated with existing objects, and used for updating those objects. Third, remaining clusters are used for initialization of new objects. In Figure 6.1, a schematic overview of the proposed solution is shown. There, the main components and their interaction is visible. In the subsequent section, we will present more details about the respective areas of the solution.

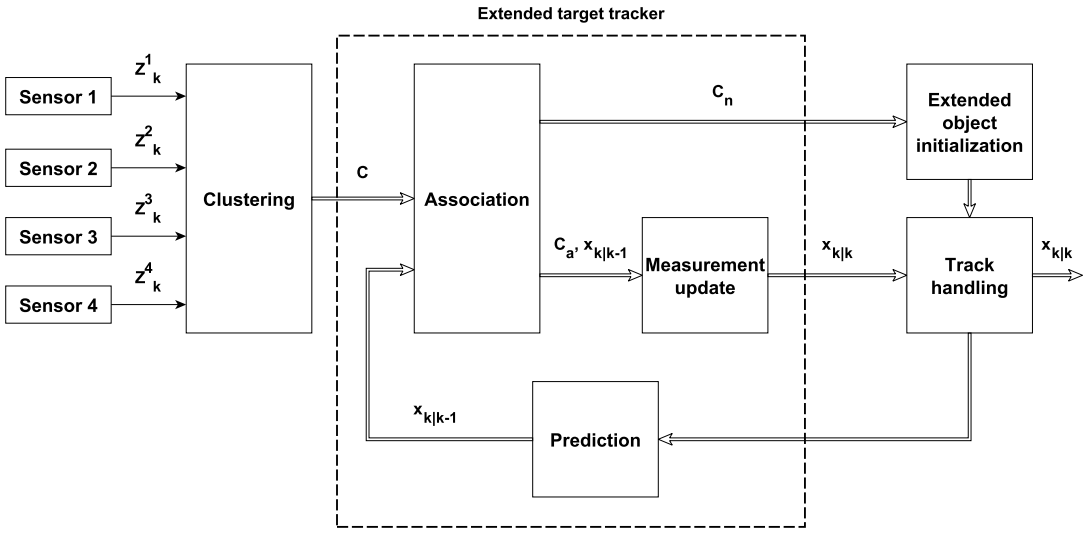


Figure 6.1: Scheme of the detection-based solution. $\mathbf{Z}_k^1$, $\mathbf{Z}_k^2$, $\mathbf{Z}_k^3$ and $\mathbf{Z}_k^4$ are measurements, C is clusters, C_a is associated clusters, C_n is non-associated clusters. $\mathbf{x}_{k|k-1}$ is the predicted state vector and $\mathbf{x}_{k|k}$ is the updated state vector.

6.1 Extended object initialization

The proposed solution uses a point tracker for initialization. The reason for this is that there is a high possibility that new objects will be detected at long ranges, meaning that they will only give rise to a few number of detections. Thus, a point target tracker is applicable. The input to the point tracker is non-associated clusters. Since the assumption of the tracker is that at most one detection can originate from

an object, the clusters are split, and all detections are handled individually. The output from the initialization block is new extended targets that are input to the extended target tracker. In the following section, we present the steps of creating a new extended object. The state vector of the point target tracker is given by

$$\mathbf{x}_k = \begin{bmatrix} x_k, & y_k, & \dot{x}_k, & \dot{y}_k \end{bmatrix}^T \tag{6.1}$$

where x_k and y_k describes the position and $\dot{x}_k$ and $\dot{y}_k$ describes the velocity. Once a point target is confirmed it is used to initialize an extended object.

6.1.1 Prediction and update

The state vector is changing over time and the change is modelled by the process model

$$\mathbf{x}_k = \mathbf{A}_{cv}\mathbf{x}_{k-1} + v_k. \tag{6.2}$$

With the assumption that the target has the same heading as the orientation of the velocity vector, the constant velocity model describes the motion of the targets such that we can get good enough information of the state of the target. Using the constant velocity model, the transition matrix, $\mathbf{A}_{cv}$, is given by

$$\mathbf{A}_{cv} = \begin{bmatrix} \mathbf{I}_{2\times 2} & T\mathbf{I}_{2\times 2} \\ \mathbf{0}_{2\times 2} & \mathbf{I}_{2\times 2} \end{bmatrix}, \tag{6.3}$$

where T is the sampling time. The process noise is modelled as Gaussian according to

$$v_k \sim \mathcal{N}(\mathbf{0}_{4\times 1}, \mathbf{Q}_{cv}) \tag{6.4}$$

where the covariance is modelled as

$$\mathbf{Q}_{cv} = \sigma^2 \begin{bmatrix} \frac{T^3}{3}\mathbf{I}_{2\times 2} & \frac{T^2}{2}\mathbf{I}_{2\times 2} \\ \frac{T^2}{2}\mathbf{I}_{2\times 2} & T\mathbf{I}_{2\times 2} \end{bmatrix}. \tag{6.5}$$

σ is a design parameter for the process noise. A track that is associated with a measurement is updated using the measurement model

$$\mathbf{z}_k = h(\mathbf{x}_k, \mathbf{x}^e, S^n) + w_k, \tag{6.6}$$

where h is given by Equation (5.12). Since the measurement model is nonlinear, the UKF update step in Equation (3.29) - (3.33) is applied.

6.1.2 Association and gating

To associate the measurements with the point targets, we use global nearest neighbour to minimize the total distance between tracks and measurements. The distance measure used is the Mahalanobis distance, which is given by

$$d_M(\hat{\mathbf{z}}_{k|k-1}, \mathbf{z}_k) = \sqrt{(\hat{\mathbf{z}}_{k|k-1} - \mathbf{z}_k)^T \mathbf{S}^{-1} (\hat{\mathbf{z}}_{k|k-1} - \mathbf{z}_k)} \tag{6.7}$$

where $\mathbf{z}_k$ is the measurement, $\hat{\mathbf{z}}_{k|k-1}$ the predicted measurement, and $\mathbf{S}$ the innovation covariance. In an UKF setting, the predicted measurement is given by

$$\hat{\mathbf{z}}_{k|k-1} = \sum_{i=0}^{2n} h(\mathcal{X}_k^i) W_i, \tag{6.8}$$

where the sigma points, $\mathcal{X}_k^i$, are given by Equation (3.24) - (3.26). Further, the covariance matrix, $\mathbf{S}$ is given by

$$\mathbf{S} = \mathbf{R} + \sum_{i=0}^{2n} (\hat{\mathbf{z}} - \mathbf{z})(\hat{\mathbf{z}} - \mathbf{z})^T W_i. \tag{6.9}$$

To limit the computational complexity, gating is applied before association. There, a gate is placed around each predicted track, and only measurements within the gate are considered for association.

6.1.3 Track handling

The track handling for point targets has three parts: track initiation, track confirmation and track deletion. The initiation and confirmation of point targets follows a two-step process. First, based on two candidate measurements, a first state vector is initiated. Then, based on prediction and filtering, a track confirmation logic is applied, where tracks fulfilling the criterion are marked as initialized and confirmed.

To initialize a track with a velocity, it is required to have the distance between two succeeding measurements less than d_{max}. d_{max} is determined by the maximum distance that is possible for an object to travel between two time steps. The maximum distance is calculated by

$$d_{max} = v_{max} T \tag{6.10}$$

where v_{max} is the maximum velocity that the algorithm tracks and T is the time step. v_{max} is in this implementation set to be 30 m/s. For the situations we are interested 30 m/s is enough to consider, but When the distance between two measurements are below the maximum distance, the initial position is set to the position of the most recent measurement and the initial velocity is set by using the velocity vector between the two measurements. A initialized track is set to be a tentative track until it is confirmed or deleted.

After a track is set to be tentative, it requires to be updated by a number of succeeding measurements to be confirmed. This is because two succeeding clutter measurements may appear close to each other and initialize a track. The track initialization uses M/N-logic to confirm a initialized track after a certain number of succeeding measurements. The confirmation used in the solution is an extension of M/N where if 2/2 + 2/3 measurements is inside the gate, the track is confirmed. If the tentative track does not get confirmed after 5 time steps, it is deleted.

6.1.4 Initiation of extended objects

When the initiation logic confirms a track, a new extended object is initiated. The initialization of the rectangular extension is done by taking the state vector of the

confirmed track and adding length, width and heading to a new state vector. The augmented state vector is then

$$\mathbf{x}_k = \begin{bmatrix} x_k, & y_k, & \dot{x}_k, & \dot{y}_k, & L_k, & W_k, & \theta_k \end{bmatrix}^T \tag{6.11}$$

where $(x_k,\ y_k)$ is the position, $(\dot{x}_k,\ \dot{y}_k)$ is the velocity, $(L_k,\ W_k)$ is the length and width respectively and θ_k is the heading. Because the tracking is limited to cars, the initial length and width is set to 5 m and 2 m respectively. That should correspond to a normal-sized car. At a long distance it is hard to determine the actual heading of the object without assuming that it is the same as the direction of the velocity vector. The initial heading is therefore set to that direction.

The position of the object describes the location of one of the 8 reference points of the rectangle (see Figure 5.1 for an illustration). The reference point depends on the position and heading of the initialized object, such that the point is the closest point of the object. If the object is traveling towards or away from the ego vehicle, the front or back is used. If the object is travelling perpendicular to the ego vehicle, one of the sides is used. For all other cases, the closest corner is used (see Figure 5.2.

6.2 Extended target tracker

In this section we describe the four parts of the proposed extended target tracker, namely gating and association, clustering, prediction and measurement update. For gating and association, the same methods are used as for point targets. The difference is that clustered detections are used instead of single detections in the association. The mean of the clustered detections is used as reference point for distance calculations. For track deletion, a target that has not been updated for 9 out of 10 time steps is deleted.

6.2.1 Clustering

The first step of the measurement update is to cluster the detections. This is necessary since an object may give rise to multiple detections, and since all four sensors are used jointly. To be able to cluster data from all four sensors, we first map the detections into the ego-vehicle coordinate system (see Figure 2.1). The clustering is done using single linkage clustering (see Section 3.7). As maximum distance criterion, the Mahalanobis distance us used, with a covariance matrix tuned to allow greater distances in the longitudinal direction. The covariance matrix used is given by

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 0.01 \end{bmatrix}.$$

This is done under the assumption that the vehicles travel on the same road as the ego vehicle, and in the same direction making the clustering allow a larger distance between points in the longitudinal direction than in the lateral direction.

6.2.2 Prediction

The dynamics of the extended object is described by the process model

$$\mathbf{x_k} = \mathbf{A}\mathbf{x}_{k-1} + v_k. \tag{6.12}$$

The prediction is using the constant velocity model for position and velocity, and the random walk model for length, width and heading. The transition model is given by

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{cv} & \mathbf{0}_{4\times3} \\ \mathbf{0}_{3\times4} & \mathbf{I}_{3\times3} \end{bmatrix}, \tag{6.13}$$

where $\mathbf{A}_{cv}$ is the constant-velocity matrix in Equation (6.3). The process noise is given by

$$\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{6.14}$$

where the covariance of the process noise is gien by

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{cv} & \mathbf{0}_{4\times1} & \mathbf{0}_{4\times1} & \mathbf{0}_{4\times1} \\ \mathbf{0}_{1\times4} & \sigma^2_{cv,L} & 0 & 0 \\ \mathbf{0}_{1\times4} & 0 & \sigma^2_{ca,W} & 0 \\ \mathbf{0}_{1\times4} & 0 & 0 & 0.001\sigma^2 \end{bmatrix}. \tag{6.15}$$

Here, $\mathbf{Q}_{cv}$ is the constant-velocity model noise in (6.5). σ is a design parameter for the measurement noise of the heading and is the same as σ for $\mathbf{Q}_{cv}$. Note that there is also process noise set on the length ($\sigma_{cv,L}$) and width ($\sigma_{cv,W}$) even though that a car always keeps its size. If no process noise on the size, there could be a possibility that the length and width converges to incorrect values.

6.2.3 Measurement update

A predicted object is updated with information from measurement clusters. As illustrated in 2.2-2.4, there is different amount of information depending on the range to the object in relation to the ego vehicle. Therefore the update is divided into two different cases. One where there are less than five measurements and one where there are five or more measurements. In both cases, a reference point is used to locate the tracked object. When the object is moving, the reference point is moved to make sure that the closest points is tracked. How this is determined is explained in Section 5.1.

Update using one measurement

Given a cluster with less than five measurements, it is hard to determine the measurements position on the object. Since we are tracking the reference point which is the closest point on the object, the measurement within the gate that is closest to the ego vehicle is used for updating the position. The predicted reference point is updated using the sensor model in (5.12). The update used is the same as in the point target tracker, i.e. the UKF update step. How the prediction might be updated is illustrated in Figure 6.2 where the position and heading is updated, but not the size.

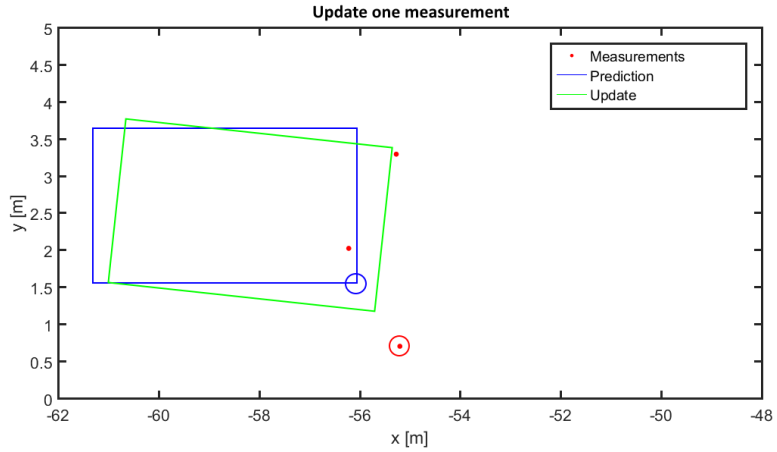


Figure 6.2: Figure showing a simulated situation when there are few measurements to update with. The circles describe which point that is updated to which measurement (blue and red respectively).

Update using multiple measurements

When more than five measurements are present, the measurements are seen as either a side of the object or two sides of the object. We determine this by fitting a line through the detections with a least mean square (LMS) condition. If the sum of the square distance divided by the number of measurements is below a threshold, and if the line has similar heading as the predicted velocity, it is viewed as a side. If not, the measurements are viewed as two sides of the vehicle.

If the measurements are viewed as belonging to one side of the object, two rectangles are created. It is assumed that the measurements represent either the length or the width of the object. If they are determined to originate from the side of the object, the width from the prediction is used for the rectangle and vice versa. This uses the assumption that the entire side is viewed in the measurements and one of the length or width is determined by that. This creates two rectangles and because only one of them actually corresponds to the object, that one will be used for updating the state. An example of this where two rectangles are created from a set of measurments can be seen in Figure 6.3.

If the measurements are considered to represent two sides of the object, multiple rectangles can be fitted to these detections. For this purpose we use a method called rotating calipers [22]. First, a convex hull of the set of points is created and then rotating calipers is applied on the convex hull. A convex hull, described bya a set of points, is formed. The convex hull is defined as the smallest polygon including all points in the set and an example of how it is created is shown in Figure 6.4a. Rotating calipers can be described as placing a caliper along one side of the convex hull, created of the measurements. The minimum box is then created by putting two sides as close as each other as possible but still containing all points. By rotating the caliper, to make one box along each side of the convex hull, all the possible minimum bounding boxes along one side are obtained. Figure 6.4b shows rotating calipers around a set of points. These set of rectangles is then used in the update of the state.

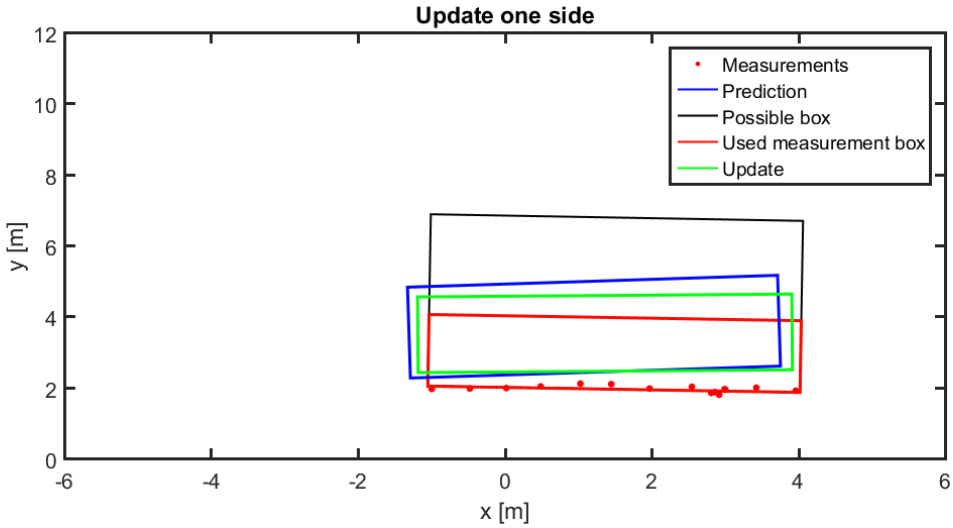
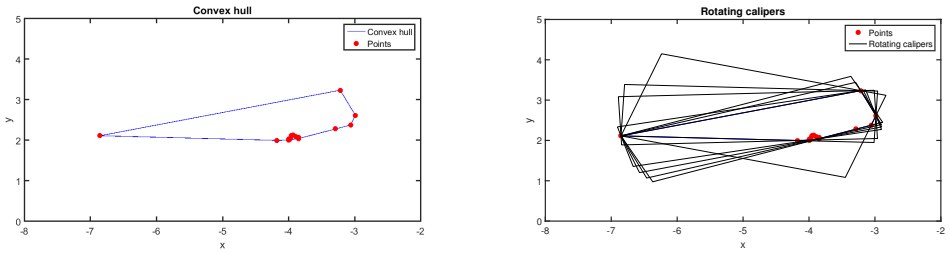


Figure 6.3: Figure showing a simulated situation when the measurements represents one side of the tracked object. The black and red boxes represents the two created boxes from measurements where the red one is used to update the state.



(a) A convex hull around a set of points.

(b) Rotating calipers around a set of points.

Figure 6.4: Example of a convex hull and rotating calipers created around at set of points.

In both cases, where we either have rectangles created from a side or from two sides, one rectangle out of the set of rectangles is used to update the state of the tracked object. The update is done by choosing the rectangle with the smallest Mahalanobis distance in position, size and heading. The velocity is disregarded because we want to handle the detections as a cluster. At close distance detections that are separated will have different range-rate even though the all detections have the same global speed. This is problematic to solve with this implementation and therefore not be investigated further. Since the information about the measurement noise is related to the detections individually, a covariance matrix for the rectangle must be created. It is given by

$$\mathbf{R}_{\text{rect}} = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 & 0 \\ 0 & 0 & \sigma_L & 0 & 0 \\ 0 & 0 & 0 & \sigma_W & 0 \\ 0 & 0 & 0 & 0 & \sigma_\theta^2 \end{bmatrix}. \tag{6.16}$$

Here, the standard deviations (σ_x,σ_y,σ_θ) and σ_L, σ_W are design parameters. The σ_L, σ_W are dependent on the range to the mean of the cluster and is therefore used to determine the uncertainty in length and width. The uncertainty of length estimation is described by the standard deviation, σ_L, of the measurement noise and is given by

$$\sigma_L = \tau_L \sqrt{x_k^2 + y_k^2} \tag{6.17}$$

where τ_L is a design parameter and is multiplied with the distance to the object. The noise of the width, σ_W, is described in the same way. Since the measurements give a better estimation of the size when the tracked objects are close, the uncertainty in size increase with the range to the object. The covariance matrix used in the calculations of Mahalanobis distance is

$$\mathbf{R} = \mathbf{R}_{\text{rect}} + \mathbf{H}_{\text{rect}} \mathbf{P}_{k|k-1} \mathbf{H}_{\text{rect}}^T \tag{6.18}$$

where $\mathbf{H}_{\text{rect}} \mathbf{P}_{k|k-1} \mathbf{H}_{\text{rect}}^T$ is the uncertainty of the prediction, transformed to the measurement space where

$$\mathbf{H}_{\text{rect}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.19}$$

The chosen box updates the prediction if the distance is below a certain threshold. The linear Kalman update given in (3.8)-(3.12) is applied to update with the created box and the measurement model as (6.19) and the measurement noise $\mathbf{R}_{\text{rect}}$.

An example of how a prediction is updated with the chosen box is seen in Figure 6.5 where one may see that the red box is created around the measurements, and the blue box is updated to the green box.

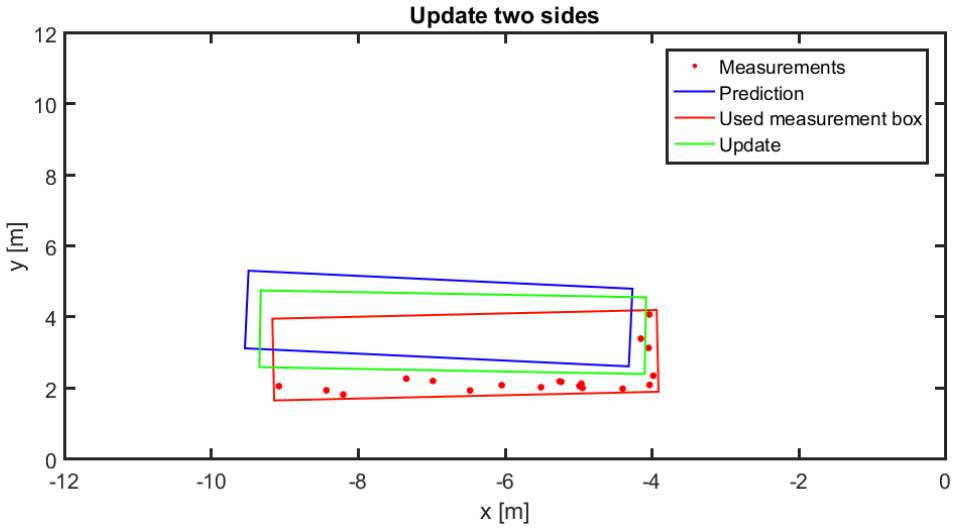


Figure 6.5: Figure showing a simulated situation when the prediction is updated with a rectangle created of measurements.

7

PHD-based solution

To avoid the complexity associated with many of the existing extended target tracking methods, the developed PHD-based algorithm relies on a point target tracker to find the position and velocity of potential reflection points on objects. By clustering tracked reflection points, the estimated position, velocity, heading and size of extended objects can be extracted. The proposed solution has four main components, shown in Fig. 7.1. First, a Gaussian-mixture PHD filter is used to track reflection points of objects. Second, the tracked reflection points are clustered. Based on the clusters, object properties are extracted. Further, an extension filter is used to be conservative in changes of length and width. In the following, each of these components are described in more detail.

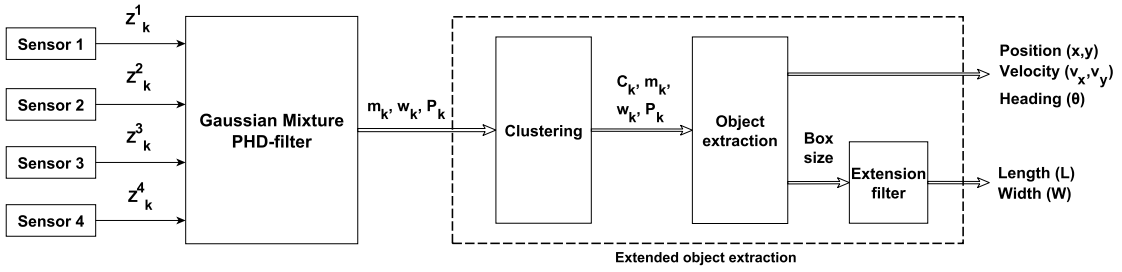


Figure 7.1: Block diagram of the proposed PHD-based extended target tracking solution. Here, $\mathbf{Z}_k^{1:4}$ is measurements from each individual sensor. The GM-PHD filter output is weights (w_k), means ($\mathbf{m}_k$) and covariances ($\mathbf{P}_k$) of the Gaussian components. The clustering index vector (C_k), describes which cluster each component belongs to. The size of the clusters are filtered in extension filter.

7.1 Gaussian mixture PHD filter

The Gaussian mixture PHD (GM-PHD) is a point target tracker, with a posterior intensity function described by a Gaussian mixture, as described in Section 3.5.1. In Fig. 7.2, we illustrate how the proposed multiple-sensor GM-PHD filter is implemented. As seen in the figure, predicted components are updated with measurements from each sensor separately. Then, in the merging step, the components from all sensor updates are combined and potentially merged.

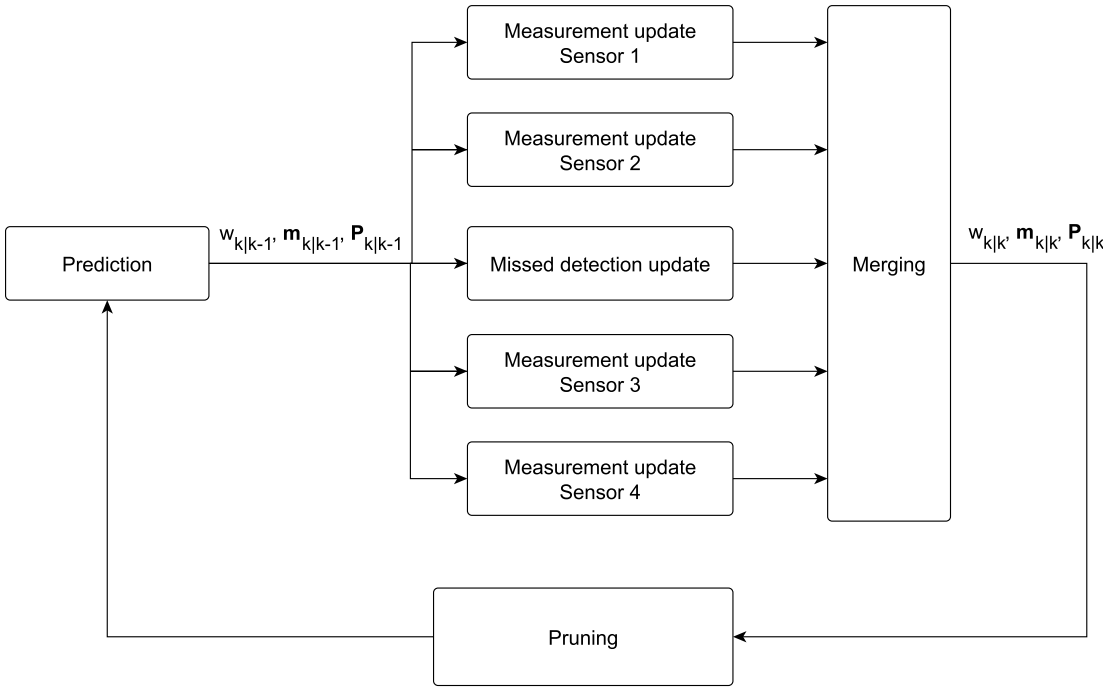


Figure 7.2: Block diagram showing the implementation of the proposed multi-sensor PHD filter.

Like in the detection-based solution, the motion model of PHD filter is decided to be a constant velocity model with states described by

$$\mathbf{x}_k = \begin{bmatrix} x_k, & y_k, & \dot{x}_k, & \dot{y}_k \end{bmatrix}^T$$

where $(x_k,\ y_k)$ represents the position of the tracked reflection points on the object. The $(\dot{x}_k,\ \dot{y}_k)$ describes the velocity of the tracked reflection points and are differentiations with respect to time of the states $(x_k,\ y_k)$. Based on what we are interested to estimate from the objects in the surrounding, a constant velocity model will be enough to describe the object if we assume that the tracked object at all time is oriented in the direction of the velocity. This is a fairly common assumption for automotive applications. The Gaussian components are described by this state vector.

7.1.1 Prediction and birth

Every Gaussian component from the previous time step is in each iteration predicted to the current time step. In addition to predictions, in each time step there is a birth of new components, described by a birth process. In the following, both prediction and birth are described for the proposed implementation.

The prediction of a component is done using the standard motion and covariance matrices for a constant velocity model, $\mathbf{A}_{cv}$ and $\mathbf{Q}_{cv}$ respectively, same as in

Section 6.1.1. The Gaussian components are predicted according to

$$\mathbf{m}_{k|k-1} = \mathbf{A}_{cv}\mathbf{m}_{k-1|k-1} \tag{7.1}$$

$$\mathbf{P}_{k|k-1} = \mathbf{A}_{cv}\mathbf{P}_{k-1|k-1}\mathbf{A}_{cv} + \mathbf{Q}_{cv} \tag{7.2}$$

$$\mathrm{w}_{k|k-1} = p_S \mathrm{w}_{k-1|k-1}. \tag{7.3}$$

Additionally, the prediction step also includes ego vehicle motion compensation, as described in Section 5.2. In order to handle initialization of new Gaussian components in the filter, birth components are added, as described in Section 3.5.1. Since the focus is to track objects travelling on the same road as the ego vehicle, the of Gaussian birth components ($\mathbf{m}_\gamma$) are located along the longitudinal axis in the tracking coordinate system, i.e. where we expect object to appear. For this implementation, two Gaussian birth components are used. Since it cannot be known beforehand in which direction and object travels, the birth components have zero velocity. The covariance for birth components ($\mathbf{P}_\gamma$) is the same for all components and is chosen to cover the every dimension where an object is expected to appear.

7.1.2 Measurement update

Since the measurement model is non-linear, the standard GM-PHD cannot be used. Two implementations for nonlinear GM-PHD filter is proposed in [15] where we chose to work with the Extended Kalman update. For such an implementation a linearized sensor model is needed. The sensor model, for each radar, given by Equation (5.12), is differentiated with respect to the states in the state vector according to

$$\mathbf{H}_k = \left.\frac{\partial h(\mathbf{x}, \mathbf{x}_k^e, S^n)}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{m}_{k|k-1}}. \tag{7.4}$$

Here, $\mathbf{x}$ is the state vector, $\mathbf{x}^e$ the ego-vehicle state, S^n the mounting specification of sensor n, and $\mathbf{m}_{k|k-1}$ the mean of a predicted component.

The implementation in [15] is adapted for one sensor, while the proposed solution uses four sensors simultaneously. Thus, the single sensor update step had to be revised such that it can handle several sensors at each time step. Each measurement from a sensors must be updated with the predicted components from previous time step and with birth components. The resulting components from each of the measurement update, are combined to one posterior intensity function.

Due to overlapping fields-of-view of the sensors, it is important change the detection probabilities, p_D, for the Gaussian components accordingly. Whenever a component is located in fields-of-view of two or more sensors, the probability of missed detection can be expressed as $(1 - p_D)^n$, where n is the number of sensors that can detect that component. In this implementation we assume that the p_D is constant for all four radar sensors, i.e. that they have equal probability of detecting an object if it is within a sensors field-of-view.

7.1.3 Merging and pruning

Since each component in the Gaussian mixture is update with multiple measurements, the number of components in the mixture grows exponentially over time.

Thus a function to limit the number of component is needed. In [15] such a function is described, which merges components with similar states and prune components with low weight. An example to illustrate this is if Mahalanobis distance between two components ($\mathbf{m}_{1,k}$, $\mathrm{w}_{1,k}$, $\mathbf{P}_{1,k}$) and ($\mathbf{m}_{2,k}$, $\mathrm{w}_{2,k}$, $\mathbf{P}_{2,k}$) is below the threshold, the two components are merged to a single component ($\mathbf{m}_k$, w_k, $\mathbf{P}_k$) according to

$$\mathrm{w}_k = \mathrm{w}_{1,k} + \mathrm{w}_{2,k} \tag{7.5}$$

$$\mathbf{m}_k = \frac{1}{\mathrm{w}_k}(\mathbf{m}_{1,k}\mathrm{w}_{1,k} + \mathbf{m}_{2,k}\mathrm{w}_{2,k}) \tag{7.6}$$

$$\mathbf{P}_k = \frac{1}{\mathrm{w}_k}(\mathrm{w}_{1,k}\mathbf{P}_{1,k}(\mathbf{m}_k - \mathbf{m}_{1,k})(\mathbf{m}_k - \mathbf{m}_{1,k})^T + \mathrm{w}_{2,k}\mathbf{P}_{2,k}(\mathbf{m}_k - \mathbf{m}_{2,k})(\mathbf{m}_k - \mathbf{m}_{2,k})^T). \tag{7.7}$$

Since the Mahalanobis distance is comparable to inverse-chi-squared distribution, we use it with four degrees of freedom and a probability $p = 0.999$ in order to obtain a threshold value for merging components. In pruning step, components with a weight lower than a threshold are removed. This threshold is also a design parameter and is tuned in order to have a good result.

7.2 Extended object extraction

The posterior from PHD filter is a Gaussian mixture composed of Gaussian components. Each of the components have an estimated set of states. The tracking concerns extended objects, hence the point target assumption the PHD filter assumes is not a valid way of describing the object. This section describes the approach of extracting information from point target objects and make extended objects using clustering and describing the extension in the 2-dimensional Cartesian space.

7.2.1 Clustering

To find the state vector describing the extended targets, the Gaussian components that originates from the same objects needs to be found. We assume, if components are closely spaced it is likely that they are describing the same object. Hence, partitioning these components into clusters is necessary. The used clustering method is single linkage clustering, similar to the clustering of measurements in the Detection-based solution.

A single linkage cluster is described such as: any component in the cluster, should have a distance to at least one other component in that cluster, which is below a threshold. Since it is known that object are travelling in same direction as the ego vehicle, and that cars typically are longer then they are wide, we would like to allow clusters to associate components with greater distance in longitudinal direction than in lateral direction in a similar fashion as detection based solution. In order to have this behaviour, the distance can be calculated with the Mahalanobis distance for the position state. With a covariance constructed as

$$\mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 0.01 \end{bmatrix}$$

which gives same distance if components are located 1 m separated in longitudinal direction, as if they are 0.1 m separated in lateral direction, i.e. the clusters are prone to allow larger extension in longitudinal direction because it allows larger separation between components. If a component can not be clustered to an existing cluster it forms a new cluster.

7.2.2 Object extraction

Having a cluster of components, each with individual states, an estimate of the states for that entire cluster can be obtained. The states that we are interested in is position, velocity, heading and size.

In order to estimate the velocity of the cluster, the velocity of all components in the cluster is regarded. The weights of the components are used to create a weighted mean of the longitudinal velocity, v_x^C, from the components in the cluster according to

$$v_x^C = \frac{1}{\sum_{j=1}^{n} \mathrm{w}_j^C} \sum_{i=1}^{n} \mathbf{m}_{v_x,i}^C \mathrm{w}_i^C, \tag{7.8}$$

where $\mathbf{m}_{v_x}^C$ is the longitudinal velocity state of Gaussian components in the cluster (C). Estimation of lateral velocity(v_y^C) is calculated in the same manner.

In this solution we have assumed that object are heading along the velocity vector i.e. no skidding or drifting. Hence, the heading (θ^C) can be expressed as

$$\theta^C = \arctan\left(\frac{v_y^C}{v_x^C}\right). \tag{7.9}$$

whenever an object is moving in the same direction as the ego vehicle. If an object travels in opposite direction we have to compensate for that arctan only is defined for angles between $-90°$ and $90°$.

A rectangle can represent an object in a spatial 2-dimensional space. The orientation of the rectangle is decided by the heading of the cluster, thus the sides of the rectangle representing the right and left side of the vehicle are oriented along with the heading. In Figure 7.3, two rectangular shapes based on simulated Gaussian components with same position but with different velocity vectors are presented. The red and blue rectangle are constructed based on the red and blue speed vector respectively. This illustrates the approach for finding the rectangle described with components. Rectangles are formed if there are more than four elements in a cluster. The length and width of the rectangle calculated and filtered in the extension filter.

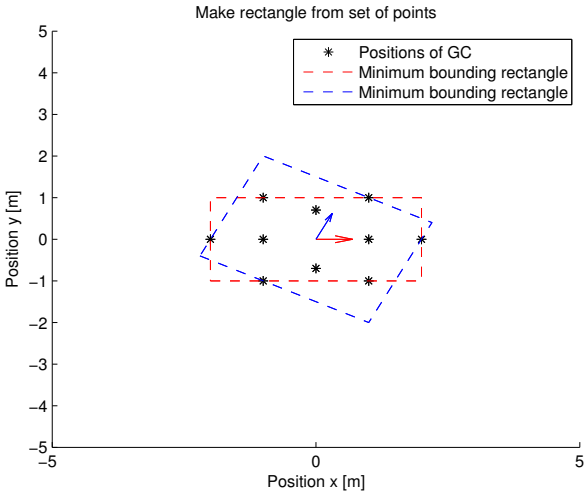


Figure 7.3: The minimum bounding box on a set of generated Gaussian components that have same positions but different velocity vectors.

It is important to have a good position estimate of the tracked object. The position of the Gaussian components are based on the measurements of the extended object and it is not clear how to use this information to describe the definite position. In this solution we choose to track reference points of the objects, described in Section 5.1. From the Gaussian components of the cluster, the weighted mean of the position state is assumed to represent the tracked reference point of the object. An example to illustrate this is, if the front of the tracked object is facing the one of sensors, it will induces detections over the surface oriented against the sensor. This will make the Gaussian components adopt similar to the spreading of measurements. Hence, the weighted mean of position states of the Gaussian components shall represent the tracked reference point.

7.2.3 Extension filtering

The information of object extension is varying due to the sensor resolution. Therefore we must consider that the entire extension of the object is not captured at each time step. Since we know that object extension is constant over time, it is possible for to utilize information over several time steps, hence a linear Kalman filter is utilized in order to estimate the size. The state vector describing the extension of the object is given by

$$\mathbf{x}_k^E = [L_k, W_k]^T \tag{7.10}$$

where L_k is estimated length of the object and W_k is the estimated width. The initial value for the filter are set to $L_0 = 5$ m and $W_0 = 2$ m, same as in the detection-based solution. A personal car, that is the objective for us to track, are typically of a size 5-by-2 m. The uncertainty is set to be high, $P_0 = \sigma^E \mathbf{I}_{2\times 2}$ where $\sigma^E = 5$. The resulting equations describing the prediction step in extension estimation is

$$\mathbf{x}_{k|k-1}^E = \mathbf{I}_{2\times 2}\mathbf{x}_{k-1|k-1}^E \tag{7.11}$$

$$\mathbf{P}_{k|k-1}^E = \mathbf{I}_{2\times 2}\mathbf{P}_{k-1|k-1}^E\mathbf{I}_{2\times 2}^T + \mathbf{Q}^E \tag{7.12}$$

where the prediction is constant, even if the the transition model of the size is true, we still need to add process noise to be able to correct the estimation.If having no process noise there is a possibility that the estimation of size is converging to a incorrect value and is not being able to adjust. That is not a behaviour that is desired, and instead we have a $\mathbf{Q}^E = \text{diag}(\sigma_{cv,L}, \sigma_{cv,W})$ where $\sigma_{cv,L}$ and $\sigma_{cv,W}$ are design parameters. The update step is

$$\begin{aligned}\mathbf{S}_k^E &= H^E \mathbf{P}_{k|k-1}^E H^E \\ \mathbf{v}_k^E &= \mathbf{x}_{k|k-1}^E - \mathbf{z}_k^E \\ \mathbf{K}_k^E &= \mathbf{P}_{k|k-1}^E \mathbf{H}_k (\mathbf{S}_k^E)^{-1}.\end{aligned}$$

The filter will only be updated when there are any information to be obtained from the cluster i.e. four components or more. Since it is not desirable to underestimate the size, updating size to a larger estimation should be easier than reversed, updating to a smaller size. To control this behavior, the variance of the measurement noise is modelled with respect to the change from prediction according to innovation ($\mathbf{v}_k^E$) where the error in length is described by $\varepsilon_{L,k}$ and width $\varepsilon_{W,k}$. In Figure 7.4 the function describing the measurement noise of the length estimation is presented. The function is a discontinuous function, modeled to have the desired behaviour of the filter. The measurement noise for width is calculated in same way.

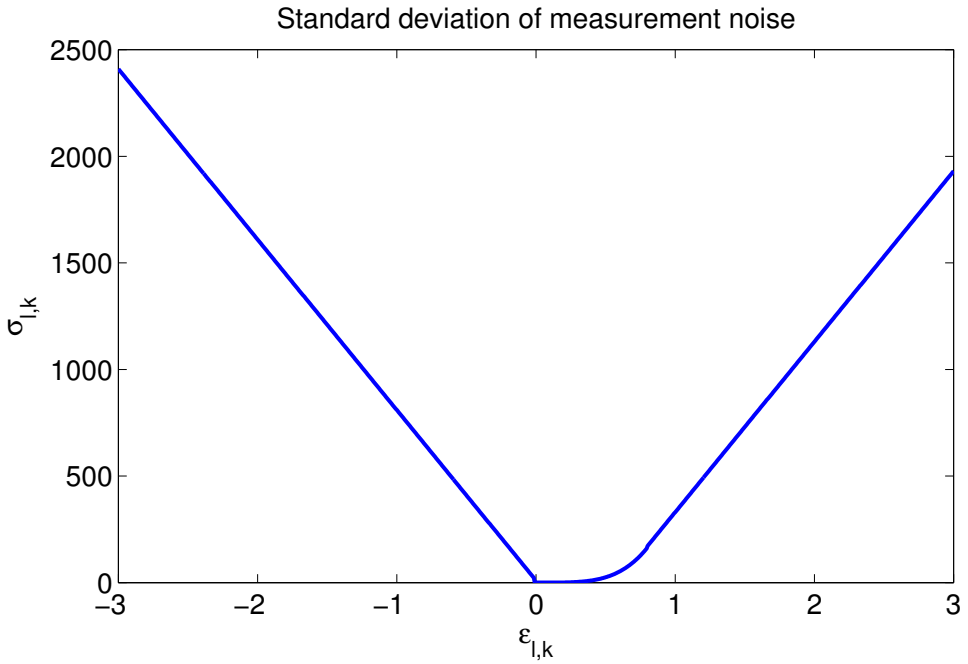


Figure 7.4: The plot shows how the standard deviation of measurement noise σ changes with the error ε_k.

8
Results

This section contains detailed results of the two algorithms developed during this thesis. The focus is on estimation of position, velocity, heading and size. The estimates are presented together with information from a reference system providing ground truth.

For the evaluation we consider an overtaking scenario involving the ego vehicle and a single target car. The target vehicle is a Volvo XC70 with length 4.838 meters and the width 1.925 meters. The target vehicle is starting at ~ 50 m from the ego vehicle and standing still. The ego vehicle is accelerating up to ~ 6 m/s and shortly after, the target is accelerating up to ~ 14 m/s. Both the ego vehicle and the target is maintaining their speed until the last couple of seconds were the target starts to decelerate. The scenario is illustrated in Figure 8.1 and this is only illustrating the longitudinal position relative to the ego vehicle.

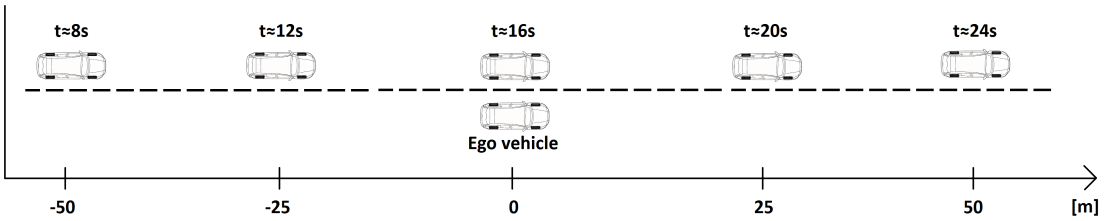


Figure 8.1: Overtaking scenario describing the longitudinal position at different time steps.

This scenario is chosen because it covers the difference appearances of measurements. It gathers measurements of the target at both long range and up close which includes the cases seen in Figure 2.2-2.4. By the overtaking, multiple sensors is detecting the target and it is detected from different angles. Moreover, the target in this scenario was equipped with a reference system, hence the scenario is suitable for evaluation.

8.1 Detection based solution

This section presents the results from running the detection-based algorithm on data from the scenario in Figure 8.1. For the evaluation, the algorithm is implemented using the parameters in Table 8.1.

Table 8.1: Design parameter choice for detection-based solution on the chosen scenario. These values are estimated based on visual inspection of the data and tuned to suit the scenario.

Parameters	
Standard deviation of process noise (σ)	1
Standard deviation of process noise for size ($\sigma_{cv,L}$,$\sigma_{cv,W}$)	0.7
Standard deviation in x,y for meas. update (σ_x, σ_y)	3
Range influence on length and width (τ_L,τ_W)	2
Standard deviation in heading for measurement update (σ_θ)	0.1

In Figure 8.2 the estimated position is presented while Figure 8.3 shows the corresponding estimation error. In Figure 8.2, we see that the estimated longitudinal position matches the ground truth very well throughout the scenario. For the lateral position, the estimates are good when the target is close to the ego vehicle. The filter initialize tracking of the target when it has accelerated for a short time, since the filter does not track stationary targets. The tracking of the target stops at approximately 28 s. That is because the target leaves the field-of-view of the sensor system.

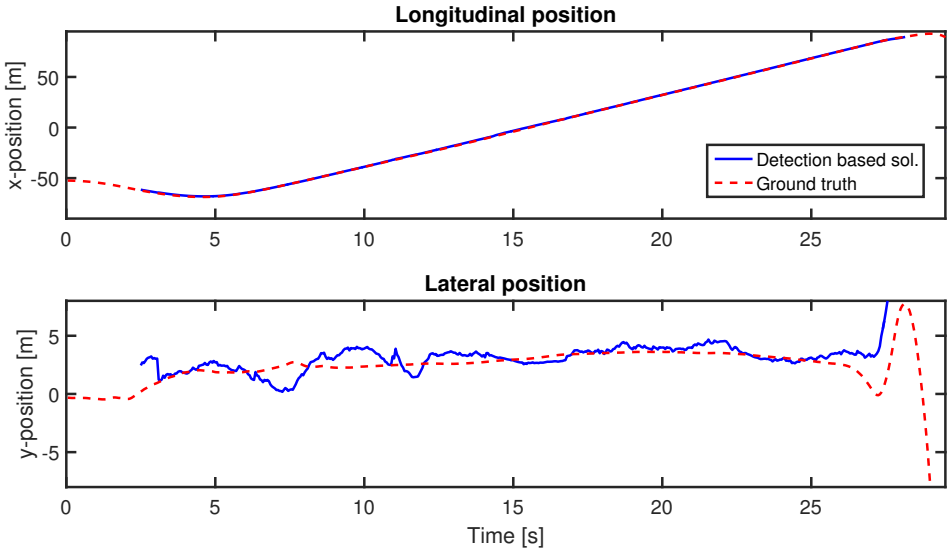


Figure 8.2: Estimated longitudinal and lateral positions compared with ground truth.

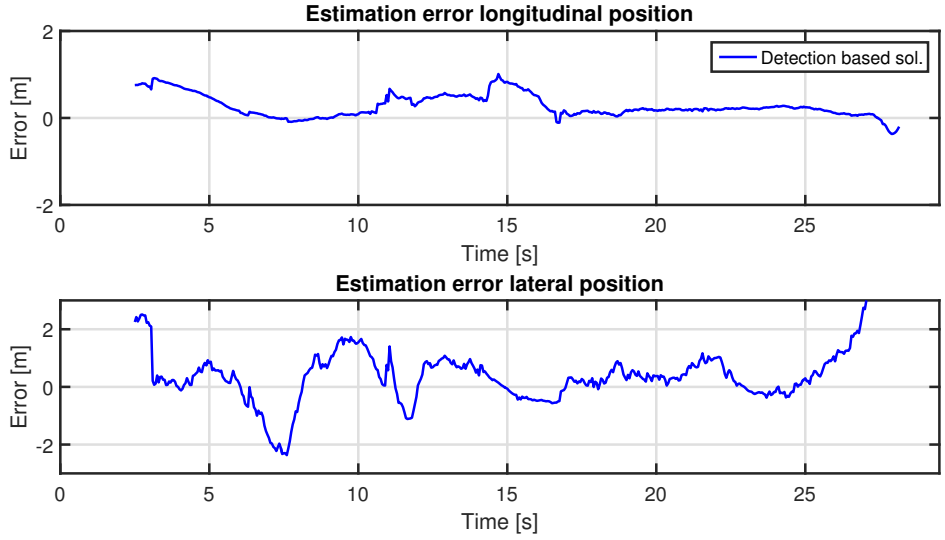


Figure 8.3: Detection-based estimation error of the position relative to ground truth.

The longitudinal velocities of the detection-based solution is close to the ground truth (see Figure 8.4). It even finds the behavior in the beginning of the acceleration (around 4s) where the acceleration dips for a short while. The lateral velocity is very unstable compared to ground truth even if it becomes more accurate after the overtaking.

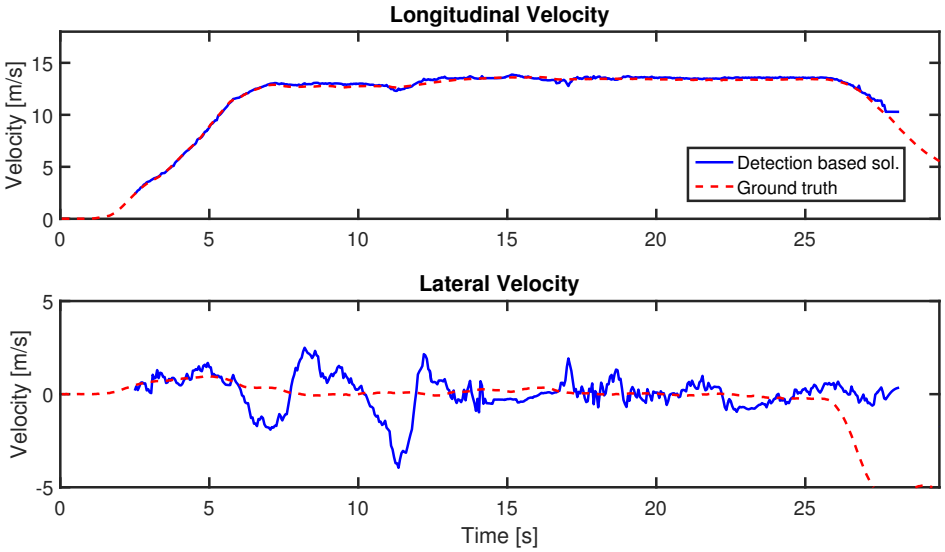


Figure 8.4: The estimated longitudinal and lateral velocities compared with ground truth.

As illustrated in Figure 8.5, the heading is velocity unstable, similar to the lateral velocity. This is because the heading is calculated from the velocity vector of the longitudinal and lateral velocity, except when it is updated with a rectangle. This is done only 2 times at around 10 s and multiple times between 14 and 17 s.

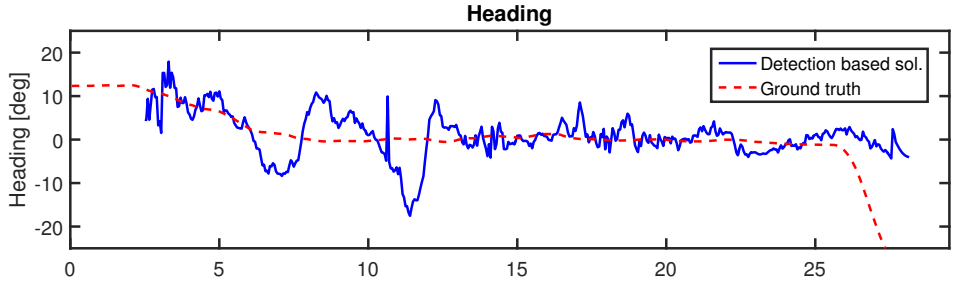


Figure 8.5: Estimated heading compared with ground truth.

As can be seen in Figure 8.6, the estimation of the extension is poor. The initial guess on length and width are 5 m and 2 m respectively. Because of the detection-based solution does not update the size from less than 5 measurements per time step, it is not updated until the target is up close. When an update is done, it is updating to a smaller size than the ground truth.

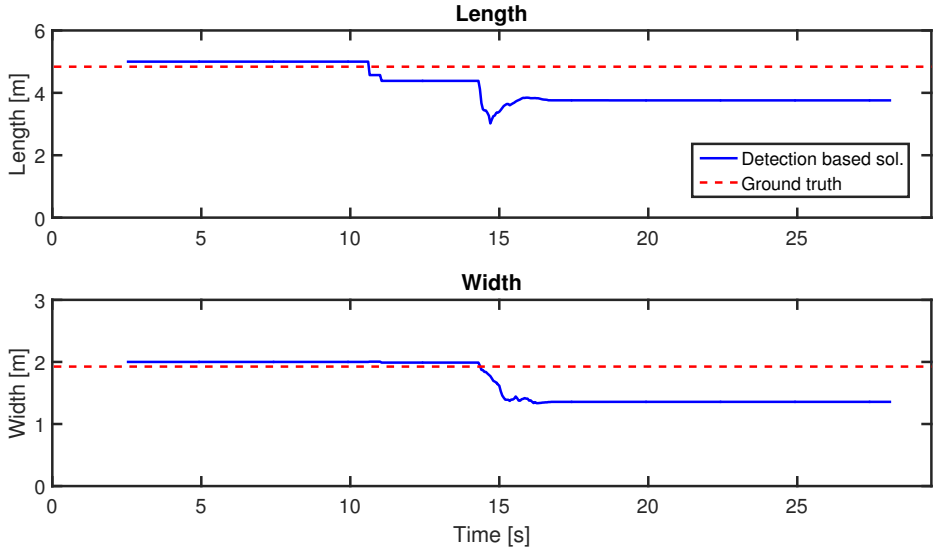


Figure 8.6: Estimated extension compared with ground truth.

8.2 PHD-based solution

The results from the PHD-filter are obtained from the same set of data as the detection-based solution i.e., the scenario in Figure 8.1. The PHD-based solution offers the possibility to tune the filter with a few parameters. The Gaussian birth components are set to have the longitudinal position as -30 m and 30 m. The corresponding covariance matrices have a value of $\mathbf{P}_\gamma = \text{diag}(100, 100, 100, 100)$. The weights for the birth components are set to $w_\gamma = 0.005$. In Table 8.2, we state the other parameter values that were implemented during the evaluation.

Table 8.2: Design parameters for the PHD-based solution in the chosen scenario. These values are estimated based on visual inspection of the data and tuned to suit the scenario.

Parameters	
Detection probability (p_D)	0.9
Survival probability (p_S)	0.95
Standard deviation of process noise (σ)	1
Standard deviation of extension ($\sigma_{cv,L}, \sigma_{cv,W}$)	0.7

In Figure 8.7 the estimated position is presented, while Figure 8.8 show the corresponding errors. As shown in the figures, the lateral position varies, at long range, as much as 4 meters in the initial phase of the tracking sequence. The longitudinal position show good estimations but during the overtaking, the reference position of the vehicle changes which introduces inaccurate estimations of position of the extended object. This causes a change in longitudinal position estimation over just one time step which is clear if looking at Figure 8.8 at time $\sim$ 15 s and $\sim$ 16 s.

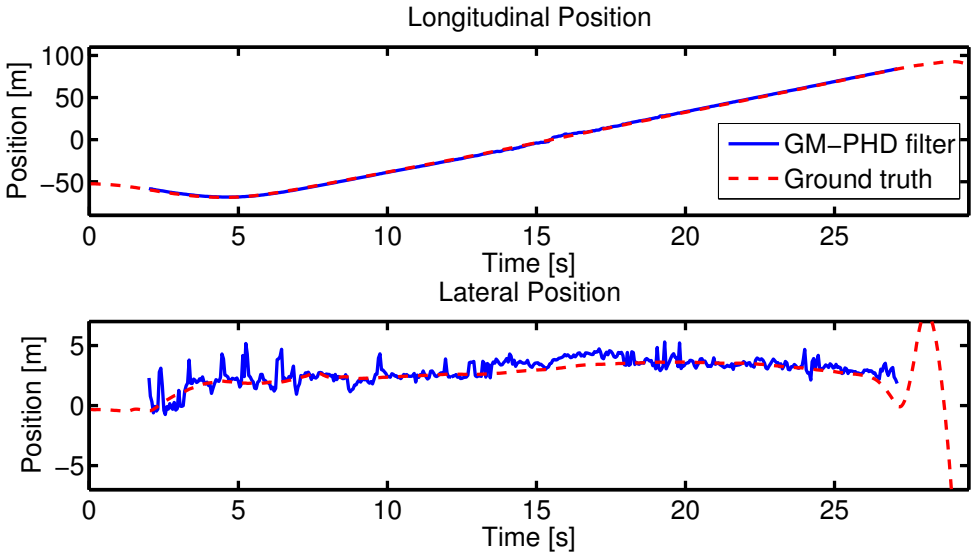


Figure 8.7: Longitudinal and lateral position compared compared with ground truth.

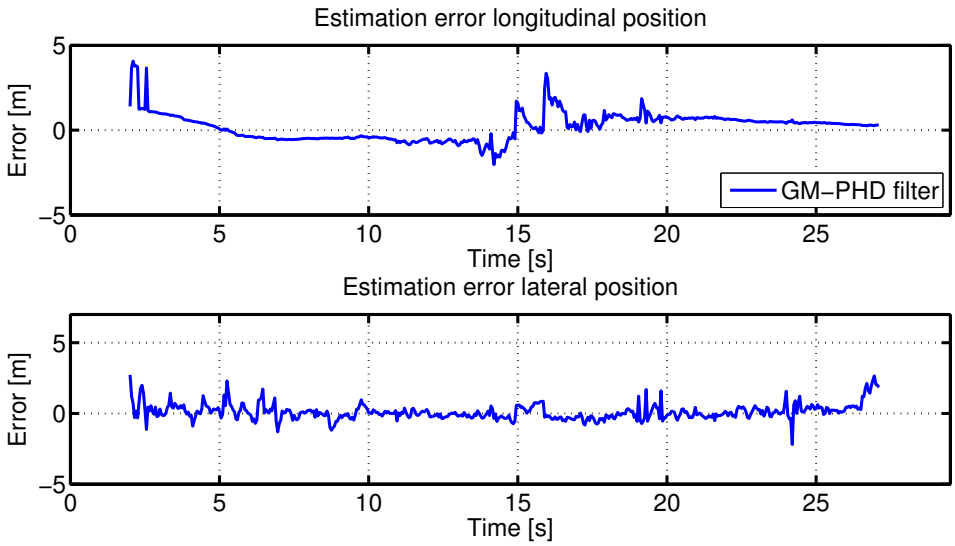


Figure 8.8: GM-PHD filter estimation error of the position relative to ground truth.

In Figure 8.9, the longitudinal and lateral velocities are presented. At long range, the longitudinal velocity is close to the ground truth. The filter only uses measurement from a target that moves in 1 m/s or more and a track is initiated almost immediately when it start to get measurements. Closer to the overtaking situation, the longitudinal speed is marginally more unstable, at this position the range-rate measurement provide less information of speed. The lateral velocity is not stable in the initial tracking sequence, but as the target approaches the ego vehicle, the lateral velocity becomes more stable, apart from a few inaccurate estimations.

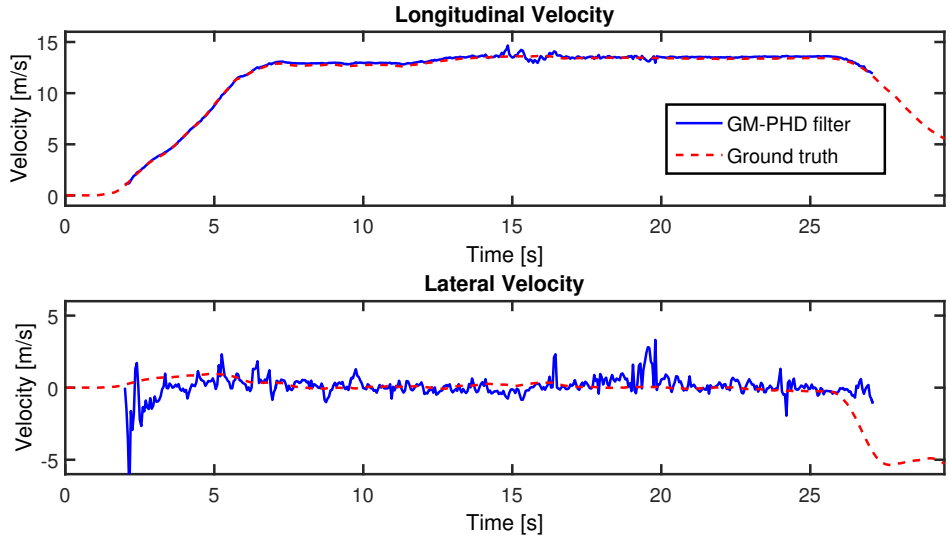


Figure 8.9: Longitudinal and lateral velocity compared with ground truth.

In Figure 8.10 the estimated heading is presented. Since the heading is estimated based on the velocity vectors, and the velocity is rather noisy, the heading

estimation is also noisy. It is unstable especially at low speeds. As illustrated in Figure 8.9, it is difficult to estimate lateral velocity when the target is far away from the ego vehicle, in combination with low longitudinal speed the heading is unstable. As the longitudinal velocity increases, the variation of heading is significantly lower.

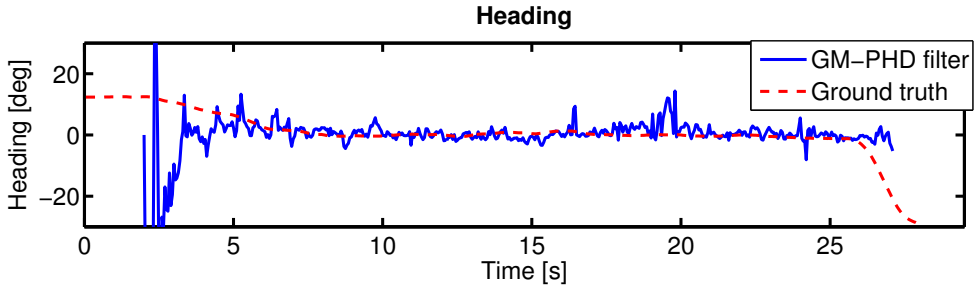


Figure 8.10: Estimated heading from velocity vectors compared with ground truth.

In Figure 8.11, the estimated extension is presented. The presented values are the output from the extension filter, which is only updated when there is enough information about the object size. Notice that the size estimation is solely based on the initial guess until 14 seconds in to the scenario, the reason for this is that the clusters does not contain enough information in order to update the extension of the object i.e. less than four components in the cluster. When there are information about the size, the estimation is quite accurate results, given that the measurement does not guarantee that the entire extension is represented in the measurement. But the fact that the estimations is based on initial guess and that the size is updated for such a short period, makes it difficult to rely of the radars for estimating an size for an autonomous vehicle.

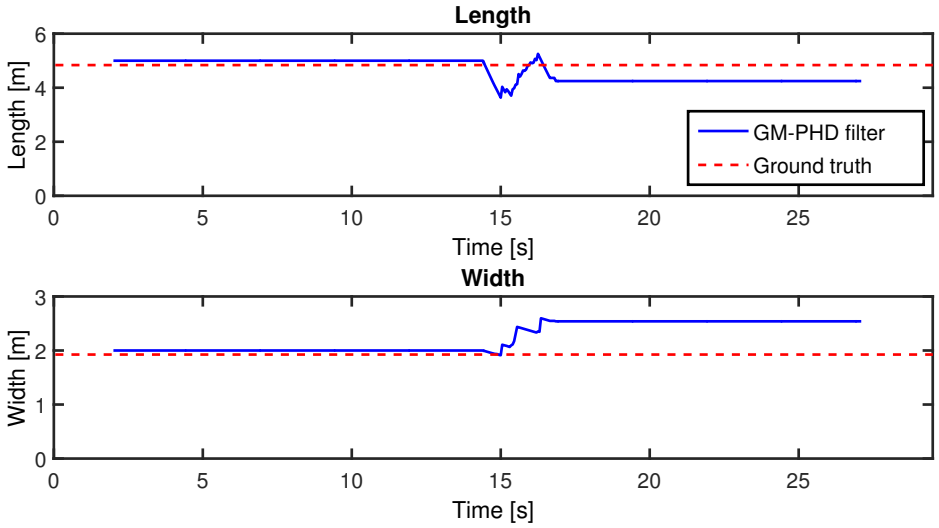


Figure 8.11: Estimated extension after linear filter compared with ground truth.

9
Discussion and conclusion

In this sections we summarize the results of the two proposed algorithms for extended target tracking. In general, we see that both approaches are feasible when it comes to tracking objects continuously over a longer period of time. Furthermore, when it comes to tracking the kinematic states, such as position and velocity, both algorithms work well. However, the estimations of the target extensions are questionable. In the following, we discuss and summarize the results, and present directions for future work.

9.1 Discussion

The detection-based solution shows promising results in general. The idea of using a point target tracker until an object is confirmed, initialize it and then track it as an extended target is working well. The PHD-based solution also show promising results. It uses a point target tracker in order to estimate the state of extended targets. The key is to extract information about the extended object from the posterior density from the GM-PHD filter.

For the scenario that we have been evaluating in this thesis, which is an overtaking scenario, both solutions are accurate when estimating the longitudinal states, including position and velocity, at long ranges. This indicates that the information of range and range-rate are accurate and give good information of longitudinal states at a distance, for an overtaking situation when an object approaches the ego vehicle on a straight road in the same or in adjacent lanes. This also gives an indication of that the single point target assumption is valid for estimating speed at long ranges. Though, close to the ego vehicle, the longitudinal states have less accurate result. When the number of measurements from an object increases, the estimation in general becomes worse.

The lateral state estimations of the detection-based solution is in general poor at long ranges. This is a problem, since at long ranges the assumption that the measurements originates from the same reflection point on the target makes the estimation to change velocity and direction. The lateral state estimation is better when the target is beside the ego vehicle, and for a short time after the overtaking. The lateral state estimation of the PHD-based solution is, in opposite to the longitudinal estimations, poor at long ranges and slightly better close to the ego vehicle. This behaviour is expected, since range and range-rate measurements are more accurate that the angular measurements.

In terms of track initiation and deletion, both solutions show similar results.

The tracking starts right after the target has a velocity of 1 m/s and drops the target when it disappears from the sensors fields-of-view. Since we do not have any restriction on how long we have to keep predicting the target, as soon it is not detectable by the sensors it is acceptable to drop the target. Since track initiation is conceptually different between the two methods, there is a difference in the start-up time of new tracks. The PHD filter starts a track directly at the first measurement, while the detection-based method has an initiation procedure that requires a number of detections. However, when it comes to the initiation rate, or detection rate, of targets, both methods are similar.

A point target assumption does not account for the extension of objects. This is problem when tracking extended objects in general. When few detections are present per object, it is difficult to determine the reflection point on the object. In our cases, it is problematic to estimate states based on that a single measurement should describe a reference point that we do not know the origin of. For instance, it can be that the algorithm uses the front right corner as reference point and updates with a detection from front left side of the car. In this case, it is interpreted as a movement in the wrong direction. Even if a measurement would be accurate, there would still be possible that we estimate states incorrectly because we do not know the origin of the detection in terms of reflection point.

The object size is initialized to 5 by 2 meters which in this case is a good estimation of the size. None of the solutions provide a robust solution for finding the size of the objects. The detection-based solution relies on that the entire extension is captured by the sensors to be able to make a rectangle of the correct size. Since accurate measurements without clutter never yield a larger rectangle than the actual size, the detection-based solution is prone to underestimate the size for all situations. In an automotive application it is important not to underestimate the size of vehicles, hence it is problematic to estimate based on this condition. Estimating size with the PHD-based solution have similar problems as the detection-based solution. However, since the components of the PHD filter spreads around the measurements, the size estimation gives a wider and longer object than what the measurements show. Since the estimation in the chosen scenario gives a shorter but wider object than the actual size, one may question the choice of creating a rectangle around all components. The extraction is tuned to decrease the error in length and width together. How to extract the size from the components is a remaining difficulty of the PHD filter.

9.2 Conclusion

It is concluded that both proposed tracking algorithms works well in terms of detecting and tracking vehicles. In terms of longitudinal position errors, the two methods are similar, with a slight advantage for the detection-based method. Further, in terms of lateral position and velocity, the errors for both methods are larger, due to poorer angular accuracy than range accuracy, and due to the fact that the sensors give few detections at long ranges. Finally, for the size estimation, both methods perform poorly. The conclusion is that there is a possibility to track objects using detections from the four considered corner radars. However, while the kinematic states appear to be possible to track with good accuracy, the estimation of target

extension is more difficult due to a limited number of detections per object. Future work is needed in order to see if other sensor models could enable better estimation, or if a more sensitive and high-performing radar is required for accurate size estimation.

9.3 Future Work

In this work we have only considered a single scenario with one vehicle traveling in the same direction as the ego vehicle. To also consider scenarios with multiple cars and with more challenging environments, e.g. intersections, would be a natural step forward. The algorithms are not tested on other scenarios but a good guess is that problems will arise. Scenarios with multiple objects would probably require more accurate clustering. Objects that are closely spaced might be clustered together and larger objects might be clustered to multiple objects. Scenarios with crossing traffic might need another or multiple motion models depending on the behavior of the objects. An approach is to use an interacting multiple model filter to cover all possible movements of a car.

Further, the implemented measurement model uses the assumption that the measurements have the same reflection point on the vehicle in each time step. Developing the measurement model to handle that the measurements are from different reflection points on the object at different time steps could be further investigated.

Bibliography

[1] National Highway Traffic Safety Administration, "U.S. department of transportation releases policy on automated vehicle development." `http://www.nhtsa.gov/About+NHTSA/Press+Releases/U.S.+Department+of+Transportation+Releases+Policy+on+Automated+Vehicle+Development`, May 2013.

[2] T. Litman, "Autonomous vehicle implementation predictions," *Victoria Transport Policy Institute*, vol. 28, 2014.

[3] R. H. Rasshofer and K. Gresser, "Automotive radar and lidar systems for next generation driver assistance functions," *Advances in Radio Science*, vol. 3, pp. 205–209, 2005.

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

[5] C. P. Robert, *The Bayesian choice*. Springer-Verlag, 2001.

[6] A. H. Jazwinski, *Stochastic processes and filtering theory*. Courier Corporation, 2007.

[7] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *AeroSense'97*, pp. 182–193, International Society for Optics and Photonics, 1997.

[8] J. K. Uhlmann, *Dynamic map building and localization: New theoretical foundations*. PhD thesis, University of Oxford, 1995.

[9] L. Svensson, D. Svensson, M. Guerriero, and P. Willett, "Set JPDA filter for multitarget tracking," *Signal Processing, IEEE Transactions on*, vol. 59, pp. 4677–4691, Oct 2011.

[10] P. C. Mahalanobis, "On the generalised distance in statistics," in *Proceedings National Institute of Science, India*, vol. 2, pp. 49–55, Apr. 1936.

[11] Y. Bar-Shalom and W. D. Blair, *Multitarget-Multisensor Tracking: Applications and Advances*. Artech House, 2000.

[12] A. Wald, *Sequential Analysis*. Wiley series in probability and mathematical statistics. Probability and mathematical statistics, J. Wiley & Sons, Incorporated, 1947.

[13] R. Mahler, "PHD filters of higher order in target number," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 43, no. 4, pp. 1523–1543, 2007.

[14] K. Granström, C. Lundquist, and U. Orguner, "A gaussian mixture PHD filter for extended target tracking," *Linköping University Electronic Press*, 2010.

[15] B.-T. Vo, B.-N. Vo, and A. Cantoni, "Analytic implementations of the cardinalized probability hypothesis density filter," *Signal Processing, IEEE Transactions on*, vol. 55, pp. 3553–3567, July 2007.

[16] D. Macagnano and G. T. F. de Abreu, "Gating for multitarget tracking with the gaussian mixture phd and cphd filters," in *Positioning Navigation and Communication (WPNC), 2011 8th Workshop on*, pp. 149–154, April 2011.
[17] R. L. Streit and T. E. Luginbuhl, "Probabilistic multi-hypothesis tracking," tech. rep., DTIC Document, 1995.
[18] M.Wieneke and W.Koch, "A PMHT approach for extended objects and object groups," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 48, no. 3, 2012.
[19] U. Orguner, C. Lundquist, and K. Granström, "Extended target tracking with a cardinalized probability hypothesis density filter," in *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pp. 1–8, July 2011.
[20] L. Rokach and O. Maimon, "Clustering methods," in *Data mining and knowledge discovery handbook*, pp. 321–352, Springer, 2005.
[21] K. Granstrom, S. Reuter, D. Meissner, and A. Scheel, "A multiple model PHD approach to tracking of cars under an assumed rectangular shape," in *Information Fusion (FUSION), 2014 17th International Conference on*, pp. 1–8, IEEE, 2014.
[22] M. I. Shamos, "Computational geometry," 1978.