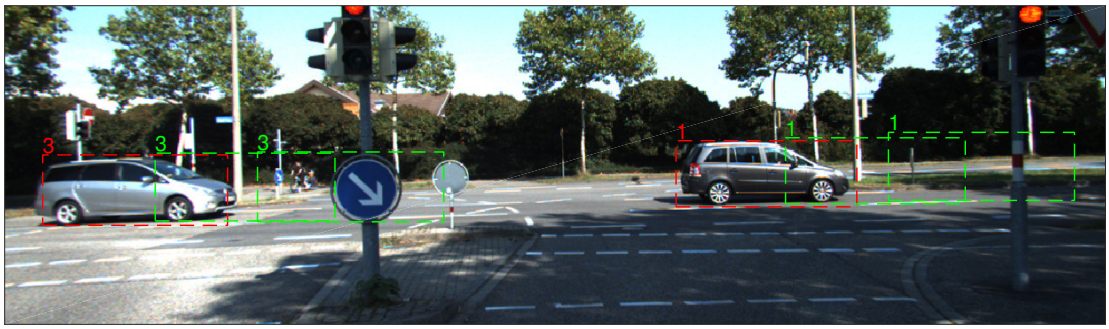


Multiple Object Tracking with Camera Data and Deep Learning Detectors

Master's thesis in Communication Engineering

Joachim Benjaminsson
Emil Rosenberg

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis EX075/2017

Multiple Object Tracking with Camera Data and Deep Learning Detectors

Joachim Benjaminsson
Emil Rosenberg

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2017

Multiple Object Tracking with Camera Data and Deep Learning Detectors
Joachim Benjaminsson
Emil Rosenberg

Supervisor: Samuel Scheidegger, Zenuity
Karl Granström, Department of Signals and Systems
Examiner: Karl Granström, Department of Electrical Engineering

Master's Thesis EX075/2017
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Estimated trajectories of tracked cars using the `PMBM` filter

Typeset in LaTeX
Gothenburg, Sweden 2017

Multi Object Tracking with Camera Data and Deep Learning Detectors
JOACHIM BENJAMINSSON, EMIL ROSENBERG
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Autonomous vehicles is a hot topic nowadays, with major car manufacturers trying to incorporate autonomous features into their vehicles. There is much more to an autonomous vehicle than only controlling the steering angle and throttle, perception of a vehicles surroundings is necessary for the vehicle to know how it should behave. One crucial aspect is to know where other vehicles and objects are positioned in relation to the ego–vehicle. Other objects can for example be detected via radar or camera. Although detecting objects in images is a simple task for humans, it is a complex task for a computer to perform. During recent years, fast advances in deep learning have been made, enabling development in certain sectors such as extracting data in images. Training Convolutional Neural Networks (CNNs) to detect objects in images has proven to yield good results. However, only relying on object detection can have drawbacks. Namely the network may not detect all objects due to different reasons, such as occlusion. Additionally, common deep learning detectors do not output information about the kinematics of the object. It is therefore of interest to investigate the combination of a deep learning detector and a recently developed multiple target tracking filter, namely the Poisson Multi–Bernoulli Mixture filter (PMBM), using camera data as input. The PMBM filter is based on the classical tracking approach, by using Bayesian statistics. An implementation of the PMBM filter has been made for tracking multiple objects in an urban setting with good tracking results. It has been shown that the filter compensates for the weaknesses of the CNN, providing a more robust view of the scene in front of the ego–vehicle. Some artifacts have also presented themselves, which need to be solved in order for the system to achieve an even better results. Furthermore suggestions for future improvements are presented, which could further increase the performance of the system.

Keywords: Multiple Object Tracking, PMBM, Deep Learning, Kalman Filtering, Bayesian Recursion, Similarity Score

Acknowledgements

We would like to thank our examiner Karl Granström and our supervisor Samuel Scheidegger for their guidance, support and feedback. It is greatly appreciated. Furthermore, a special thanks to Amrit Krishnan for showing a great interest in our thesis, for helping us move forward, having technical discussions and providing interesting ideas. A thanks to Lennart Svensson, whom in the early stages of this thesis took part of the fundamental discussions. Finally, we would like to express our thanks towards Autoliv for giving us the opportunity to carry out our thesis work.

Joachim Benjaminsson, Emil Rosenberg, Gothenburg, August 2017

Contents

List of Figures **xiii**

List of Tables **xv**

1 Introduction **1**
1.1 Purpose . . . 2
1.2 Scope . . . 2
1.3 Sensor Setup . . . 3
1.4 Overview of the System . . . 4
1.5 Related Work . . . 5
1.6 Contributions . . . 7
1.7 Report Structure . . . 8

2 Object Detection **9**
2.1 Change of Coordinate System . . . 9
2.2 Camera . . . 10
2.2.1 Camera Coordinate System . . . 10
2.2.2 Forward Projection . . . 10
2.2.3 Backward Projection . . . 11
2.3 Image Comparison . . . 11
2.4 Artificiell Neural Networks . . . 12
2.4.1 Activation Functions . . . 13
2.4.2 Convolutional Neural Networks . . . 13
2.4.2.1 Convolution . . . 14
2.4.2.2 Pooling . . . 15
2.4.3 Using CNNs for Object Detection . . . 15

3 Multiple Object Tracking **17**
3.1 Probability Theory . . . 17
3.2 Bayesian Estimation . . . 19
3.2.1 Time Update . . . 19
3.2.2 Measurement Update . . . 20
3.2.3 Kalman Filter . . . 20
3.2.3.1 Time Update . . . 21
3.2.3.2 Measurement Update . . . 21
3.2.4 Unscented Kalman Filter . . . 22
3.2.4.1 Time Update . . . 22

3.2.4.2 Measurement Update . . . 23
3.3 Motion Models . . . 24
3.3.1 Motion Model Discretization . . . 24
3.3.2 Constant Velocity . . . 25
3.3.3 Constant Acceleration . . . 26
3.4 Random Finite Sets in Multiple Object Tracking . . . 27
3.5 Poisson Multi-Bernoulli Mixture Filter . . . 27
3.5.1 The Conjugate Prior . . . 28
3.5.2 Time Update . . . 30
3.5.3 Measurement Update . . . 31
3.5.3.1 Poisson Prior Update . . . 31
3.5.3.2 Updating One Bernoulli Component . . . 32
3.5.3.3 Conjugate Prior Update . . . 33
3.5.4 Estimator . . . 34
3.6 Modeling Object Births . . . 34
3.6.1 Modelling the Poisson Component using a Gaussian Mixture . 35
3.6.2 Modelling the Poisson Component using a Uniform Distribution 35
3.7 Complexity Reduction . . . 35
3.7.1 Gating . . . 36
3.7.2 Optimal Assignment Algorithm . . . 36
3.8 Evaluation . . . 37
3.8.1 Generalized Optimal Subpattern Assignment . . . 37
3.8.2 CLEAR–MOT . . . 38

4 Implementation **41**
4.1 Complexity Reduction . . . 41
4.2 Similarity Score . . . 42
4.3 Time Update . . . 43
4.4 Measurement Update . . . 44
4.5 Estimator . . . 45
4.6 Numerical Instability . . . 46
4.7 Overall Algorithm . . . 46
4.8 Tracking in the image plane . . . 48
4.8.1 Motion Models . . . 48
4.8.1.1 Constant Velocity . . . 49
4.8.1.2 Constant Acceleration . . . 49
4.8.2 Measurement Models . . . 50
4.9 Tracking in 3D World Coordinate . . . 51
4.9.1 Coordinate System . . . 51
4.9.2 Motion Model . . . 52
4.9.3 Measurement Models . . . 52
4.9.4 Generating Births . . . 53
4.9.5 `GOSPA` evaluation . . . 55

5 Results **57**
5.1 Tracking in the image plane . . . 57
5.2 Tracking in 3D World Coordinate . . . 60

5.2.1 Removing False Positives . . . 60
5.2.2 Keeping Estimates when Measurements are Missing . . . 62
5.2.3 GOSPA Comparison with CNN detections . . . 64
5.2.4 Similarity Score . . . 67
5.2.5 CLEAR–MOT . . . 68

6 Discussion **69**
6.1 2D Image Tracking . . . 69
6.2 3D World Coordinate Tracking . . . 70
6.2.1 Suppressing False Positives . . . 70
6.2.2 Keeping Estimates when Measurements are Missing . . . 71
6.2.3 GOSPA Comparison with CNN detections . . . 71
6.2.4 Similarity Score . . . 74
6.2.5 CLEAR–MOT . . . 74
6.2.6 Delay in Newly Detected Targets and Dying Trajectories . . . 75
6.3 Future Work . . . 75
6.3.1 Objects Moving Out of Field Of View . . . 76
6.3.2 Bounding Box Representation . . . 76
6.3.3 Confirming Targets . . . 76
6.3.4 Similarity Score . . . 77
6.3.5 Probability Map of Missed Detection . . . 77
6.3.6 Motion Models . . . 78
6.3.7 Sensor Fusion . . . 79

7 Conclusion **81**

8 Bibliography **83**

List of Figures

1.1 Overview of the car. The coordinate systems that are shown in the figure are the local coordinate systems for each sensor [8]. 4
1.2 Top view of the car. The coordinate systems that are shown in the figure are the local coordinate systems for each sensor and their placement [8]. 4
1.3 Overview of the system. Images are fed as input to the CNN which performs object detection. For each object that is detected, a bounding box containing pixel position and size of the bounding box is fed as an input to the PMBM filter. The PMBM filter tracks the input and the outputs state estimates. 5

2.1 Transformation of a point (x, y) in a global coordinate system to a point (x', y') in a rotated and translated coordinate system. 10
2.2 Overview of the internals of the CNN. The earlier layers in the network is referred to as feature extraction, and the later layers are refered to as a classifier. The output from the feature extractor is sent to the classifier, and the regressors in parallel [30]. 14
2.3 Visualization of max–pooling with kernel window is 2×2, and step size 2. The red square highlights the maximum value from the pooling operand. 15

3.1 Illustration of Bayesian filtering and dependencies on time and measurement updates. 19
3.2 Illustration of recursively performing the time and measurement update steps. 20
3.3 Assume that during the first time step we have made the association that Target 1 is assigned to measurement 1. During the next time step, the target can either be missed (M.D), or associated to the one measurement obtained at the second time step. Additionally, a second target is created that is either non-existent (N.E), or associated to the measurement obtained during the second time step. There are two possible global hypotheses; 1) The measurement is connected to Target 1 and Target 2 is non existence. 2) Target 1 is missed and Target 2 is associated to measurement 2. 28

4.1 Illustration of the camera coordinate system, including distances and angles to a measurement. 54

5.1 GOSPA Score for tracking in the image plane. The filter performs much worse than the CNN on multiple occasions. 58
5.2 Figure that for four consecutive time steps highlights the problem with tracking with image state space. The *Car* changes ID due to the filter cannot predict the future state accurately, thus creating new objects. *Note: Teal box is the measurement, Red box is the estimate from the filter.* 59
5.3 Figure that for three consecutive time steps highlights the problem with false positives. The filter manages to suppress false positives. *Note: Teal box is the measurement, Red box is the estimate from the filter.* 61
5.4 Figure that for five consecutive time steps highlights the problem with missed detections and how the filter handles it. *Car* is the leftmost car with ID 234. *Note: Teal box is the measurement, Red box is the estimate from the filter.* 63
5.5 Image plane GOSPA evaluation of each sequence for both the CNN detections and the PMBM filter, averaged over all frames in each sequence. 64
5.6 Number of false positives for each sequence for both the CNN detections and the PMBM filter. 65
5.7 Number of false negatives for each sequence for both the CNN detections and the PMBM filter. 65
5.8 3D GOSPA evaluation of each sequence for both the CNN detections and the PMBM filter, averaged over all frames in each sequence. 66
5.9 Average GOSPA in image plane, per sequence. It is noticeable that the similarity score ever so slightly improve the GOSPA score. Albeit, it is quite even between using the similarity score or not. However, compared to the CNN, it is often a higher performance in favour of the filter, for both setups. 67

6.1 Figure illustrating the change of definition for the center of the bounding boxes of the CNN detections. The car of interest denoted Car in the sub–figure captions is the red, rightmost car. CNN detections of other objects are not displayed. *Note: Teal boxes are the CNN detections.* 72
6.2 Probability map of missed detection probability. There are two objects in the camera FOV, which results in occluded areas seen from the camera. Thus, the probability of detecting objects outside the FOV or inside the occluded areas is low (blue). Similarly, the probability of detection inside the FOV but outside the occluded areas is high (red). 78
6.3 The covariance of the distance estimation from the CNN as a function of distance. 79

List of Tables

1.1 Available sensors. 3

3.1 Example of the Hungarian method. Each task shall be assigned to one person and each person can only be assigned to one task, such that the total cost is minimized. The optimal assignment is marked with bold numbers, with total cost of 3. 37

5.1 Average GOSPA score over all sequences. 58
5.2 Total number of false positives and false negatives, recall rate and precision through all the KITTI tracking sequences. 66
5.3 Average GOSPA score over all sequences, both in image plane and in 3D coordinates. 66
5.4 Comparison of GOSPA score averaged for all sequences between the PMBM–filter and CNN in the data set from KITTI tracking data set with and without similarity score. 67
5.5 CLEAR-MOT score over all sequences for the filter. For the class *Car* the MOTA is almost 72 % and MOTP is almost 76 %. The ID switches are 59 and 70, respectively. The results shows that the filter is better at tracking *Cars* compared to tracking *Pedestrians*. 68

1

Introduction

In the last couple of years interest in autonomous cars has grown rapidly both in industry and research [1]. Reliable and safe urban autonomous driving requires several technologies, one of them is high performance environment perception. It is crucial to have information about the surroundings, where other vehicles are and where they are going, in order to avoid collisions. In recent years, image analysis and object detection have been revolutionized by the advances in deep learning, starting with `Alexnet` [2]. `Alexnet` is a deep Convolutional Neural Network (`CNN`) for classifying images and has millions of parameters and it was the first deep neural network to achieve a top 5 error rate of 15.4% (the next best network achieved an error rate of 26.2%) [2]. Deep learning is computationally expensive, but as computational power increases, large calculations in deep learning becomes more and more feasible. With the advances in deep learning, images could be used in conjunction with `CNN`s for environment perception.

One approach to increase the perception of the surrounding environment is to try to detect objects around the vehicle using for example a `CNN`, with images obtained from a camera. This approach works fairly well, however it comes with some problems. Objects might be undetected by the detection algorithm, or due to occlusion. Also, the position estimation of the objects might not be as accurate as required and could be erratic due to the frame by frame approach. Furthermore, the most common `CNN` for object detection does not generate information about the kinematic states, such as velocity, of the object. Such information is crucial in autonomous driving. The vehicle does not only requires knowledge of the current surroundings, but also requires knowing how the scene is changing and how it will look in the future to make decisions. In order to increase performance and to obtain more information about the objects, one could keep track of the objects also between images from the camera. By doing so, we introduce the problem of Multiple Object Tracking (`MOT`), where the task is to jointly estimate the number of objects in the area of interest and their states. A `MOT` filter would enable predicting the future states of objects, and thus play a crucial part in autonomous driving. Up until a few years ago, the most common way of solving the `MOT` problem has been to use a standard Bayes filter together with additional layers of logic [3]. However, these methods are not optimal and sometimes requires solving data association separately, which could be computationally expensive.

MOT problems in a Bayesian framework are often modelled by the Random Finite Set (RFS) framework. A RFS is defined as a set of random variables, also the cardinality of the set is a random variable. In MOT, objects of interest are modelled as random variables. We can also model the number of objects and measurements as random variables, as they are time–varying and unknown. Hence, RFS is a great tool for solving MOT problems. In MOT, both objects and measurements are modelled as RFSs [4].

One type of MOT filter that builds on the philosophy of RFS, is the Multi-Bernoulli filters. A challenge with MOT are missed objects, due to sensor noise, imperfect detections or occlusion. The Bernoulli part of the filter takes uncertain target existence into account when they are designed which models the objects that has been detected [5], and the Poisson models the objects that have not been detected [6].

1.1 Purpose

The purpose of this thesis is an in–depth investigation into how well MOT problems are solved when deep learning detectors are used to process camera data. Specifically, we investigate the possibilities of how to make the tracking more stable in regards to false targets and missed targets, but also how to exploit information within the image, such as the color of an object, and if it will produce an increase in performance. Hence, the main purpose of this thesis is to evaluate the performance of a MOT filter called Poisson Multi–Bernoulli Mixture filter (PMBM) with object detections made by a CNN in the image plane.

1.2 Scope

The focus will be on tracking *Cars*, *Cyclists* and *Pedestrians* using the publicly available KITTI object tracking data set [7]. The CNN is trained on KITTI data. Specifically, the CNN is trained on data obtained from the camera used by KITTI. Thus it is not recommended to change data set because there can be a decrease in performance if another camera is used, which is why the data set is limited to the KITTI object tracking data set.

Elapsed time is not included in the evaluation, hence comparison between other methods only focuses on the performance in tracking, irregardless of run time. Note that some methods are used to decrease computational complexity in order to make the system run in a feasible time frame. The training set consists of 20 image sequences in an urban setting. Each sequence contain a different number of images.

Generating the detections from the CNN is not a part of this thesis. The detections are made beforehand and therefore they cannot be affected. The focus lies solely on designing and implementing a PMBM filter for tracking both in the image plane

and 3D world coordinates. Furthermore, since tracking in 3D requires a distance estimate, it is not possible to change the network that performs the detections as this requires changing the network structure and especially the loss function.

1.3 Sensor Setup

The dataset that has been used is the publicly available dataset `KITTI` [7]. The vehicle in use for obtaining measurements is a Volkswagen Passat B6. Mounted on the vehicle there are multiple sensors which are presented in Table 1.1. Their placement and corresponding coordinate systems are defined according to Figure 1.1 and Figure 1.2 respectively. Many sensors focus on the surroundings of the ego–vehicle, however there is also an Inertial Measurement System (`IMU`) that provides information about the position and motion of the ego–vehicle.

Table 1.1: Available sensors.

Type of Sensor	Name	Update Frequency	Other
1 Inertial Measurement Unit (IMU)	OXTS RT 3003	100 Hz	Res: 0.02m/1°
1 Laser scanner	Velodyne HDL–64E	10 Hz	–
2 Grayscale Cameras	Point Grey Flea 2 (FL2–14S3M–C)	–	Res: 1.4 Megapixels
2 Color cameras	Point Grey Flea 2 (FL2–14S3C–C)	–	Res: 1.4 Megapixels
4 Varifocal lenses	Edmund Optics NT59–917	–	4 mm, 90°[1] and 35°[2]

[1]Horizontal angle
[2]Vertical angle

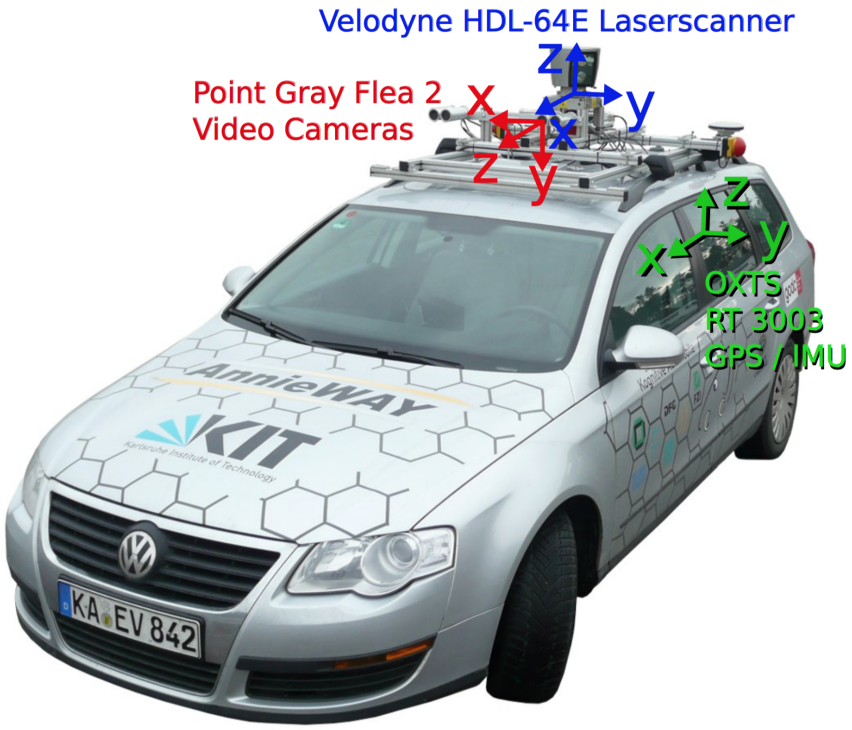


Figure 1.1: Overview of the car. The coordinate systems that are shown in the figure are the local coordinate systems for each sensor [8].

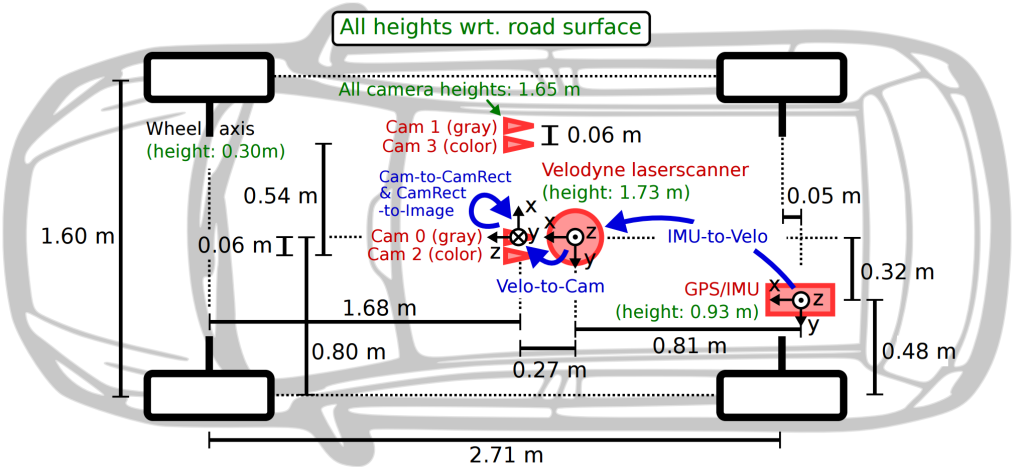


Figure 1.2: Top view of the car. The coordinate systems that are shown in the figure are the local coordinate systems for each sensor and their placement [8].

1.4 Overview of the System

The proposed system contains two entities. The first entity is the CNN, which takes an image as input and outputs a bounding box for each detection. A bounding box

contains pixel position and size of the detection object. The second entity is the PMBM filter. For each image, it takes bounding boxes as input and outputs state estimates. An overview of the system is shown in Figure 1.3.

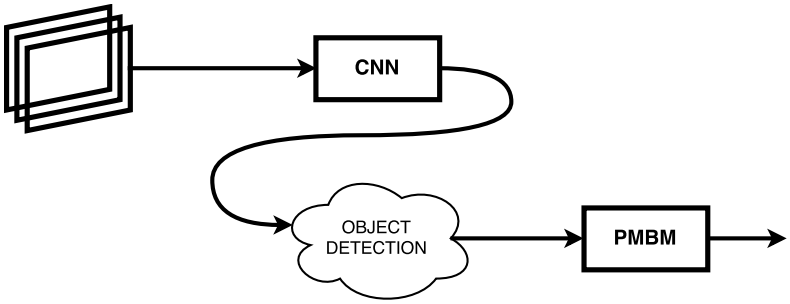


Figure 1.3: Overview of the system. Images are fed as input to the CNN which performs object detection. For each object that is detected, a bounding box containing pixel position and size of the bounding box is fed as an input to the PMBM filter. The PMBM filter tracks the input and the outputs state estimates.

1.5 Related Work

State–of–the–art single target trackers focus on strong appearance models for performing tracking [9]. However, it is non–trivial how a single target tracker can be extended to a multi–object–tracker, as the single target trackers assume that the object of interest will always be detected. Hence, a single target tracker does not take into account that objects can enter and leave the Field of View (FOV) [9].

The two most common approaches to solve the MOT problems are Multiple Hypothesis Tracking (MHT) and Joint Probabilistic Data Association (JPDA) [4]. MHT builds on generating hypotheses with respect to if a new measurement belongs to a previously known target or not. It considers all hypotheses for associating measurement to targets, thus making the algorithm computationally complex [10]. The posterior is obtained by creating a mixture of conditional probabilities, where each conditional probability is obtained from weighting how likely each hypothesis is to be the correct data association [10]. However, the algorithm does not consider nonlinear measurements, nonlinear dynamics or maneuvering targets [11]. Even though it has been some years since MHT was developed, it is still competitive in visual tracking, albeit computationally complex [12].

JPDA is another well–known algorithm that is used for MOT. This filter computes the measurement–to–target association probabilities jointly across the set of targets. However, JPDA makes some assumptions. The number of targets are known, where each target has a motion and measurement model that is affected by zero-mean, mutually independent white Gaussian noise [13]. Because of the assumption that the number of targets must be known, JPDA does not handle birth and deaths of

objects explicitly, thus another component must take care of this if JPDA is used. In contrast to MHT, JPDA makes decision based on all measurements in each time step using a weighted sum given by the marginalization of the measurement–to–target association probabilities, while MHT propagate the hypotheses to get better estimates [12].

Both MHT and JPDA are extensions of their single target tracking algorithm, and though they do work for MOT they do not solve one important characteristic, the number of objects is unknown and time–varying. This is where RFS models are useful. In a standard Bayes filter the distribution is propagated using recursion to perform tracking. However, when there is a multi–object–set, a combinatorial problem occurs. This was solved by introducing the Probability Hypothesis Density (PHD) [14], and the following Cardinalized–PHD (CPHD). The recursion is similar to the single target case, however in RFS we work with sets instead of single states. In single target tracking, the posterior is computed with the first and second moments. Sufficient statistics ensures that the propagation of these two moments are complete. However, it is sufficient to only propagate the first order in filtering [14]. This holds in MOT too when applying the RFS philosophy, but here the first moment is the PHD.

A PHD to a set is similar to what the expected value is for a random variable. The PHD can be seen as a intensity function and is defined as follows: "PHD is a density whose integration over a space results in the number of objects present within that space" [14]. Since the PHD–filter only propagates the first moment it achieves low computational complexity. However, it has difficulties estimating the cardinality of the random set [14], especially with high precense of false targets and missed targets. Thus, CPHD filter was introduced, which similar to the PHD, propagates the first moment, but it also propagates the complete cardinality distribution [14].

Another tracking algorithm is presented in [15] using a Markov Chain Monte Carlo (MCMC) approach to the MOT problem. The method uses the illumination intensity of full images instead of detections in order to not lose any information in the pre–processing stage. The main idea is to use MCMC to sample from the variables of interest, see [15] for details. During experiments the tracker seems to perform well, even in the case of overlapping targets. The tracker is compared with the Multi–Bernoulli filter in [16], which it outperforms [15]. However, the algorithm is computationally heavy and it took more than six times longer than the Multi–Bernoulli filter to run. Furthermore, the authors suggests not to replace an online tracker with their MCMC–tracker, it is rather to be used to refine online estimates [15].

In image target tracking, deep learning methods has also been in the scope of research. In [17], the authors propose learning hierarchical features for tracking, which they integrate with a (at that time) state–of–the–art visual tracking system, namely the adaptive structural local sparse appearance model presented in [18]. The idea is to first pre–learn complicated motion transformations offline, and then adapt the pre-learned features according to the appearance of a specific object of interest in an online fashion [17]. Please refer to [17] and [18] for details. Experiments have

shown that the method is robust to both motion transformations and appearance changes of a target. However, it seems that this system only handles the single target tracking case for specific objects, so it might be insufficient to use in a MOT situation. As the tracker is tracking one specific object, the use of the method in MOT would probably require setting up a filter for each class. For each filter, first perform the offline pre–learning of the motion transformation for the class. After which, adapt the pre–learned features according to every detected specific object of the class. However this seems to be a tedious approach.

In [19] the authors also views the MOT problem as a learning task, using Markov Decision Processes (MDP) and hierarchical Recurrent Neural Networks for the appearances, motions and interactions of the objects to estimate their states. The idea is to compute similarity scores between targets and detections for each of the three properties and then combine the similarity scores in order to obtain a final weighting [19]. Another method that is suggested in [20] where again the MOT is seen as decision making in MDPs. They also use reinforced learning to learn a policy for the MDP, which in this case is equivalent to learning a similarity function for the data association.

Other tracking approaches that has been developed in visual tracking have used a PHD filter to deal with false detections, as it is robust to noisy detections, seeing some improvement in both detection and tracking. The authors of [21] states that their algorithm can become more competitive if state–of–the-art appearence features are used instead of color histograms. In [22], an online convolutional network is used for visual object tracking, where a regression model is used for predicting the position of a target and achieves good performance in location precision. In [16], they use visual detections and the tracking algorithm is based on the Multi–Bernoulli approach which achieves great results of non-overlapping objects and claim to achieve more effective results than other existing methods. However, they struggle with inefficient particle implementations. In [3], they detect single target objects using corner detection and track targets with a Bernoulli filter. In a video tracking experiment, their solution had fair accuracy but the trajectory was not smooth due to the high process noise. Based on these works, it is indicated that the combination of a CNN and a MOT, filter such as a Multi–Bernoulli filter, is an interesting idea and using a combination of both has a potential of performing robust tracking.

1.6 Contributions

The traditional tracking solutions include some sort of JPDA or MHT filter. The tracking algorithms that use computer vision for generating detections (for example feature detections) performs tracking using optical flow or other computer vision methods. The main approach in this thesis is to use a deep neural network for detections, combined with a classical, Bayesian approach for tracking objects. That is, using a PMBM filter that builds on the RFS philosophy for modelling objects and measurements. Additionally, an investigation if pixel information within the detec-

tions' bounding boxes can be exploited for tracking purposes is conducted. Finally, an investigation of which of the image plane or 3D world coordinate system is the most suitable domain for tracking.

1.7 Report Structure

The structure of the thesis is as following. First, Chapter 2 explains the theory about object detection and neural networks. Chapter 3 covers the necessary theory for the `MOT` problem, including details about the `PMBM` filter, the different steps are explained and a section containing a discussion about the evaluation techniques is included. Chapter 4 contains details about the different setups and implementation details. The results are presented in Chapter 5 and a discussion about the obtained results are in Chapter 6, as well as a discussion about future work. Finally, our conclusions are presented in Chapter 7.

2

Object Detection

The aim with the chapter is to introduce the reader to some basics in image processing and object detection, which are important aspects in this thesis. First, as it is important to be able to describe the position of objects in different coordinate systems, theory about how to move between different coordinate systems is presented. Similarly, it is necessary to transform a position in the image plane to a $3D$ coordinate and contrarily. Furthermore, one method of comparing similarities between images is presented and theory about neural networks and object detection is given.

2.1 Change of Coordinate System

In order to use the different sensors in the vehicle, the ability to change the state vectors between the different coordinate systems is needed. This is done by using a transformation, consisting of a rotation matrix $\mathbf{R}_a^b \in \mathbb{R}^{3x3}$ and a translation vector $\mathbf{t}_a^b \in \mathbb{R}^{3x1}$, which takes a state vector from coordinate system a to the coordinate system b by

$$\begin{aligned} \mathbf{T}_a^b &= \begin{bmatrix} \mathbf{R}_a^b & \mathbf{t}_a^b \end{bmatrix} \\ \mathbf{x}_b &= \mathbf{T}_a^b \begin{bmatrix} \mathbf{x}_a \\ 1 \end{bmatrix}. \end{aligned} \tag{2.1}$$

It is also of interest to be able to move a point in a global coordinate system into a corresponding point in a local coordinate system. The local coordinate system may be both rotated and translated in comparison with the global coordinate system, a 2D example can be seen in Figure 2.1. Proceeding from the compounding operations presented in [23] and the transformations given in [24], [25], the change of coordinate system from global coordinates (x, y) to local coordinates (x', y'), rotated by θ and translated by $[t_x, t_y]^T$, is given by

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x - t_x \\ y - t_y \end{bmatrix}.$$

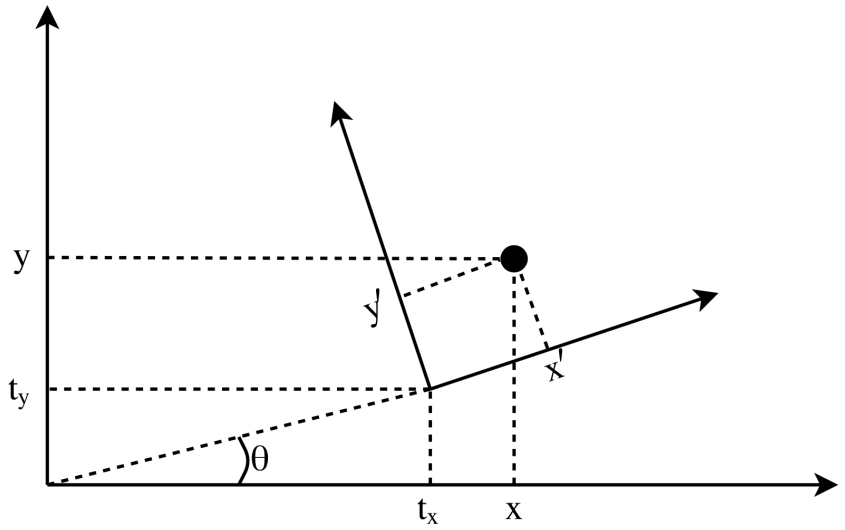


Figure 2.1: Transformation of a point (x, y) in a global coordinate system to a point (x', y') in a rotated and translated coordinate system.

2.2 Camera

The camera that was used was the left color camera (`Cam` 2 (color) in Figure 1.2, now denoted $\texttt{Cam}_2$), mounted on the roof of the vehicle.

2.2.1 Camera Coordinate System

The camera coordinate system is given in meters and the origin is defined as the center of the camera. The reader is referred to Figure 1.1 for an understanding of how the coordinate system is rotated in relation to the ego-vehicle and other coordinate systems.

2.2.2 Forward Projection

Forward projection is an operation that maps some coordinates $\left[X, Y, Z\right]^T$ onto a smaller space with coordinates $\left[x, y\right]^T$, scaled with some factor w using a matrix $\mathbf{P}$. $\mathbf{P}$ is composed by a 3×3 rotation matrix, $\mathbf{M}$, and a translation vector $\mathbf{p}$, $\mathbf{P} = [\mathbf{M}|\mathbf{p}]$.

The projection from an object in a 3D coordinate system onto the image plane is defined as

$$w\mathbf{x} = w\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{PX} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.2}$$

Here, the matrix $\mathbf{P}$ is the projection matrix where $P_{i,j}$ denotes the i^{th} row and j^{th} column, and $\mathbf{X}$ is the homogeneous coordinate (a 3D point) in the camera coordinate system. Usually, the projected point $\mathbf{x}$ is given in homogeneous coordinates, thus it has to be normalized with w. Note, the depth is lost due to this normalization.

2.2.3 Backward Projection

Backward projection is the operation that maps some coordinates $\mathbf{x}$ onto a a larger space with coordinates $\mathbf{X}$, that is, it is reversing the forward projection. Performing backward projection is a greater challenge than forward projection due to the unknown distance (or depth), which has to be estimated in order to recover the original coordinates. $\mathbf{x}$ has to be on the form

$$\mathbf{x} = \hat{w}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where $\hat{w}$ is the estimated depth of the object. The 3D point can be recovered if the depth is known by

$$\mathbf{X} = \mathbf{P}^{+}\mathbf{x} + \lambda\mathbf{C} \tag{2.3}$$

where $\mathbf{P}^{+}$ is the pseudo–inverse of the projection matrix $\mathbf{P}$, $\mathbf{x}$ the image point, λ a scaling factor and $\mathbf{C}$ the camera center. The pseudo–inverse is given by $\mathbf{P}^{+} = \mathbf{P}^{T}(\mathbf{P}\mathbf{P}^{T})^{-1}$ such that $\mathbf{P}\mathbf{P}^{+} = \mathbf{I}$. Furthermore, the camera center is given by the inverse of the rotation matrix $\mathbf{M}$ times the translation vector $\mathbf{p}$, $\mathbf{C} = -\mathbf{M}^{-1}\mathbf{p}$.

2.3 Image Comparison

One way to describe an image is to describe the distribution of the colors within the image using a histogram. The histogram is constructed by quantizing the pixel

values for each color. Thus, in the RGB–color space three histograms are needed to represent the image. Furthermore, since a color histogram only represent the distribution of the colors within the image and no spatial information, two images can have similar color distributions and therefore similar histograms, even though they may look different [26].

Different approaches exist for comparing similarities in images. One solution is to use the distribution of color inside an image. To check for similarities in two images one can compare two histograms and see if the color distribution match. One distance metric that is commonly used for comparing histograms in image classification is the Chi–square distance [27]. It is defined as

$$\chi^2(h_1, h_2) = \frac{1}{2}\sum_{i=1}^{N}\frac{(h_1(i) - h_2(i))^2}{h_1(i) + h_2(i)} \tag{2.4}$$

where h_1 and h_2 are the histograms of interest, and i is the i^{th}–bin. N is equal to the range of pixel values.

A cost function which is widely used in tracking [28] is the negative log function. It can be used for linking similarities by

$$w_{\chi^2} = -\ln(\chi^2(h_1, h_2)). \tag{2.5}$$

2.4 Artificiell Neural Networks

A network consisting of neurons is called a neural network, and exist in nature. Neurons have several inputs, and when the sum of the inputs is above some threshold, neurons sends a signal further into the network. In computer science, an artificial neural network performs similarly. By using an activation function, the sum of inputs will either trigger or not trigger the activation function, where each input is associated with an input weight. Neural networks can be used to predict future outputs, or trends, based on some input. For the network to be able to perform prediction, it has to be trained on a set of training data. Training a neural network is done by minimizing a cost function, which is achieved by changing the weights of the activation function.

However, the least cost for the training set might not yield the best performance on real data, due to overfitting. In deep networks, the most common reason for overfitting is the large number of parameters. Overfitting is the problem when the network can perform prediction on the training data very accurately (that is, the network fit the training set well), but generalizes poorly, and thus the performance is suboptimal on the validation set and test set. An efficient countermeasure against overfitting for deep networks is for example drop–out [29]. We refer the reader to [29]

for further details on how to avoid overfitting.

2.4.1 Activation Functions

An activation functions is a function that takes an input and computes the output. As the name implies, the function "activies" when the input is above some threshold. However, the output does not have to be binary, it could be continuous as well. Common activation functions are the Sigmoid

$$f(x) = \frac{1}{1 + e^{-x}}$$

and Hyperbolic tangent function [30]

$$f(x) = \tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

Another activation function that can be used to estimate a posterior probability of each class, is the soft–max function. Given the input vector $\mathbf{x}$, the output probability for class i is defined as

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}.$$

Propagating the outputs from the neurons at layer $l-1$ to the next layer l, where each layer consists of many neurons, a feedforward neural net is achieved. The output at layer 1 with the input vector $\mathbf{x}$, and the output $\mathbf{o}_l$ at layer l with input vector $\mathbf{o}_{l-1}$ are calculated as follows

$$\mathbf{o}_1 = \mathbf{f}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$
$$\mathbf{o}_l = \mathbf{f}(\mathbf{W}_l\mathbf{o}_{\mathbf{l-1}} + \mathbf{b}_l)$$

where $\mathbf{W}_l$ is a weight matrix describing how neurons are connected between the layers $l-1$ and l, f is the activation function and $\mathbf{b}_n$ is a bias vector [31].

2.4.2 Convolutional Neural Networks

With the advances in deep learning, object detection has become more robust. A CNN takes an image as an input and performs object detection. The output of

each detected object could be its class, but also a bounding box, which contains pixel positions and size. One technique for detection is called detection through classification. It has been shown that CNNs are a promising tool for classification, thus CNNs can be used for detection through classification [30].

In the image, the detector makes the decision if it is an object or background. Then, using a bounding box regressor and a range regressor in parallel with the classifier, a position of the object and a distance is obtained and which class it belongs to, respectively. CNNs are specifically designed to handle spatial grids as inputs making it promising for object detection and classification in images. A CNN's architecture consists of many layers, making it a deep network.

Each layer has a dimension related to the number of input channels. Typically, the first layer has an input dimension related to the number of color channels, while layers at greater depths have a greater number of channels, referred to as *feature maps*, but a lower spatial dimension. The end layer has a decreased spatial dimension that matches the desired output dimension. First, in the earlier layers (referred to as a feature extractors) feature extraction is performed by the CNN. The later layers are in this context referred to as a classifier [30]. The output from the feature extractor is, in addition to the classifier, propagated to the two regressors in parallel. This is visualized in Figure 2.2. The classifier applies a soft–max function at the end layer and a probability distribution of the classes are given as the output. The first regressor outputs the position of each detected object, and the second outputs the range to each detected object [30]. The main idea is to train the network to detect certain features such as appearance in images, details of the networks used in this thesis can be found in [32] and [30].

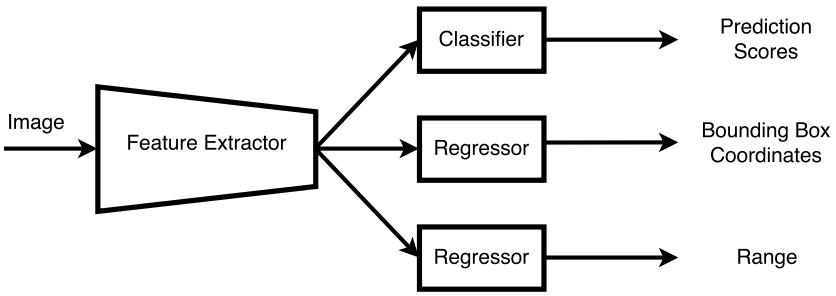


Figure 2.2: Overview of the internals of the CNN. The earlier layers in the network is referred to as feature extraction, and the later layers are refered to as a classifier. The output from the feature extractor is sent to the classifier, and the regressors in parallel [30].

2.4.2.1 Convolution

Using a convolution kernel $\mathbf{k}$ with size $W \times H$, and the input image $\mathbf{X}$ has size $M \times N$, the result $\mathbf{Y}$ of the convolution will have the size $(M - W + 1) \times (N - H + 1)$, where

the value at pixel position i, j will be

$$Y_{i,j} = \sum_{i'=1}^{W} \sum_{j'=1}^{H} k_{i'j'} X_{i+i',j+j'}$$

and the convolution operand is denoted

$$\mathbf{Y} = \mathbf{k} * \mathbf{X}.$$

One conseqeuence of convolution is a decrease in the spatial dimension between layers. Let $\mathbf{X}_j$ be the feature map in some layer with N total feature maps, then the corresponding feature map in the next layer $\mathbf{Y}_i$ is calculated by using the element–wise activation function f_i, the feature map $\mathbf{X}_j$ and a bias $\mathbf{b}_i$ [30],

$$\mathbf{Y}_i = f_i(\sum_{j=1}^{N} k_{ij} * \mathbf{X}_j + \mathbf{b}_j).$$

2.4.2.2 Pooling

One way to decrease the spatial dimensions of the feature maps is to perform pooling. There are two popular pooling functions, the max–pooling and the average–pooling. Since they produce similar results, only max–pooling will be covered.

By sliding a kernel window of size $k \times k$ over the feature map, and at each slide step, applying the max–function, only the max value inside the sliding kernel window is kept. The rest of the values inside the window are not stored [30]. The max–pooling function is visualized in Figure 2.3.

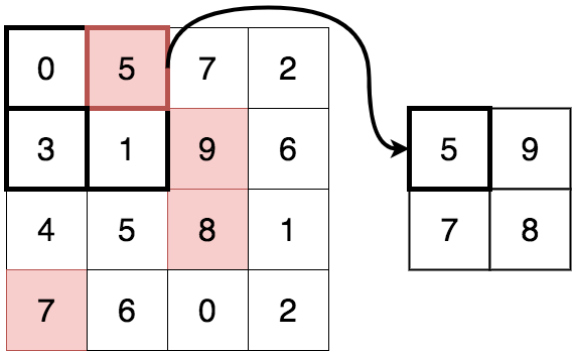


Figure 2.3: Visualization of max–pooling with kernel window is 2×2, and step size 2. The red square highlights the maximum value from the pooling operand.

2.4.3 Using CNNs for Object Detection

In CNNs, object detection can be performed by a technique called "detection–through–classification". This is commonly done by first obtaining smaller patches of the input

image, so called proposals, in which the network finds pre–trained features. In the last layer, the number of feature maps present are set to be the same as the number of classes where each feature map is associated to one class. By using the soft–max function the output can be interpreted as a probability distribution of the classes [30].

3

Multiple Object Tracking

The task of MOT is a difficult problem due to the need of jointly estimating the number of targets and their states. Since objects may appear and disappear randomly, by entering and exiting the FOV, the solution must be able to handle a random number of objects. Due to a random number of objects, the receiving number of measurement will also be random. In addition, there could be missed objects, or a measurement could be considered as clutter. Hence, both the set of object and set of measurements will be random, and the cardinality of each set will also be random. Therefore, the measurement–to–target assignment is nontrivial. The purpose of this chapter is to explain some of the most important theory behind the filter used to perform the MOT in this thesis. The reader is referred to [5] for full derivation and support during this section.

3.1 Probability Theory

In the classical view of probability, also known as frequentist statistics, the quantity of interest is modeled as unknown and deterministic. It is also interpreted by which frequency an event occurs. However, in case of an unrepeatable event, this definition is not plausible. Another approach is to model the uncertainties of the quantity as random, thus the quantity is described as random. This methodology is known as Bayesian statistics, and is an important topic in this thesis.

A probability density function (PDF) is a function describing how a continuous random variable is distributed. Integrating the PDF over an interval yields a probability of the variable being in that interval. Similarly, a discrete random variable has a probability mass function (PMF), at each point the function yields a probability of the variable taking that value. The PDF and PMF should integrate and sum to one, respectively. In the case when approximating PDF's and PMF's, it may be necessary to scale the functions to validate that this property is fulfilled, this is referred to as normalization. Some important distributions in this thesis are Bernoulli, Gaussian, Poisson, and uniform.

Bernoulli distribution: A random variable following a Bernoulli distribution takes the value 1 with probability p and the value 0 with probability $1 - p$, thus the PMF

is given by

$$f(k) = \begin{cases} p & \text{for } k = 1 \\ 1 - p & \text{for } k = 0. \end{cases} \tag{3.1}$$

Univariate Gaussian distribution: Also known as the normal distribution. The PDF is defined by two variables, namely the mean μ and the variance σ^2. The PDF is

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \text{ for } k = 0, 1, 2, \ldots.$$

Poisson distribution: The Poisson distribution is a discrete distribution describing the rate of events occurring. The function is determined by the average rate, and the occurrence of an event is independent of the time since the last event happened. The PMF of a Poisson variable is

$$f(k) = \frac{\lambda e^{-\lambda}}{k!},$$

where k is the number of events and λ is the average rate of events occurring.

Uniform distribution: A uniform distribution can be either continuous or discrete. In this thesis, the continuous uniform distribution is of interest. As the name suggests, a variable with a uniform distribution takes any value within the limits with equal probability. The PDF is

$$f(x) = \begin{cases} \frac{1}{b-a} & x \in [a, b] \\ 0 & \text{otherwise.} \end{cases}$$

The following concepts regarding probability theory are important:

Prior distribution: The prior knowledge about the distribution of the random variable of interest x, denoted $p(x)$.

Likelihood representation: The likelihood function, often denoted $l(x|z) = p(z|x)$ is a function of the data z, given a random variable x. Note that the likelihood is not a density with regards on the random variable x.

Posterior distribution: The posterior distribution is the distribution of the random variable x given a measurement z, thus is denoted $p(x|z)$.

Bayes' theorem: The relation between the mentioned distributions can be expressed by Bayes' theorem, which is a central equation in Bayesian statistics,

$$p(x|z) = \frac{p(z|x)p(x)}{p(z)}. \tag{3.2}$$

3.2 Bayesian Estimation

Bayesian estimation refers to the task of recursively estimating the state $\mathbf{x}_k$, at time k, from observations $\mathbf{z}^k$, where $\mathbf{z}^k$ are the measurements obtained up to and including time k. The dependencies are illustrated in Figure 3.1.

The algorithm of estimating the state is divided into two steps, the time update step (also called prediction) and the measurement update step. Commonly used algorithms/filters to perform the Bayesian estimation are different variations of the Kalman Filter (KF), such as Extended KF and Unscented KF. One can also use Monte Carlo samples, so called Particle filters. The choice of filter is dependent on the type of distributions and the nonlinearities and uncertainties of the motion and measurement models.

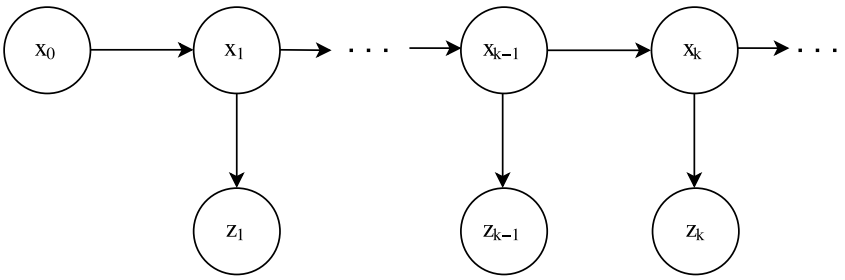


Figure 3.1: Illustration of Bayesian filtering and dependencies on time and measurement updates.

3.2.1 Time Update

In the time update step, the idea is to predict the state $\mathbf{x}_k$ given measurements up to time $k-1$, $\mathbf{z}^{k-1}$, this is commonly done using the Chapman–Kolmogorov equation,

$$p(\mathbf{x}_k|\mathbf{z}^{k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}^{k-1})d\mathbf{x}_{k-1}. \tag{3.3}$$

The transition density $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is defined from the choice of motion model $\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{v_{k-1}})$, where $\mathbf{v_k}$ is a random noise process included in order to handle uncertainties and model errors [33]. Namely the time update predicts the motion of the object.

3.2.2 Measurement Update

The predicted state is updated with the information from the measurement at time k. The connection between the state and the measurement is given by a measurement model $\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{w}_k)$, where $\mathbf{w}_k$ is noise. The measurement model gives rise to the likelihood of the measurement, namely $p(\mathbf{z}_k|\mathbf{x}_k)$. Since the state is estimated, it is common that the state is described by its distribution. Thus, it also includes information about the uncertainty of the estimation. We denote the prediction distribution $p_{k|k-1}(\mathbf{x}_k|\mathbf{z}^{k-1})$ and the posterior distribution $p_{k|k}(\mathbf{x}_k|\mathbf{z}^k)$. Subscript $k|k-1$ means that the variable was computed for time k given measurements up to time $k-1$. Similarly, $k|k$ where measurements up to time k was used. From Bayes' theorem (3.2) it follows that

$$\begin{aligned} p_{k|k}(\mathbf{x}_k|\mathbf{z}^k) &\propto p(\mathbf{z}^k|\mathbf{x}_k)p(\mathbf{x}_k) \\ &\propto p(\mathbf{z}_k|\mathbf{x}_k)p_{k|k-1}(\mathbf{x}_k|\mathbf{z}^{k-1}). \end{aligned} \tag{3.4}$$

3.2.3 Kalman Filter

The Kalman filter [34, p. 56] is a way of recursively finding the Bayesian estimate $\hat{\mathbf{x}}_k$ of the true state $\mathbf{x}$, see Figure 3.2. Thus it is usually more accurate than filters that compute their estimates using only the current measurement. First, at time k, a prediction is made according to the Chapman–Kolmogorov equation (3.3). When a new measurement is received, an update of said prediction is calculated based on Bayes' theorem (3.4). How much the update relies on the prediction and the new measurement is determined by the Kalman gain, which is a way to weight the two update steps against each other, depending on their respective uncertainty. It can be shown that the (linear) Kalman filter is optimal (in the sense of minimizing the mean square error) in the cases where the noise is Gaussian [35].

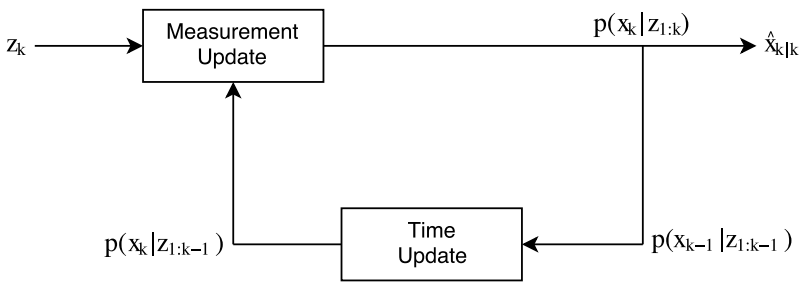


Figure 3.2: Illustration of recursively performing the time and measurement update steps.

3.2.3.1 Time Update

As mentioned, during the time update step a prediction of the state is performed using a motion model $\mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{v_{k-1}})$. In the case of linear motion models and independent Gaussian noise, the motion model of the state can be written

$$\mathbf{x_k} = \mathbf{F_{k-1}}\mathbf{x_{k-1|k-1}} + \mathbf{q_{k-1}}.$$

The time update of the mean and covariance can be computed as

$$\begin{aligned}\hat{\mathbf{x}}_{k|k-1} &= \mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1|k-1} \\ \mathbf{P}_{k|k-1} &= \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1},\end{aligned}$$

where $\mathbf{Q_{k-1}}$ is the process noise covariance at time $k-1$.

3.2.3.2 Measurement Update

Given a new measurement $\mathbf{z_k}$ at time k with measurement covariance $\mathbf{R}_k$, we can update the predicted state. In the case of linear measurement models and independent Gaussian noise, we can describe the measurement model $\mathbf{z}_k = \mathbf{h}(\hat{\mathbf{x}}_k, \mathbf{w}_k)$ as

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{w}_k.$$

The update equations of the Kalman filter are

$$\begin{aligned}\hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\mathbf{v}_k \\ \mathbf{P}_{k|k} &= \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T\end{aligned}$$

where the Kalman gain $\mathbf{K}_k$, innovation $\mathbf{v}_k$, the innovation covariance, $\mathbf{S}_k$ at time k are

$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_{k|k-1}\mathbf{H}_k^T\mathbf{S}_k^{-1} \\ \mathbf{v}_k &= \mathbf{z}_k - \mathbf{H}_k\hat{x}_{k|k-1} \\ \mathbf{S}_k &= \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k.\end{aligned}$$

The innovation captures the new information that the new measurement brings and the Kalman gain determines how much we should rely on this information.

3.2.4 Unscented Kalman Filter

The Kalman filter is derived from linear models, thus is only optimal for this case and not for nonlinear models. If the models are nonlinear, linearization can be used, known as the Extended Kalman Filter (`EKF`). The `EKF` works well in many cases, although it may perform poorly if the model is significantly nonlinear within the uncertainties of the models. However, there are other methods of solving the estimation task that are more robust to nonlinearities. The derivation of the Kalman filter contains multiple integrals of the type

$$\int \mathbf{g}(\mathbf{x})\mathcal{N}(\mathbf{x}; \hat{\mathbf{x}}, \mathbf{P})d\mathbf{x} = \mathbb{E}[\mathbf{g}(\mathbf{x})], \tag{3.5}$$

where $\mathbb{E}[.]$ denotes the expected value. Furthermore $\mathbf{g}(\mathbf{x})$ is either a motion or measurement model, which can be nonlinear. Integrals as in (3.5) occurs for example in the Chapman-Kolmogorov equation (3.3) and may look like

$$\int \mathbf{g}(\mathbf{x}_{k-1})\mathcal{N}(\mathbf{x}_{k-1}; \hat{\mathbf{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1})d\mathbf{x}_{k-1}.$$

These integrals can be solved for linear models, however with nonlinear models this is not the case. Thus, these integrals needs to be approximated, one way of doing so is to use the Monte Carlo method. The idea is to generate independent and identically distributed samples $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, ..., $\mathbf{x}^{(N)}$ from its distribution $p(\mathbf{x})$, then we can approximate

$$\int \mathbf{g}(\mathbf{x})p(\mathbf{x})d\mathbf{x} \approx \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}(\mathbf{x}^{(i)}).$$

However as the Monte Carlo method relies on picking random samples, it might require a lot of samples in order to represent the true distribution. As an alternative, there are so called σ–point methods which use deterministic samples chosen in clever ways to cover a large area of the space even though the number of samples is small. We will focus on one of these, the so called Unscented Kalman Filter (`UKF`) [34, p. 86]. Two advantages of this filter are that it is efficient since it uses quite few samples, and it also only has one tuning parameter.

3.2.4.1 Time Update

With a state vector $\mathbf{x}_k$ of dimension n, the idea is to generate $2n + 1$ σ–points $\boldsymbol{\chi}$,

$$\begin{aligned}
\boldsymbol{\chi}_{k-1}^{(0)} &= \hat{\mathbf{x}}_{k-1|k-1} \\
\boldsymbol{\chi}_{k-1}^{(i)} &= \hat{\mathbf{x}}_{k-1|k-1} + \sqrt{\frac{n}{1-W_0}}\mathbf{P}_{i,k-1|k-1}^{(1/2)}, \quad i = 1, 2, ..., n \\
\boldsymbol{\chi}_{k-1}^{(i+n)} &= \hat{\mathbf{x}}_{k-1|k-1} - \sqrt{\frac{n}{1-W_0}}\mathbf{P}_{i,k-1|k-1}^{(1/2)}, \quad i = 1, 2, ..., n,
\end{aligned}$$

where W_0 is the tuning parameter (namely the weight of $\boldsymbol{\chi}^{(0)}$). $\mathbf{P}^{(1/2)}$ is a matrix such that

$$\mathbf{P} = \mathbf{P}^{(1/2)}(\mathbf{P}^{(1/2)})^T$$

and $\mathbf{P}_i^{(1/2)}$ is its ith column. The `UKF` prediction equations are then given by

$$\begin{aligned}
\hat{\mathbf{x}}_{k|k-1} &= \sum_{i=0}^{2n} \mathbf{f}(\boldsymbol{\chi}_{k-1}^{(i)})W_i \\
\mathbf{P}_{k|k-1} &= \mathbf{Q}_{k-1} + \sum_{i=0}^{2n} (\mathbf{f}(\boldsymbol{\chi}_{k-1}^{(i)}) - \hat{\mathbf{x}}_{k|k-1})(\cdot)^T W_i,
\end{aligned}$$

where $W_i = (1 - W_0)/2n$ for $i = 1, 2, ..., n$.

3.2.4.2 Measurement Update

Similarly, in the measurement update step, the σ–points are chosen to

$$\begin{aligned}
\boldsymbol{\chi}_{k}^{(0)} &= \hat{\mathbf{x}}_{k|k-1} \\
\boldsymbol{\chi}_{k}^{(i)} &= \hat{\mathbf{x}}_{k|k-1} + \sqrt{\frac{n}{1-W_0}}\mathbf{P}_{i,k|k-1}^{(1/2)}, \quad i = 1, 2, ..., n \\
\boldsymbol{\chi}_{k}^{(i+n)} &= \hat{\mathbf{x}}_{k|k-1} - \sqrt{\frac{n}{1-W_0}}\mathbf{P}_{i,k|k-1}^{(1/2)}, \quad i = 1, 2, ..., n.
\end{aligned}$$

W_0 is again a tuning parameter. We can now compute the necessary moments

$$\begin{aligned}\hat{\mathbf{z}}_{k|k-1} &= \sum_{i=0}^{2n} \mathbf{h}(\boldsymbol{\chi}_k^{(i)}) W_i \\ \mathbf{P}_{xy} &= \sum_{i=0}^{2n} \left(\boldsymbol{\chi}_k^{(i)} - \hat{\mathbf{x}}_{k|k-1}\right) \left(\mathbf{h}(\boldsymbol{\chi}_k^{(i)}) - \hat{\mathbf{z}}_{k|k-1}\right)^T W_i \\ \mathbf{S}_k &= \mathbf{R}_k + \sum_{i=0}^{2n} \left(\mathbf{h}(\boldsymbol{\chi}_k^{(i)}) - \hat{\mathbf{z}}_{k|k-1}\right) (\cdot)^T W_i, \end{aligned} \tag{3.6}$$

from which the estimate and its covariance can be calculated

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{P}_{xy}\mathbf{S}_k^{-1}(\mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1}) \tag{3.7}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{P}_{xy}\mathbf{S}_k^{-1}\mathbf{P}_{xy}^T. \tag{3.8}$$

3.3 Motion Models

Motion models are used to describe the motion of moving objects. There are different ways to describe these motions, with various complexity. This section aims to explain the motion models used in this project.

3.3.1 Motion Model Discretization

In a linear continuous system, the kinematics of the state–vector $\mathbf{x}$ is described by the motion model

$$\dot{\mathbf{x}}(t) = \tilde{\mathbf{F}}\mathbf{x}(t) + \tilde{\mathbf{q}}(t), \quad \tilde{\mathbf{q}} \sim \mathcal{N}(\mathbf{0}, \tilde{\mathbf{Q}}),$$

where $\tilde{\mathbf{F}}$ describes how the derivatives, $\dot{\mathbf{x}}$, of the state vector is dependent on the states, and $\tilde{\mathbf{q}}$ is noise. Furthermore, $\mathcal{N}(.)$ denotes a Guassian (Normal) distribution. In order to use the models in a recursive filter, we need to discretize the continuous system. Rearranging and multiplying with $e^{-\tilde{\mathbf{F}}t}$ gives

$$\underbrace{e^{-\tilde{\mathbf{F}}t}\dot{\mathbf{x}}(t) - e^{-\tilde{\mathbf{F}}t}\tilde{\mathbf{F}}\mathbf{x}(t)}_{\frac{d}{dt}e^{-\tilde{\mathbf{F}}t}\mathbf{x}(t)} = e^{-\tilde{\mathbf{F}}t}\tilde{\mathbf{q}}(t),$$

we can now integrate both sides from t to $t+T$, where T is the time between two consecutive samples,

$$e^{-\tilde{\mathbf{F}}(t+T)}\mathbf{x}(t+T) - e^{-\tilde{\mathbf{F}}t}\mathbf{x}(t) = \int_t^{t+T} e^{-\tilde{\mathbf{F}}\tau}\tilde{\mathbf{q}}(\tau)d\tau.$$

Rearranging and multiplying with $e^{\tilde{\mathbf{F}}(t+T)}$ gives the final expression,

$$\mathbf{x}(t+T) = e^{\tilde{\mathbf{F}}T}\mathbf{x}(t) + \int_t^{t+T} e^{\tilde{\mathbf{F}}(t+T-\tau)}\tilde{\mathbf{q}}(\tau)d\tau,$$

which can be rewritten into a discrete recursive form

$$\begin{aligned}\mathbf{x}_k &= \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{q}_{k-1} \\ \mathbf{q}_{k-1} &\sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{k-1})\end{aligned}$$

namely

$$\mathbf{x}_k = \underbrace{e^{\tilde{\mathbf{F}}T}}_{\mathbf{F}}\mathbf{x}_{k-1} + \underbrace{\int_t^{t+T} e^{\tilde{\mathbf{F}}(t+T-\tau)}\tilde{\mathbf{q}}(\tau)d\tau}_{\mathbf{q}_{k-1}}, \tag{3.9}$$

where the noise covariance $\mathbf{Q}_{k-1}$ under the assumption of time invariant noise is

$$\begin{aligned}\mathbf{Q}_{k-1} &= Cov\{\mathbf{x}(t+T), \mathbf{x}(t)\} \\ &= Cov\{\int_t^{t+T} e^{\tilde{\mathbf{F}}(t+T-\tau)}\tilde{\mathbf{q}}(\tau)d\tau\} = ... \\ &= \int_0^T e^{\tilde{\mathbf{F}}\tau}\tilde{\mathbf{Q}}e^{\tilde{\mathbf{F}}^T\tau}d\tau.\end{aligned} \tag{3.10}$$

3.3.2 Constant Velocity

Using the Constant Velocity (`CV`) model, the velocity is modeled as a random walk process. Recall the equation for a continuous system

$$\dot{\mathbf{x}}(t) = \tilde{\mathbf{F}}\mathbf{x}(t) + \tilde{\mathbf{q}}(t), \quad \tilde{\mathbf{q}} \sim \mathcal{N}(\mathbf{0}, \tilde{\mathbf{Q}}).$$

For the `CV` model, we have that

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{p}(t) \\ \mathbf{v}(t) \end{bmatrix}$$
$$\tilde{\mathbf{F}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$
$$\tilde{\mathbf{Q}} = \begin{bmatrix} 0 & 0 \\ 0 & \sigma_Q^2 \end{bmatrix},$$

where $\mathbf{p}(t)$ is the position and $\mathbf{v}(t)$ is the velocity. Using (3.9) and (3.10), we get that

$$\mathbf{F} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \tag{3.11}$$
$$\mathbf{Q} = \sigma_Q^2 \begin{bmatrix} T^3/3 & T^2/2 \\ T^2/2 & T \end{bmatrix}. \tag{3.12}$$

3.3.3 Constant Acceleration

For the Constant Acceleration (`CA`) motion model, the acceleration is modeled as a random walk. The motion model is defined as

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{p}(t) \\ \mathbf{v}(t) \\ \mathbf{a}(t) \end{bmatrix}$$
$$\tilde{\mathbf{F}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$
$$\tilde{\mathbf{Q}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \sigma_Q^2 \end{bmatrix},$$

where $\mathbf{a}(t)$ is the acceleration. Again using (3.9) and (3.10), we obtain

$$\mathbf{F} = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}$$
$$\mathbf{Q} = \sigma_Q^2 \begin{bmatrix} T^5/20 & T^4/8 & T^3/6 \\ T^4/8 & T^3/3 & T^2/2 \\ T^3/6 & T^2/2 & T \end{bmatrix}.$$

3.4 Random Finite Sets in Multiple Object Tracking

A Random Finite Set (RFS) is a set of random variables, where the cardinality is also modeled as a random variable. Thus, given a scenario where the number of objects is unknown, and where the measurement–to–target association is also unknown, RFS is a mathematical concept that is useful when modeling such problems. Different filters may be derived when using RFS to model the MOT problem. If probability generating functionals (PGFL's) are used different filters can be derived. For example, Track–Oriented Multi–Bernoulli (TOMB) and Measurement–Oriented Multi–Bernoulli (MOMB) filters. We refer the reader to [36]–[38] for a deeper discussion about PGFL's and the derivations of MOMB and TOMB. However, by applying the conjugacy property instead of using PGFL's, a filter called Poisson Multi–Bernoulli Mixture filter can be derived.

Describing the MOT problem using the RFS framework, the set of objects and the set of measurements are modeled as RFS. The target state, $\mathbf{x} \in \mathbb{R}^{n_x}$, includes variables describing the interesting parameters of a target. For example, the target state can include position, rotations and kinematics of a target. The task in Bayesian filtering is to estimate the target state, however in MOT the scene is not restricted to only one object. Therefore, in MOT the task is rather to estimate the multi–target state, $X \in \mathcal{F}(\mathbb{R}^{n_x})$, which is a set of single target states $\mathbf{x}$. $\mathcal{F}(\mathbb{R}^{n_x})$ is the space of all finite subsets of $\mathbb{R}^{n_x}$. The measurement set is denoted $Z \in \mathcal{F}(\mathbb{R}^{n_z})$. Note that the multi–target set X and the measurement set Z can be of different sizes.

3.5 Poisson Multi-Bernoulli Mixture Filter

A possible solution for the MOT problem is the Poisson Multi–Bernoulli Mixture Filter. The aim with the PMBM filter is to find the Bayesian estimate of the state vector of all targets within the area of interest and it builds upon the idea of RFS and conjugate priors. How MOT uses RFS is explained in Section 3.4. The following paragraph explains the concept of conjugacy.

The concept of conjugacy is that the distribution of a conjugate initial prior is preserved in the prediction and measurement update steps. Thus, the prediction and posterior distributions will belong to the same family of distributions as the initial prior [39]. This property simplifies the problem and the distributions of the targets can be explained in the same manner through the recursion. Especially, the concept of conjugacy allows us to write the posterior in terms of single target prediction and measurement updates, which is less difficult to compute than the full multi–target prediction and measurement update [5]. A common distribution used for modeling the state of the object, the transition density and likelihood is the Gaussian distribution [33].

One problem in MOT is solving the data association, that is, deciding which measurement originates from which confirmed object and also, which measurements are generated by new objects. An assumption for the PMBM filter in this thesis is that an object can give rise to a maximum of one measurement, which is also known as point targets. The PMBM filter handles the data association problem by creating multiple hypotheses. A single target hypothesis is an association between a measurement to either a new or an old target.

For each old target, a single target hypothesis is created connecting the old object to each of the newly obtained measurements. Furthermore, a missed detection hypothesis is created for each old target, capturing the fact that the old target might not have given rise to any measurement. Additionally, for each obtained measurement a single target hypothesis is generated associating one measurement to a new target. One simple hypothesis tree is visualized in Figure 3.3. During each time step, all of the measurements must be assigned to an object, either an old or new target. This is done by generating global hypotheses, which is a way of combining the single target hypotheses.

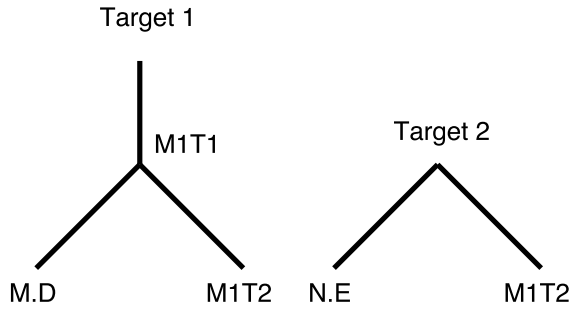


Figure 3.3: Assume that during the first time step we have made the association that Target 1 is assigned to measurement 1. During the next time step, the target can either be missed (M.D), or associated to the one measurement obtained at the second time step. Additionally, a second target is created that is either non-existent (N.E), or associated to the measurement obtained during the second time step. There are two possible global hypotheses; 1) The measurement is connected to Target 1 and Target 2 is non existence. 2) Target 1 is missed and Target 2 is associated to measurement 2.

3.5.1 The Conjugate Prior

The main idea of the PMBM filter is to model both the potentially detected targets and the undetected targets. An object is described by a Bernoulli distribution, which includes a probability of existence $r_{j,i}$ and also a state density $p_{j,i}$ of the potential target i in global hypothesis j. In this section, the time dependency has been dropped for brevity. Recall the Bernoulli PMF in (3.1), the Bernoulli RFS has set density

$$f_{j,i}(X) = \begin{cases} r_{j,i}p_{j,i}(\mathbf{x}) & X = \{\mathbf{x}\} \\ 1 - r_{j,i} & X = \emptyset \\ 0 & \text{otherwise.} \end{cases}$$

Since the task involves tracking multiple targets, the distribution of all the targets are described by a Multi–Bernoulli (MB), which is a union of independent Bernoulli distribution. Each individual set in the union captures whether or not a target exists. Furthermore, in the PMBM filter there is an uncertainty of which measurement belongs to which target, leading to the creation different global hypotheses. By building a tree of single target hypotheses and combining those into unique global hypotheses an estimate of the current objects that exists given a set of measurements can be calculated. Multiple global hypotheses generates a MB Mixture (MBM), which is the distribution of the potential targets, namely

$$f^{mbm}(X) \propto \sum_{j} \sum_{X_1 \uplus \ldots \uplus X_n = X} \prod_{i=1}^{n} w_{j,i} f_{j,i}(X_i), \tag{3.13}$$

where X_i is an empty or single target set. Here, j is the global hypothesis index, $w_{j,i}$ and $f_{j,i}$ are the weight and Bernoulli distribution of potentially detected target i, and n is the total number of potentially detected targets. Additionally, $\uplus$ is the notation for disjoint union.

Moreover, the undetected targets are modeled as a Poisson Point Process (PPP). By doing so, the number of hypotheses to cover the potential targets can be efficiently managed [5]. The idea of the Poisson part is to model potential (but undetected) targets that are either occluded or outside the sensors field of view (FOV). There are three scenarios whenever a new measurement is obtained, either the measurement originates from a new potential target, a previously detected target or clutter. The PPP is state-dependent, new objects will be detected in the edges of the FOV or around other objects, where occlusion is present. The Poisson density is defined as

$$f^{p}(X) = e^{-\int \mu(\mathbf{x}) d\mathbf{x}} [\mu(\cdot)]^{X} \tag{3.14}$$

where μ is the Poisson intensity and $[\mu(\cdot)]^{X} = \prod_{\mathbf{x} \in X} \mu(X)$, $[\mu(\cdot)]^{\emptyset} = 1$. By combining (3.13) and (3.14) we obtain the joint distribution between undetected and potential targets as

$$f(X) = \sum_{Y+W=X} f^{p}(Y) f^{mbm}(W), \tag{3.15}$$

where Y is the set of undetected targets and W is the set of potentially detected targets. Combining (3.13) and (3.15) the PMBM–density can be written as

$$f(X) \propto \sum_{Y \uplus X_1 \uplus \ldots \uplus X_n = X} f^p(Y) \sum_j \prod_{i=1}^{n} w_{j,i} f_{j,i}(X_i). \tag{3.16}$$

where X_i is an empty or single target set, and Y is the set with any cardinality subject to the constraint $Y \uplus X_1 \uplus \ldots X_n = X$.

The likelihood of a point target measurement model is characterized by the probability that an object with density $p(\mathbf{z}|\mathbf{x}_i)$ is detected, given by the probability of detection $p_d(\mathbf{x}_i)$, and if the object is missed it corresponds to a probability of $1 - p_d(\mathbf{x}_i)$. As an object can give rise to at most one measurement per time instance, the measurement from object i, Z_i, whose cardinality is greater than 1 has a probability of 0. Following this, the likelihood can be written as,

$$l(Z|\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}) = e^{-\lambda_c} \sum_{Z^c \uplus Z_1 \ldots \uplus Z_n = Z} [c(\cdot)]^{Z^c} \prod_{i=1}^{n} \hat{l}(Z_i|\mathbf{x}_i) \tag{3.17}$$

where Z_i is the set of measurement from target i, Z^c is the set of clutter measurement and it is a Poisson point process with intensity $c(\cdot)$ (also called probability hypothesis density, which can be interpreted as a multi–target density), $\lambda_c = \int c(\mathbf{z}) d\mathbf{z}$ and

$$\hat{l}(Z|\mathbf{x}) = \begin{cases} p_d(\mathbf{x}) p(\mathbf{z}|\mathbf{x}) & Z = \{\mathbf{z}\} \\ 1 - p_d(\mathbf{x}) & Z = \emptyset \\ 0 & |Z| > 1. \end{cases}$$

The sum over $Z^c \uplus Z_1 \ldots \uplus Z_n = Z$ goes through all possible sets that meet the requirement, that is, it goes through all measurement–to–target associations.

3.5.2 Time Update

Performing the time update of the conjugate prior consists of two steps, namely predicting the undetected target and the potential targets separately. The undetected objects are modelled by a Poisson process which has a predicted intensity [5]

$$\mu(\mathbf{x}) = \lambda^b(\mathbf{x}) + \int g(\mathbf{x}|\mathbf{y}) p_s(\mathbf{y}) \lambda^u(\mathbf{y}) d\mathbf{y},$$

where $\lambda^b(\mathbf{x})$ is the birth intensity, $\lambda^u(\mathbf{x})$ is the posterior intensity of the Poisson part, and $g(\mathbf{x}|\mathbf{y})$ is the transition density. The prediction of the Multi–Bernoulli mixture is

$$
\begin{aligned}
w_{j,i} &= w^u_{j,i} \\
r_{j,i} &= r^u_{j,i} \int p^u_{j,i}(\mathbf{y}) p_s(\mathbf{y}) dy \\
p_{j,i} &\propto \int g(\mathbf{x}|\mathbf{y}) p_s(\mathbf{y}) p^u_{j,i}(\mathbf{y}) d\mathbf{y},
\end{aligned}
$$

where the parameters with superscript u is the previously updated parameter for the posterior Multi–Bernoulli Mixture given in Section 3.5.3.2, see [5] for details.

3.5.3 Measurement Update

Similarly as the time update step, the measurement update also consists of two separate steps, namely a measurement update of the undetected targets and the potential targets, respectively. As the measurement update is more complex than the time update, the update of the different components are divided into sections.

3.5.3.1 Poisson Prior Update

For a target set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and a set of measurement $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$, the likelihood can be written as

$$
l(Z|X) = e^{-\lambda_c} \sum_{U \uplus Y_1 \ldots \uplus Y_m = X} [1 - P_d(\cdot)]^U \times \prod_{i=1}^{m} \tilde{l}(\mathbf{z}_i|Y_i) \tag{3.18}
$$

where

$$
\tilde{l}(\mathbf{z}|Y) = \begin{cases} p_d(\mathbf{y}) p(\mathbf{z}|\mathbf{y}) & Y = \{\mathbf{y}\} \\ c(\mathbf{z}) & Y = \emptyset \\ 0 & |Y| > 0. \end{cases}
$$

The set X can be decomposed into the set of undetected targets U, and the set $Y_1, \ldots, Y_m$ such that $U \uplus Y_1 \cdots \uplus Y_m = X$ for all possible sets. Y_i can either be empty, that is, the obtained measurement i is clutter, or single target state if the measurement belongs to a target [5].

Recall Bayes' rule (3.2) for scalars and vectors. Similarly, Bayes' rule for `RFS` is given by

$$
q(X|Z) = \frac{l(Z|X) f(X)}{\rho(Z)}, \tag{3.19}
$$

where

$$\begin{aligned}\rho(Z) &= \int l(Z|X)f(X)\delta X \\ &= \sum_{n=0}^{\infty} \int l(Z|\{\mathbf{x}_1, ..., \mathbf{x}_n\}) \times f(\{\mathbf{x}_1, ..., \mathbf{x}_n\})d(\mathbf{x}_1, ..., \mathbf{x}_n).\end{aligned}$$

Substituting (3.18) into Bayes's rule, the posterior can be calculated as

$$q^p(X|Z) \propto \sum_{U \uplus Y_1 ... \uplus Y_m = X} q^p(U) \prod_{i=1}^{m} q^p(Y_i|\mathbf{z}_i) \tag{3.20}$$

where

$$q^p(U) \propto [(1 - p_d(\cdot))\mu(\cdot)]^U \tag{3.21}$$

$$\begin{aligned}q^p(Y_i|\mathbf{z}_i) &= \tilde{l}(\mathbf{z}_i|Y_i)f^p(Y_i)/(e^{-\int \mu(\mathbf{x})d\mathbf{x}}\rho^p(\mathbf{z}_i)) \\ &= \begin{cases} r^p(\mathbf{z}_i)p^p(\mathbf{y}|\mathbf{z}_i) & Y_i = \{\mathbf{y}\} \\ 1 - r^p(\mathbf{z}_i) & Y_i = \emptyset \\ 0 & \text{otherwise} \end{cases}\end{aligned} \tag{3.22}$$

and

$$r^p(\mathbf{z}_i) = e(\mathbf{z}_i)/\rho^p(\mathbf{z}_i) \tag{3.23}$$

$$\begin{aligned}p^p(\mathbf{y}|\mathbf{z}_i) &= p_d(\mathbf{y})p(\mathbf{z}_i|\mathbf{y})\mu(\mathbf{y})/e(\mathbf{z}_i) \\ e(\mathbf{z}_i) &= \int p(\mathbf{z}_i|\mathbf{y})p_d(\mathbf{y})\mu(\mathbf{y})d\mathbf{y} \\ \rho^p(\mathbf{z}_i) &= \int \tilde{l}(\mathbf{z}_i|Y_i)f^p(Y_i)/e^{-\int \mu(\mathbf{x})d\mathbf{x}} \\ &= c(\mathbf{z}_i) + e(\mathbf{z}_i).\end{aligned} \tag{3.24}$$

The Poisson posterior in (3.20) represent the undetected objects. The Poisson component in the RFS U has an intensity given by (3.21), while the Bernoulli component in the RFS Y_i has a density given in (3.22) and a probability of existence given in (3.23).

3.5.3.2 Updating One Bernoulli Component

The posterior for one Bernoulli component is denoted by

$$q_{j,i}(X_i|Z_i) = t(Z_i|X_i)f_{j,i}(X_i)/\rho_{j,i}(Z_i) \tag{3.25}$$

where

$$\rho_{j,i}(Z_i) = \int t(Z_i|X_i)f_{j,i}(X)\delta X \tag{3.26}$$

and $t(Z_i|X_i)$ is the likelihood of a set without clutter, and is defined as

$$t(Z_i|X_i) = \begin{cases} p_d(\mathbf{x})l(\mathbf{z}|\mathbf{x}) & Z_i = \{\mathbf{z}\} \text{ and } X_i = \{\mathbf{x}\} \\ 1 - p_d(\mathbf{x}) & Z_i = \emptyset \text{ and } X_i = \{\mathbf{x}\} \\ 1 & Z_i = \emptyset \text{ and } X_i = \emptyset \\ 0 & \text{otherwise.} \end{cases} \tag{3.27}$$

From (3.27), two cases are of interest, namely when $t \neq 0$, which is when $Z_i = \{\mathbf{z}\}$ or $Z_i = \emptyset$. When $Z_i = \{\mathbf{z}\}$, t is non–zero if $X_i = \{\mathbf{x}\}$. Hence,

$$\rho_{j,i}(\{\mathbf{x}\}) \ = r_{j,i} \int p_d(\mathbf{x})l(\mathbf{x}|\mathbf{x})p_{j,i}d\mathbf{x}. \tag{3.28}$$

Combining (3.28) and (3.25), the probability of existence is 1 and the state density is proportional to $p_d(\mathbf{x})l(\mathbf{z}|\mathbf{x})p_{j,i}$ [5].

When $Z_i = \emptyset$, t is non–zero when X_i is either empty or contains one target, that is, $X_i = \emptyset$ or $X_i = \{\mathbf{x}\}$. Therefore,

$$\rho_{j,i}(\emptyset) = 1 - r_{j,i} + r_{j,i} \int (1 - p_d(\mathbf{x}))p_{j,i}(\mathbf{x})d\mathbf{x}. \tag{3.29}$$

Combining (3.25) with (3.29), the probability of existence is now given by

$$r_{j,i} \left[\int (1 - p_d(\mathbf{x}))p_{j,i}(\mathbf{x}) \right] / \rho_{j,i}(\emptyset)$$

and the state density is proportional to $(1 - p_d(\mathbf{x}))p_{j,i}(\mathbf{x})$, see details in [5].

3.5.3.3 Conjugate Prior Update

The posterior for the conjugate prior is given by combining the likelihood (3.17) and the prior (3.16) with Bayes' law (3.19). We obtain

$$q(X|Z) \propto \sum_{Y \uplus X_1 \ldots \uplus X_n = X} l(Z|Y \uplus X_1 \uplus \ldots \uplus X_n) f^p(Y) \times \sum_j \prod_{i=1}^{n} w_{j,i} f_{j,i}(X_i).$$

In [5] it is shown that this posterior can be written as the union of two independent processes, a Poisson and a Bernoulli mixture which proves that this is the conjugate with the respect to the point target measurement model.

If a newly potentially detected object is assigned to a measurement in the global hypothesis, the probability of existence for this object is set to $r_{j,i} = 1$, and its hypothesis weight is described by (3.24) and Bernoulli component by (3.22). If a new object is not considered in the global hypothesis j, its hypothesis weight is set to 1 and its probability of existence to 0 [5]. For a previously detected targets, its hypothesis weight is updated by getting multiplied by $\rho_{j,i}$ from (3.26), and depending on whether the measurement Z_i contains a detection or not, the Bernoulli component is one of the two cases described in Section 3.5.3.2.

3.5.4 Estimator

Finding the best global hypothesis corresponds to finding the global hypothesis that in total contains the highest weights in each Bernoulli component. Thus, the index for the best global hypothesis is found by

$$j^* = \arg\max_j \prod_{i=1}^{n} w_{j,i}.$$

The mean of the spatial distribution of each Bernoulli component in the global hypothesis j^* is outputed as an estimated object if the probability of existence for said Bernoulli component is above a threshold Γ.

3.6 Modeling Object Births

This section describes two ways of modeling the undetected targets. The first method explains the intensity of undetected targets as a Gaussian mixture, which can be quite computationally heavy. Instead, the birth of objects can be modelled by a uniform distribution, which is the second method explained.

3.6.1 Modelling the Poisson Component using a Gaussian Mixture

Approximating the Poisson component as a Gaussian mixture (GM) can be done by generating a large number of undetected object hypotheses, randomly spread inside the FOV, combined with a covariance matrix to cover the uncertainties. The state of the undetected object hypotheses are then updated by the time and measurement update steps according to the KF equations. In addition, the Bernoulli components are updated similarly as for potentially detected targets, see Section 3.5.2 and Section 3.5.3.2. However, approximating the Poisson component with a GM has high computational complexity, as a large number of Gaussian components has to be included in the GM to cover the whole FOV. The Gaussian components does not only have to be generated, but also updated during the time and measurement update steps.

3.6.2 Modelling the Poisson Component using a Uniform Distribution

Instead of generating samples from a Gaussian distribution and forming a mixture, new objects can be created using the uniform distribution. This is an approximation with higher or similar accuracy as a Gaussian mixture. It was shown in [40] that the accuracy is higher for uniform births if the number of components in the GM is low, and similar accuracy as a GM with large number of parameters. Thus, the trade–off between accuracy and complexity is not really existing, given the similar performance claimed in [40]. Hence, by allowing the births to be uniformly distributed in position, they can be expressed as

$$\gamma(\mathbf{x}, \mathbf{z}) = \arg\max_{\mathbf{x}} w_k U(\mathbf{z}; \boldsymbol{\beta}) \mathcal{N}(\mathbf{z}; \mathbf{h}(\mathbf{x}), \mathbf{P})$$

where $\boldsymbol{\beta}$ is the the FOV and w_k is the expected number of new targets at time k. $\mathbf{h}$ is the measurement model and $\mathbf{P}$ is the tunable birth covariance matrix. Note that instead of generating multiple objects by sampling from a Gaussian distribution to create a mixture. Here only one sample is needed, which is the measurement itself.

3.7 Complexity Reduction

One issue with data association is the computational complexity. Namely as the number of measurements and targets increases, the number of possible measurement–to–target associations increases, leading to a drastic increase in the number of possible combinations. Although, since all measurements are assigned to all objects,

many of the hypotheses will have low weight, i.e. be unlikely. So, different methods can be used in order to remove low weighted hypotheses and reduce the complexity of the filter. In this section, the most common methods are explained.

3.7.1 Gating

One way to reduce computational complexity is to pre–select only relevant measurement for doing computations. In tracking, this is used by avoiding unnecessary measurement-to-target associations. One common technique that is widely used is elliptical gating [41]. The score that is used to determine if a measurement is suitable or not is the Mahalanobis distance [41].

Given a measurement prediction $\hat{\mathbf{z}}_i$ for the ith potential target, and a measurement $\mathbf{z}_j$, their uncertainty from prediction and measurement noise is represented by the residual covariance matrix $\mathbf{S}$. The Mahalanobis distance is then calculated by

$$d_M^2 = [\hat{\mathbf{z}}_i - \mathbf{z}_j]^T \mathbf{S}^{-1} [\hat{\mathbf{z}}_i - \mathbf{z}_j]. \tag{3.30}$$

If $d_m^2 \leq G$ where G is a threshold, then the measurement j is considered relevant for object i. Another score that can be used is the Euclidean Distance

$$d_E^2 = [\hat{\mathbf{z}}_i - \mathbf{z}_j]^T [\hat{\mathbf{z}}_i - \mathbf{z}_j] \tag{3.31}$$

which, unlike (3.30), does not take the residual covariance matrix into account. However, since (3.31) does not require a matrix inversion it may be faster to calculate. Elliptical gating using the Mahablanobis distance is still widely used due to it takes the prediction uncertainty and measurement noise into account.

3.7.2 Optimal Assignment Algorithm

The Hungarian algorithm is a way of finding the best assignment by using a cost matrix [42]. As an example, say that three tasks are to be executed by three persons. Each person has a different cost for each task, see Table 3.1, and only one person can be assigned to each task. Running the Hungarian algorithm on the task would find the lowest cost of 3, by assigning *Person 1* to *Task 1*, *Person 2* to *Task 3* and *Person 3* to *Task 2*.

Table 3.1: Example of the Hungarian method. Each task shall be assigned to one person and each person can only be assigned to one task, such that the total cost is minimized. The optimal assignment is marked with bold numbers, with total cost of 3.

	Task 1	Task 2	Task 3
Person 1	**1**	2	2
Person 2	2	2	**1**
Person 3	2	**1**	2

This assignment problem is similar to assigning measurements to point targets. However, it might be of interest to not only find the absolute lowest cost assignment, but the K lowest cost assignments. An extension to the Hungarian algorithm is Murty's algorithm [43], which does exactly that.

3.8 Evaluation

This section aims to introduce the two different evaluation metrices that were used to validate the developed system.

3.8.1 Generalized Optimal Subpattern Assignment

Generalized Optimal Subpattern Assignment (`GOSPA`) is a metric that is mathematically consistent. `GOSPA` penalizes not only localization error, but also cardinality mismatch between the estimate and ground truth. That is, the `GOSPA` score will increase with increasing localization error and increasing cardinality mismatch [44]. It can be defined as

$$d_p^{(c,a)}(X,Y) \triangleq \min_{\gamma\in\Gamma} \left[\left(\sum_{(j,i)\in\gamma} d(x_i,y_j)^p + \frac{c^p}{a}(|X|+|Y|-2|\gamma|) \right) \right]^{1/p}$$

where X is the set for ground truth objects and Y is the set of estimates. $d(x_i, y_i)$ is a distance measure, γ the assignment set between $\{1,\ldots,|X|\}$ and $\{1,\ldots,|Y|\}$ in all the possible assignment sets Γ, where $|.|$ denotes cardinality. p penalizes the outliers and c determines the maximum location error. If $|X| > |Y|$ then $d_p^{(c,a)}(X,Y) \triangleq d_p^{(c,a)}(Y,X)$. For a `RFS` and setting $d(x_i, y_j)$ as the Euclidean distance, a metric is the root mean square `GOSPA` $\sqrt[p]{\mathbb{E}[d_p^{c,a}(X,Y)^p]}$ [44].

3.8.2 CLEAR–MOT

Another way to evaluate MOT is by the CLEAR-MOT performance measure. This measure contains multiple ranking criteria, but the two most used performance measures are *Multiple Object Tracking Precision* (MOTP) and *Multiple Object Tracking Accuracy* (MOTA). MOTP measures the total error in position for a matched pair between an estimate and an object, averaged over the number of total matched pairs [45], namely

$$\texttt{MOTP} = \frac{\sum_{i,t} d_t^i}{\sum_t c_t}.$$

Where d_t^i denotes the distance in assignment i at time t between an estimated object and a ground truth object, and c_t is the number of assignments at time t. t goes through all time instances and i though all assignments at each time. The other measure, MOTA, takes into account all the false negatives (fn, missed objects), false positives (fp, estimates corresponding to no true object) and the mismatches (mme, for example when the tracker changes label on two nearby objects) over all frames,

$$\texttt{MOTA} = 1 - \frac{\sum_t \texttt{fn}_t + \texttt{fp}_t + \texttt{mme}_t}{\sum_t g_t}$$

where g_t is the number of ground truth detections in frame t. Other measures that are included are for example *Mostly Tracked*, *Partly Tracked* and *Mostly Lost*. The metric in CLEAR-MOT is a way to measure closeness between trajectories.

An evaluation script is available from KITTI [7] where the CLEAR-MOT measure is calculated. This script separates the object classes between *Car* and *Pedestrian*. According to [7], due to multiple ranking criteria, there is no clear way to rank the performance of a method with respect to CLEAR-MOT, that is, there is no clear way to say that MOTA carries more weight than for example MOTP or MT. Furthermore, MOTA and MOTP are measures of closeness between trajectories, and can distinguish good trackers from bad. However it does so based on a heuristic procedure, which is unintuitive. In addition, MOTP is also mathematically inconsistent, hence MOTP is not a metric in a mathematical sense [46] even if it is presented as one in [45]. Although, as other tracking methods have evaluated their systems on the KITTI data set using this measure, it is still considered in this thesis.

Furthermore, Recall Rate and Precision Rate will be used. Recall Rate describes the sensitivity, that is, the ratio of how many positives were actually true positives. Precision Rate measures the ratio of how many negatives were correctly associated as negatives. These are defined as

$$\begin{aligned} \texttt{Precision Rate} &= \frac{\texttt{tp}}{\texttt{tp} + \texttt{fp}} \\ \texttt{Recall Rate} &= \frac{\texttt{tp}}{\texttt{tp} + \texttt{fn}}. \end{aligned}$$

4

Implementation

This section will describe the method and implementation details in a concrete manner. All code was written in MATLAB. Two different approaches were implemented, by using a state vector in the 2D image plane and in the 3D world coordinates, respectively. There are benefits and disadvantages for both methods, but ultimately tracking in 3D world coordinates yields better performance, which is presented in Section 5 and discussed in Section 6. First, general methods and choices used for both ways of modeling the problem are presented, after which specifics for the different methods are described.

4.1 Complexity Reduction

Four methods are used in order to reduce computational complexity, these are uniform births, elliptical gating, Murty's algorithm and pruning.

Uniform Births: In order to increase computational efficiency, the birth of objects were modeled as a uniform distribution in the `FOV`. In Section 3.6.2 it is stated that this approach does not suffer from lower accuracy compared to a more computationally complex method such as generating births as a `GM`. With this approach, we save computational time by not having to generate and update the components in the `GM`.

Elliptical gating: The elliptical gating described in Section 3.7.1 is used such that a measurement–to–target association is considered if the the measurement lies within three standard deviations of the innovation, which corresponds to almost 99% of the probability mass. Using (3.30) results in a Mahalanobis distance of $d_m < 3$.

Murty's algorithm: After all single target hypotheses are generated, that is, when all measurement–to–target associations after gating are computed for all old global hypotheses and hypotheses for newly detected targets are created, the new global hypotheses needs to be calculated. This is done by creating a cost matrix according to the weights for associating the measurements to the old and new objects. The weight that the measurement belongs to a new target is defined as

$$W_{new}(\mathbf{z}) = \ln(\text{diag}(\rho^p(\mathbf{z}))),$$

where $\rho^p(\mathbf{z}) = e(\mathbf{z}) + c(\mathbf{z})$ as in (3.24). The hypothesis weight is normalized with the weight that corresponds to a missed detection,

$$W(\mathbf{z}) = \frac{w_{j,i} r_{j,i} p_d \mathcal{N}(\mathbf{z}; \mathbf{h}(\bar{\mathbf{x}}_{j,i}), \mathbf{S}_{j,i}))}{(w_{j,i}(1 - r_{j,i} + r_{j,i}(1 - p_d)))}.$$

By using logarithmic weights, we obtain the normalized weight

$$\begin{aligned} W_{old}(\mathbf{z}) = w_{j,i} &+ \ln\left(r_{j,i} p_d \mathcal{N}(\mathbf{z}; \mathbf{h}(\bar{\mathbf{x}}_{j,i}), \mathbf{S}_{j,i})\right) \\ &- \ln\left(w_{j,i}(1 - r_{j,i} + r_{j,i}(1 - p_d))\right). \end{aligned}$$

With this, the cost matrix for associating measurements to old and new potential targets can be written as

$$C = -[W_{old},\ W_{new}]. \tag{4.1}$$

Note, $W_{old} \in \mathcal{R}^{m \times n_j}$ and $W_{new} \in \mathcal{R}^{m \times m}$, where n_j is the number of targets in global hypothesis j and m is the number of measurements. The K best new global hypotheses are generated by minimizing the $\text{tr}(S^T C)$ with the help of Murty's algorithm, where S is a assignment matrix where each row sums to one and each column can sum to 0 or 1. Following the advice from [5], we selected $K = \lceil N_h \cdot w_j \rceil$ number of new global hypotheses generated from each old global hypothesis, where the global hypothesis weight is $w_j = \sum_{i=1}^{N} w_{j,i}$ and N_h is the desired number of total global hypotheses.

Pruning: Targets with a probability of existence lower than a threshold Γ are simply discarded in the end of each iteration.

4.2 Similarity Score

To improve the frame–to–frame measurement–to–object assignments a similarity score based on color comparison within the bounding box was used. The idea was that if the object at time $k-1$ and at time k share the same color distribution, the target hypothesis weight should be increased. This is realized by using the negative logarithm as a cost function.

The distance measure used is the Chi–square distance (2.4). To measure if two objects are similar, an image patch was extracted around the objects. For each

image patch three (normalized) histograms were computed in the RGB color–space. Then an average distance over all three color channels was calculated using the Chi–square distance [47]. Because the image patches were in color, it was necessary to compute the distance for each color channel and then compute the average [47]. Thus, the overall similarity for an image patch is the average similarity in each color channel. Using (2.5), the weight corresponding to the similarity is given by:

$$w_{j,i,\chi^2}(\mathbf{z}) = -\ln(\chi_{avg}(\mathbf{x}_{j,i}, \mathbf{z})^2) \tag{4.2}$$

for object i in global hypothesis j and measurement z. Because the histograms were normalized to integrate to 1, the average distance is bounded by $[0, 1]$. This ensures that the updated hypothesis weight for each target does not increase unbounded. Since the positions of the bounding boxes may not be perfect and background may be present within the box, a scaling parameter is included. The color distribution is measured within some percentage of the full bounding box size, typically 90% for our system. This way, it is more likely that the color distribution of the bounding box represents the object and not background.

The more similar two objects are, the lower the Chi–square distance will be for the color histograms, converging towards zero. With the negative logarithm as a cost function, is follows that the target hypothesis weight is increased when the histograms are similar. The maximum distance between two normalized histograms is 1, it can be seen in (4.2) that the weight will evaluate to 0 in that case, thus if the color distribution between two frames are entirely different, it will not affect the target hypothesis weight.

4.3 Time Update

As mentioned in Section 3.5.2, the time update includes both the undetected targets and the potential targets. However, as the births are modeled by a uniform distribution, the undetected part is no longer affected by the time update step. Since only linear motion models are considered, the time update of the potential targets are computed by the Kalman equations, thus the predicted states are

$$p_{j,i}(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mathbf{F}\hat{\mathbf{x}}^u_{j,i}, \mathbf{F}\mathbf{P}^u_{j,i}\mathbf{F}^T + \mathbf{Q}).$$

Note that the time dependency has been dropped to avoid a cluttered notation. Furthermore, the weight of the predicted Bernoulli component is equal to the weight of the previous posterior estimate, $w_{j,i} = w^u_{j,i}$. The probability of existence is given by $r_{j,i} = r^u_{j,i} p_s(\mathbf{F}\hat{\mathbf{x}}^u_{j,i})$, where $p_s(\mathbf{x})$ is high for $\mathbf{x}$ inside `FOV` and low if $\mathbf{x}$ is outside.

Prediction of Color distribution: The motion of the color distributions is modeled by a random walk. Thus, each component of the color distribution is updated

using the **KF** equations, see Section 3.2.3.1. It is assumed that the distribution is affected by Gaussian noise.

4.4 Measurement Update

The measurement update step consists of two separate updates, namely update of the potential targets detected for the first time and update of the previous potential targets. The birth of a new object is modelled by the uniform distribution, and is updated according to the measurement equations at the time of its birth. The measurement error is modeled as a Gaussian variable. It is assumed that the probability of detection p_d is constant.

Potential targets detected for the first time: One potential target is created for each of the obtained measurements. The update equations for the potential targets detected for the first time is given in Section 3.5.3.1. Combining the update equations with the uniformly distributed births yields for measurement $\mathbf{z}$

$$\begin{aligned} e(\mathbf{z}) &= p_d w_{birth} \mathcal{N}(\mathbf{z}; \mathbf{z}, \mathbf{P}_{birth}) \\ p^p(\mathbf{y}|\mathbf{z}) &= \mathcal{N}(\mathbf{x}; \hat{\mathbf{x}}^u_{\mu,i}(\mathbf{z}), \mathbf{P}_{\mu,\mathbf{i}}{}^u), \end{aligned} \tag{4.3}$$

where $\hat{\mathbf{x}}^u_{\mu,i}(\mathbf{z}) = \mathbf{h}^{-1}(\mathbf{z})$ and $\mathbf{P}^u_{\mu,i}$ are the estimate and estimation covariance respectively, obtained by passing the measurement through the inverse measurement model and are given by (3.7) and (3.8), respectively. Note, (4.3) is originally given by

$$e(\mathbf{z}) = p_d w_{birth} \mathcal{N}(\mathbf{z}; \mathbf{h}(\hat{\mathbf{x}}^u_{\mu,i}(\mathbf{z})), \mathbf{S}_{\mu,i})$$

where $\mathbf{S}_{\mu,i}$ is the measurement noise covariance. However as the births are modeled by a uniform distribution, this would involve finding the state vector corresponding to the measurement and then compute the estimated measurement through the measurement model. Thus, (4.3) is used instead to decrease complexity. Also note that p_d, w_{birth} and $\mathbf{P}_{birth}$ are tuning parameters. A global hypothesis j considering the potential target i detected for the first time has a Bernoulli weight given by (3.24), namely $w_{j,i} = \ln(\rho^p(\mathbf{z}))$ and probability of existence given by (3.23). If global hypothesis j does not consider the new potential target, the logarithmic weight is $w_{j,i} = 0$ and the probability of existence is set to zero.

Potential targets: Recall Section 3.5.3.2 for updating one Bernoulli component. Consider an object i in global hypothesis j, the update of this component consists of two cases, namely that the object is or is not detected. The missed detection hypothesis weight is updated to

$$w_{j,i} + \ln(1 - r_{j,i} + r_{j,i}(1 - p_d)),$$

its Bernoulli component will have probability of existence

$$r_{j,i}(1 - p_d)/(1 - r_{j,i} + r_{j,i}(1 - p_d)) \tag{4.4}$$

and the target density stays the same.

For the case where the object is assumed to be detected, a single target hypothesis is created for each relevant measurement after ellipsoidal gating. For measurement z, the updated hypothesis weight is

$$w^u_{j,i} = w_{j,i} + \ln\left(r_{j,i} p_d \mathcal{N}(\mathbf{z}; \mathbf{h}(\hat{\mathbf{x}}_{j,i}), \mathbf{S}_{j,i})\right).$$

The Bernoulli component has probability of existence equal to one and state density distributed as

$$p^u_{j,i}(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \hat{\mathbf{x}}^u_{j,i}(\mathbf{z}), \mathbf{P}^u_{j,i})$$

where $\hat{\mathbf{x}}^u_{j,i}(\mathbf{z})$, $\mathbf{P}^u_{j,i}$ and $\mathbf{S}_{j,i}$ are computed using either a `KF` or a `UKF` depending on if the measurement model is linear or not, see Section 3.2.3 and Section 3.2.4.

The hypothesis weight is then adjusted according to the color comparison mentioned in Section 4.2, thus the final single target hypothesis weight is $w^u_{j,i} + w_{j,i,\chi^2}(\mathbf{z})$.

Color Distribution: The update of the color distribution is performed by using the `KF` measurement update equations given in Section 3.2.3.2. The measurement $\mathbf{z}$ contains the color distribution within the bounding–box, which means it can be compared directly with the predicted state of the color distribution. Therefore, the measurement model for the color distribution is a one–to–one mapping.

4.5 Estimator

The theory for the estimator used is explained in Section 3.5.4. However, a slight modification is needed since the weights are in the log–domain. Thus the best global hypothesis is

$$j^* = \arg\max_j \sum_{i=1}^{n} w_{j,i}.$$

Also, as mentioned in Section 3.5.4, only Bernoulli components with a probability of existence above the threshold Γ is included in the estimate. As an extension to the estimator, one can add a feature of confirmed targets. Namely, a potential object is declared as a confirmed target and included in the estimate, if it has had a probability of existence above a threshold, n_{conf} number of times. This feature allows the filter to be tuned to remove temporal false positives arising in less than n_{conf} frames. However, note that this comes with the penalty of also delaying the estimates of true objects by n_{conf} frames.

4.6 Numerical Instability

In cases where the probabilities are significantly small, using linear probabilities may not generate satisfying results due to rounding the probabilities to zero. Thus, taking the logarithm of the probabilities can be preferred as this eliminates this problem. This is useful in situations where as mentioned, the probabilities are small, but they still have to be compared to each other. Using log–probabilities applies the standard logarithmic rules, leading to a change in the way to normalize the distribution. Namely for the discrete case, the normalization factor $w_{logb,sum}$ of unnormalized weights $\tilde{w}_{logb,1}, ..., \tilde{w}_{logb,N}$ is given by

$$w_{logb,sum} = log_b \sum_{i=0}^{N} \tilde{w}_{logb,i} = \tilde{w}_{logb,0} + log_b \left(1 + \sum_{i=0}^{N} b^{\tilde{w}_{logb,i} - \tilde{w}_{logb,0}}\right),$$

where the weights are ordered such that $\tilde{w}_{logb,0} < \tilde{w}_{logb,1} < ... < \tilde{w}_{logb,N}$. The normalized weights are then given by

$$w_{logb,i} = \tilde{w}_{logb,i} - w_{logb,sum}.$$

Each iteration, the covariance matrix $\mathbf{P}$ is estimated based on statistical properties explained in Section 3.2.3. At each iteration, rounding may occur, hence losing accuracy in $\mathbf{P}$ which may lead to numerical instability, especially when inverting the innovation matrix $\mathbf{S}$, which is dependent on $\mathbf{P}$. By forcing $\mathbf{P}$ to be symmetric by setting $\mathbf{P} = \frac{\mathbf{P}+\mathbf{P}^T}{2}$, $\mathbf{P}$ becomes robust against non–invertible problems.

4.7 Overall Algorithm

The overall algorithm of the filter is presented in Algorithm 1. Note that in the case where no measurement is obtained, the filter will perform the time update step and create miss detection hypotheses for each old target and not perform the measurement update.

When a hypothesis for a potential target detected for the first time is created, or a measurement–to–target association is made, the probability of existence is compared to a threshold. By counting the number of times a potential target has been above the threshold, we can choose to include potential targets in the estimate which are considered confirmed, by being above the threshold a number of times. This improves the robustness of the filter as false positives are less likely to be seen as true objects. This also introduces a delay in including true objects in the estimate.

The more global hypotheses that are computed and kept, the less approximate the filter is. There are also methods of merging global hypotheses that are similar. However, as we are required to have IDs in order to evaluate our system with the `CLEAR-MOT` metric [45] we can not merge the similar global hypotheses as the same true object may have similar state in two global hypotheses but have different IDs. In this situation it also leads to a potential significant increase of ID switches, as the filter may output alternating global hypotheses, that is, global hypothesis j in time k, global hypothesis $j+1$ in time $k+1$ and global hypothesis j in time $k+2$ as the states and thus the weights are similar but the IDs may differ. So in order to decrease this ID switch artifact and to reduce computational complexity, we chose

to use a few number of global hypotheses.

Algorithm 1: One prediction and update step of the filter.

Prediction:
– Perform time update step for the previous potential targets, see Section 4.3.
Potential New Targets:
– Create potential target detected for the first time hypotheses for each measurement, see Section 4.4.
Update Potential Targets:
for *each global hypotheses* j **do**
  for *each target* i **do**
    – Create miss detection hypothesis, see Section 4.4.
    for *each measurement* z **do**
      if z *meets ellipsoidal gating* **then**
        – Create measurement–to–target hypothesis, including color comparison $w_{j,i,\chi^2}(\mathbf{z})$, see Section 4.4.
      end
    end
  end
  – Create cost matrix, see (4.1).
  – Run Murty's algorithm to find the $K = \lceil N_h \cdot w_j \rceil$ best new global hypotheses, see Section 4.1.
end
– Estimate target states, see Section 4.5.
– Run Murty's algorithm to find the N_h best global hypotheses, see Section 4.1.
– Perform pruning to remove Bernoulli components with a probability of existence below threshold, see Section 4.1.
– Normalize single target hypothesis weights within each global, see Section 3.1.
– Normalize global hypotheses within all global hypotheses, see Section 3.1.

4.8 Tracking in the image plane

In this section two different setups in 2D image tracking are covered. The first setup is based on the constant velocity motion model, and the second setup is the constant acceleration motion model.

4.8.1 Motion Models

A motion model is used to generate a trajectory of an object. It exists different complex motion models, which requires different amount of measurements. Two common motion models are the constant velocity model and the constant acceleration model.

4.8.1.1 Constant Velocity

The first setup in the 2D image tracking uses a `CV` motion model. The state vector used is

$$\mathbf{x} = \begin{bmatrix} x & y & v_x & v_y & w & h \end{bmatrix}^T$$

where x, y denotes pixel positions, v_x, v_y are the pixel velocities in respective direction, and w and h denotes the width and the height of the bounding box, respectively. Using the result obtained in (3.11) and (3.12) we get that the transition matrix and process covariance of the 2D pixel position and velocity is

$$\begin{aligned} \mathbf{F} &= \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \otimes \mathbf{I}_2 \\ \mathbf{Q} &= \sigma_Q^2 \begin{bmatrix} T^3/3 & T^2/2 \\ T^2/2 & T \end{bmatrix} \otimes \mathbf{I}_2, \end{aligned}$$

where $\otimes$ denotes the Kronecker product.

For this setup, the motion of the bounding box size is set as a random walk, leading to the total motion model

$$\mathbf{F} = \begin{bmatrix} \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \otimes \mathbf{I}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_2 \end{bmatrix},$$

and the process noise covariance

$$\mathbf{Q} = \begin{bmatrix} \sigma_Q^2 \begin{bmatrix} T^3/3 & T^2/2 \\ T^2/2 & T \end{bmatrix} \otimes \mathbf{I}_2 & \mathbf{0} \\ \mathbf{0} & \sigma_{BB}^2 \mathbf{I}_2 \end{bmatrix},$$

where σ_{BB}^2 is the motion noise covariance of the bounding box.

4.8.1.2 Constant Acceleration

In the second setup, the `CA` motion model is used. Similarly as in the first setup, we also want to keep track of the bounding box size and model its motion as a random walk, this results in the state vector

$$\mathbf{x} = \begin{bmatrix} x & y & v_x & v_y & a_x & a_y & w & h \end{bmatrix}^T,$$

where a_x and a_y are the pixel accelerations in each direction.

Again using (3.11) and (3.12) we get that the transition matrix and process covariance of the 2D pixel position, velocity and acceleration is

$$\mathbf{F} = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \otimes \mathbf{I}_2$$
$$\mathbf{Q} = \sigma_Q^2 \begin{bmatrix} T^5/20 & T^4/8 & T^3/6 \\ T^4/8 & T^3/3 & T^2/2 \\ T^3/6 & T^2/2 & T \end{bmatrix} \otimes \mathbf{I}_2.$$

Thus, the total transition matrix is

$$\mathbf{F} = \begin{bmatrix} \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \otimes \mathbf{I}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_2 \end{bmatrix},$$

and the corresponding process noise covariance

$$\mathbf{Q} = \begin{bmatrix} \sigma_Q^2 \begin{bmatrix} T^5/20 & T^4/8 & T^3/6 \\ T^4/8 & T^3/3 & T^2/2 \\ T^3/6 & T^2/2 & T \end{bmatrix} \otimes \mathbf{I}_2 & \mathbf{0} \\ \mathbf{0} & \sigma_{BB}^2 \mathbf{I}_2 \end{bmatrix}.$$

4.8.2 Measurement Models

The detector used in the image tracking has the structure of [32]. The measured quantities are the center pixel position, height and width of the bounding box obtained from the detector, thus the measurement model for the `CV` is

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \mathbf{I}_2,$$

and for the `CA` model

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \otimes \mathbf{I}_2.$$

4.9 Tracking in 3D World Coordinate

Tracking in 3D presents other advantages and disadvantages compared to tracking in the image plane. In this section, details about the different coordinate systems, motion and measurement models are presented. The spawning of new objects and specific details about the implementation is also covered.

4.9.1 Coordinate System

Tracking in 3D allowed us to use different coordinate systems, a global and a local coordinate system. Because of this, common motion models in a global coordinate system were able model the reality more accurately than using the same motion model in the image plane. Furthermore, using this coordinate system simplifies taking the motion of the ego–vehicle into account.

Local Coordinate Systems: Four defined local coordinate systems are used, namely the local coordinate system in the `IMU`, `Velodyne laserscanner`, $\texttt{CAM}_0$ and $\texttt{CAM}_2$, respectively. To move from one coordinate system to another a transformation matrix is used as described in Section 2.1. First, to compensate for the movement of the ego–vehicle in between two frames, the states has to be transformed from a global `IMU` coordinate system into a local `IMU` coordinate system. The transformation matrix is given by

$$\mathbf{T}_{global}^{IMU}(\mathbf{X}, \mathbf{p}_{ego}) = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x - t_x \\ y - t_y \end{bmatrix}$$

where θ is the heading of the ego–vehicle, x,y and t_x, t_y are the horizontal positions of the object and ego–vehicle respectively in the global coordinate system. The transformation is dependent on both the object state $\mathbf{X}$ and the position and orientation of the ego–vehicle ($\mathbf{p}_{ego}$). Note that this transformation is only compensating for the change of heading and position of the ego–vehicle. One could use the full 3D rotation, including the pitch and roll angles of the ego–vehicle. However, as the changes in these angles are small it would not significantly improve the accuracy. Furthermore, it is the position in the xy–plane that is of most interest and accurate measurements of the pitch and roll may not be sufficiently provided in a standard car. Additionally, note that the position of the ego–vehicle may also not be available in a standard car, however fusing the information obtained from the `IMU` and `GPS` would provide an estimation of it.

It is also necessary to move between the local coordinate systems of the ego–vehicle. These transformation matrices are denoted as $\mathbf{T}_{IMU}^{velo}$, $\mathbf{T}_{velo}^{cam_0}$ and $\mathbf{T}_{cam_0}^{cam_2}$. They are given by the calibration documents obtained from the data set [7].

Note from the transformation matrix (2.1), the fourth column is a translation vector, and the upper 3×3 matrix is a rotation matrix. Thus, a transformation in the

opposite direction is simply inverting the transformation.

The origin of the global coordinate system was defined in the position and rotation of the IMU at time $k = 0$.

4.9.2 Motion Model

The motion of objects are modelled by a CV and the bounding box size is modelled as a random walk, the state space vector is then defined as

$$\mathbf{x} = \begin{bmatrix} x & y & z & v_x & v_y & v_z & w & h \end{bmatrix}^T. \tag{4.5}$$

For the 3D state space the motion model is defined as

$$F = \begin{bmatrix} \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \otimes \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_2 \end{bmatrix}$$

and the motion covariance matrix, defined as

$$\mathbf{Q} = \begin{bmatrix} \sigma_Q^2 \begin{bmatrix} T^3/3 & T^2/2 \\ T^2/2 & T \end{bmatrix} \otimes \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \sigma_{BB}^2 \mathbf{I}_2 \end{bmatrix}$$

and σ_Q^2, σ_{BB}^2 are the covariances of the process noise.

4.9.3 Measurement Models

The detection CNN used in this setup is the one presented in [30], which generates measurements consisting of center positions and sizes of the bounding boxes and distances between the ego–vehicle and the detected objects. Given that our states are in 3D, a measurement model must be defined, which transforms the global 3D state vector into corresponding measurement estimates. First, the object position has to be transformed from the global coordinate system to the local IMU, then to Velodyne coordinates, to Cam_0 coordinates and finally Cam_2 coordinates. From which, we can perform a projection to the image plane with (2.2). Using the transformation matrices presented in Section 4.9.1, the coordinate system transformation from the global to the image plan is now given by

$$\mathbf{P} \underbrace{\mathbf{T}_{cam_0}^{cam_2} \mathbf{T}_{velo}^{cam_0} \mathbf{T}_{IMU}^{velo}}_{\mathbf{T}_{IMU}^{cam_2}} \mathbf{T}_{global}^{IMU}(\mathbf{X}, \mathbf{p}_{ego})$$

where $\mathbf{P}$ is the camera projection matrix. Note that, $\mathbf{X}$ has to be a homogenous vector, $\mathbf{X} = [x, y, z, 1]^T$. Furthermore, to obtain the pixel positions after projection a normalization is required. That is, normalize with w as stated in Section 2.2.2. Thus, obtaining the correct pixel position (u, v) from a 3D position $\mathbf{X}$ is

$$\mathbf{P}_{fwd,prj}(\mathbf{X}) = \begin{bmatrix} u \\ v \end{bmatrix} = \frac{\mathbf{P}_{1:2,*}\mathbf{X}}{\mathbf{P}_{3,*}\mathbf{X}}$$

where $\mathbf{P}_{1:2,*}$ is the first and second rows and all columns of $\mathbf{P}$ and $\mathbf{P}_{3,*}$ is the third row and all columns of $\mathbf{P}$. Thus, the transformation of a 3D position to a pixel position is

$$\mathbf{P}_{fwd,prj}(\mathbf{T}_{IMU}^{cam_2}\mathbf{T}_{global}^{IMU}(\mathbf{X}, \mathbf{p}_{ego}))$$

Additionally, a distance measurement to the detected object is obtained from the `CNN` and we have a one–to–one mapping with the bounding box size. Hence, the measurement model is

$$\mathbf{h}(\mathbf{x}, \mathbf{p}_{ego}) = \begin{bmatrix} \mathbf{P}_{fwd,prj}(\mathbf{X}_{cam_2}(\mathbf{x}, \mathbf{p}_{ego})) \\ \sqrt{x_{cam_2}(\mathbf{x}, \mathbf{p}_{ego})^2 + y_{cam_2}(\mathbf{x}, \mathbf{p}_{ego})^2 + z_{cam_2}(\mathbf{x}, \mathbf{p}_{ego})^2} \\ w \\ h \end{bmatrix},$$

where $\mathbf{x}$ is defined in (4.5), $\mathbf{X} = [x, y, z, 1]^T$ is the global 3D position of $\mathbf{x}$. $\mathbf{X}_{cam_2} = [x_{cam_2}, y_{cam_2}, z_{cam_2}, 1]^T$ is a 3D position in the $\texttt{Cam}_2$ coordinate system, given by

$$\mathbf{X}_{cam_2} = \mathbf{T}_{IMU}^{cam_2}\mathbf{T}_{global}^{IMU}(\mathbf{X}, \mathbf{p}_{ego}).$$

4.9.4 Generating Births

The Poisson component the models the undetected objects is approximated using the uniform distribution, described in Section 3.6.2. The following text will describe the procedure of how a new object is created based on a measurement.

As the state of an object is defined in the 3D global coordinate system and the obtained measurements are in the image plane, a transformation from the point in the image plane to a 3D coordinate is required. This is done by performing backward projection, see Section 2.2.3. First, note that the measured pixel coordinate is normalized; during the backward projection the pixel position is assumed to be unnormalized. Thus we need to find the unnormalized pixel position, for which we need some estimate of the distance, Z, in the camera coordinate system. This was obtained by

$$
\begin{aligned}
\phi &= \frac{35° \frac{\pi}{180}}{\texttt{FOV}_v(0.5\texttt{FOV}_v - v)} \\
c &= d\cos(\phi) \\
\theta &= \frac{\frac{\pi}{2}}{\texttt{FOV}_u(u - 0.5\texttt{FOV}_u)} \\
Z &= c \cdot \cos(\theta)
\end{aligned}
$$

where d is the distance measurement, $\texttt{FOV}_u$ and $\texttt{FOV}_v$ are the FOV sizes in u– and v–direction, u and v are pixel positions respectively. Furthermore, 35° and 90° are the vertical and horizontal opening angles (in radians) of the camera, see Figure 4.1 for an illustration of the coordinate system.

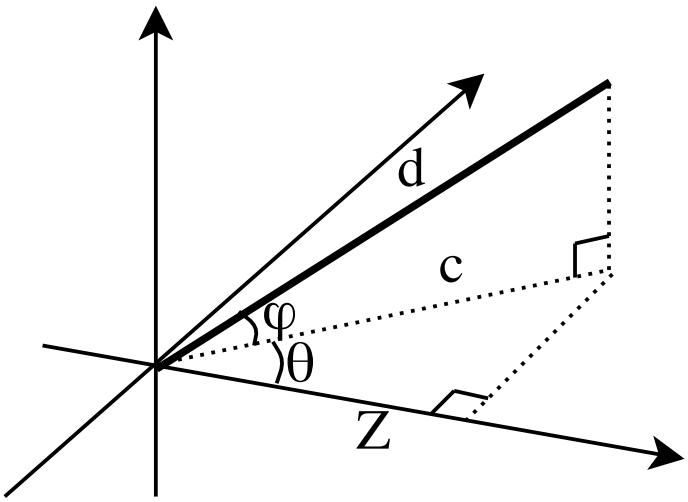


Figure 4.1: Illustration of the camera coordinate system, including distances and angles to a measurement.

The required camera projection matrix **P** was given from the calibration made by [7] and has to be specified for each sequence. The unnormalized pixel position can be computed by

$$
\mathbf{x} = w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{4.6}
$$

where w is given by

$$w = \mathbf{P}_{3,*} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

Furthermore, substituting (4.6) in the backward projection equation (2.3) gives us

$$\lambda = 1 - \mathbf{X}_4. \tag{4.7}$$

Knowing λ from (4.7), a 3D point corresponding to the pixel measurement can now be constructed by performing the matrix multiplication in the backward projection equation (2.3), thereof an object birth is generated in the $\texttt{CAM}_2$ coordinate system. The state can now be transformed into the global 3D coordinate system by first performing the transformations into the local `IMU` coordinate system and then adjusting the state according to the position and heading of the ego–vehicle, see Section 2.1 and Section 4.9.1.

Furthermore, we need to compute the 3D covariance matrix from the covariance matrix defined in the image plane. This is done in a similar manner as for the `UKF` in Section 3.2.4, namely by sampling the σ–points, $\boldsymbol{\chi}$, in the image plane and performing the backward projection for each point. The 3D covariance matrix is then given by (3.6).

The parameters of the Bernoulli component for the new objects are calculated according to the explanation in Section 4.4.

New objects appearing in the lower corners of the image is set to have an initial velocity as the ego–vehicle, as they are likely to overtake the ego–vehicle and thus is assumed to have approximately the same, or slightly higher velocity as the ego–vehicle. For objects that appear in the lower corners of the image and carries a high velocity compared to the ego–vehicle, the filter sometimes have an issue to keep old targets and thus initiates new targets instead.

4.9.5 `GOSPA` evaluation

In order to evaluate the system in the `GOSPA` metric, estimates needs to be assigned with the ground truth objects. In order to do so, the overlap between each estimated bounding box and each ground truth is computed. From this a cost matrix can be set up, where each element is given by one subtracted with the overlap between the two bounding boxes. The minimum assignment cost can then be found by using Murty's algorithm [43]. When the assignment is performed, the distance between the estimate and the ground truth is computed as the euclidean distance between the two.

In case the distance is above the threshold c [44], it is checked whether or not the estimate overlaps with a *Don't care* region. The estimate is not considered if it is within one such area. These are areas in the image where the ground truth is annotated *Don't care*, belongs to any other class than *Car*, *Cyclist* or *Pedestrian* or if the object is truncated, which happens when an object is for example leaving the scene. Furthermore if the number of estimates is higher than the number of true objects, a similar check is performed on the unassigned estimates in order to find whether any of these should not be taken into account.

5

Results

This chapter presents the result obtained from experiments. First, a brief evaluation of the 2D image tracking setup is presented. After which, a more detailed section of the obtained results of the 3D world coordinate tracking is presented. The results are further discussed in Section 6.

5.1 Tracking in the image plane

This section presents the results when tracking in the image plane. The focus lies on whether tracking in the image is suitable or not.

Tracking in the image plane comes with a drawback, namely the motion of objects are hard to describe. Thus, as objects and the ego–vehicle are moving in between frames, the motion of the objects are nonlinear. For some situations, such as when an object is moving at a near constant distance to the ego–vehicle, the motion is somewhat linear and the system manages to handle the problem. Although when an object for example is moving fast towards and past the ego–vehicle, the motion is hard to explain. That is, an object moving close to the camera moves more pixels in between frames than an object at a far distance, even though their real velocity is the same for both cases. Thus, when an object makes fast changes in distance to the ego–vehicle, the number of pixels it moves between frames also changes fast, making it hard to describe.

This setup is additionally sensitive to simultaneous changes of direction of the ego–vehicle and the object. The motion of the ego–vehicle is not taken care of separately, but is included in the estimated movement of the object. By handling the motion of the ego–vehicle in advance, the predicted state of the object can be adjusted accordingly, thus correspond better to the true position of the object. Although, compensating for the motion of the ego–vehicle in its own image plane is not obvious.

For evaluating the tracking, the metric `GOSPA` was used. The `GOSPA` score for tracking in the image plane is visualized in Figure 5.1, it is shown that the filter performs much worse than the `CNN` on multiple occasions. The mean `GOSPA` score is presented in Table 5.1. `GOSPA` takes into account the localization error and the cardinality

mismatch. Since the motion model could not predict an accurate movement of the object, the localization error increases. Sometimes, as shown in Figure 5.2c there are two estimates for one object. It follows from the increased localization error and the increased cardinality mismatch that the GOSPA score will be larger for the PMBM filter.

Table 5.1: Average GOSPA score over all sequences.

Source	GOSPA score
CNN	66.43
PMBM	81.31

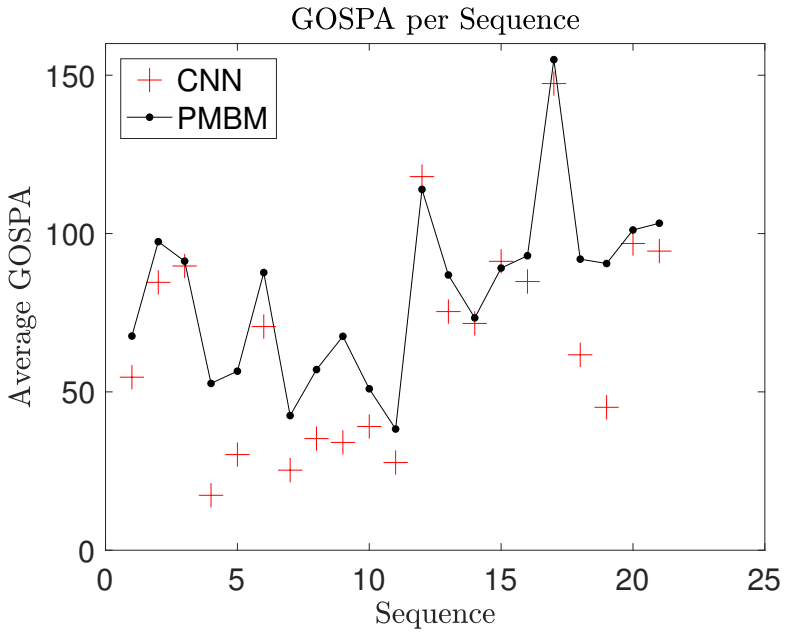
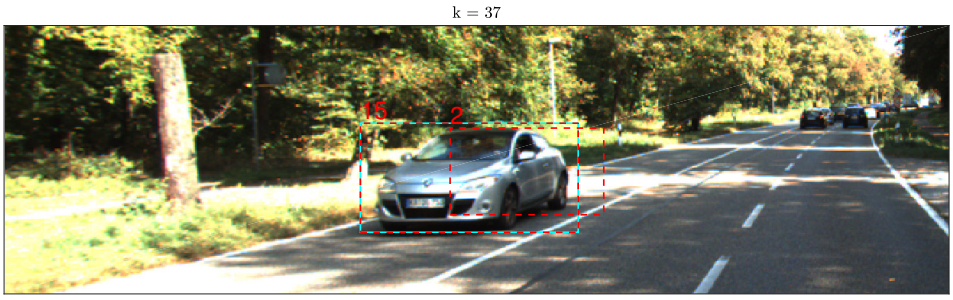


Figure 5.1: GOSPA Score for tracking in the image plane. The filter performs much worse than the CNN on multiple occasions.

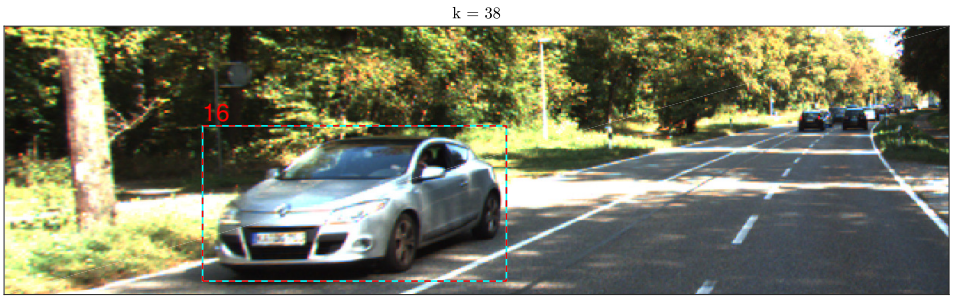
One example visualizing the problem is presented in Figure 5.2. The *Car* on the left suffers from poor prediction and ID switches even though the true motion of the *Car* is a linear movement, and the motion of the ego–vehicle is also a linear motion. First, in Figure 5.2a and in Figure 5.2b, there is no error, the *Car* is correctly tracked by the filter. However, in Figure 5.2c, the prediction is poor, thus the distance to the measurement is to large and a new object is created with ID 15. Note that the object estimate with ID 2 still is present, namely a missed detection hypothesis is created, whose Bernoulli component has a probability of existence above the threshold Γ. Finally, in Figure 5.2d, the *Car* now has ID 16 because the prediction of the object with ID 15 was inaccurate and did not match the obtained measurement at this time step. See Section 6.1 for further discussion.

(a) Correct data association.

(b) Correct data association.



(c) Incorrect data association, new object created.



(d) Incorrect data association, new object created.

Figure 5.2: Figure that for four consecutive time steps highlights the problem with tracking with image state space. The *Car* changes ID due to the filter cannot predict the future state accurately, thus creating new objects. *Note: Teal box is the measurement, Red box is the estimate from the filter.*

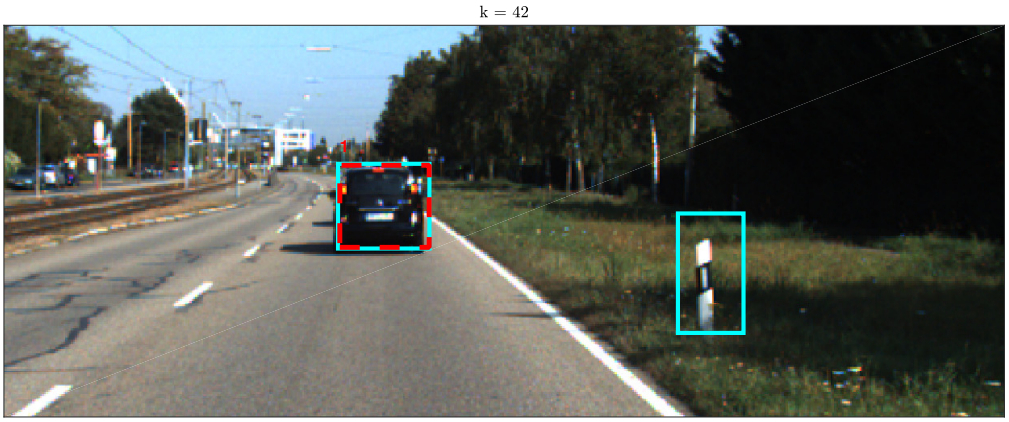
5.2 Tracking in 3D World Coordinate

This section presents different results obtained when tracking in 3D. The evaluation is divided such that the performance differences between the PMBM–filter and the CNN are distinct.

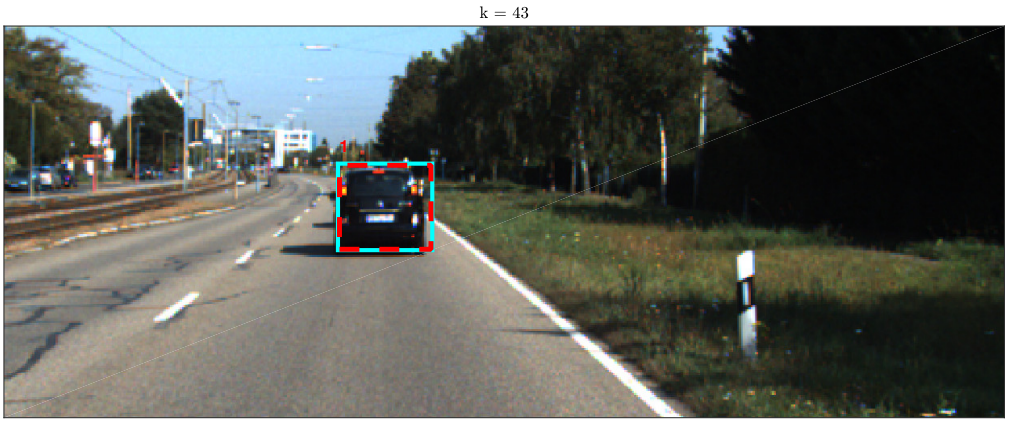
5.2.1 Removing False Positives

One issue with environmental perception is the present of false positives. Recall, a false positive is declared as an object by the system although it does not correspond to any true object, leading to a faulty estimation of the true scene. The CNN may output this kind of false positives in its process of detecting objects in the image. These false positives have to be separated from the true detections and be removed. The PMBM filter has an ability to suppress false positives due to an increase of robustness. The reason of why the filter is robust and how the level of robustness can be modified is discussed in Section 6. Moreover, the robustness against false positives provided by the filter can be seen in Figure 5.3, as a false positive is generated by the CNN in the frame displayed in Figure 5.3b, but the PMBM–filter does not include it in its estimate. The CNN classifies this false positive as a *Car*, which is of course wrong as it is a roadside pole. While roadside poles may be objects of interest, it is not within the scope of object detection. Note that the other estimates from the filter are not affected when there is a false positive occurring.

(a) Without any false positives.



(b) False positive from the `CNN` removed by the `PMBM` filter.



(c) Without any false positives.

Figure 5.3: Figure that for three consecutive time steps highlights the problem with false positives. The filter manages to suppress false positives. *Note: Teal box is the measurement, Red box is the estimate from the filter.*

5.2.2 Keeping Estimates when Measurements are Missing

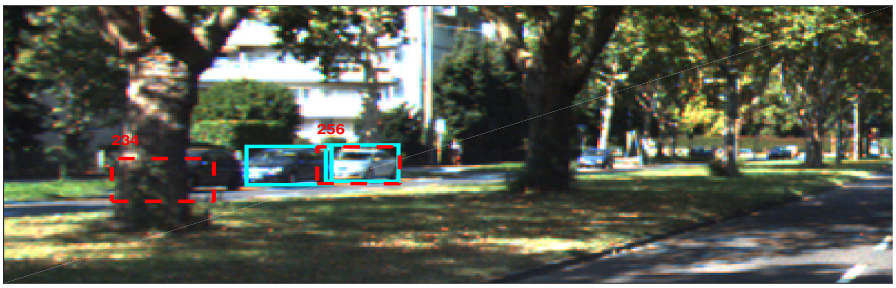
One desired property of a MOT–filter is to be more robust to missed detections. How the PMBM filter handles this issue is illustrated in Figure 5.4, the reader should focus on the *Car* with ID 234. In Figure 5.4a both the CNN and the PMBM–filter includes the *Car* denoted by ID 234 by the filter. However, in the three next time steps presented in Figure 5.4b, Figure 5.4c and Figure 5.4d the CNN does not detect the *Car* due to occlusion by a tree. One can see the performance improvement of the PMBM–filter, as its estimate still includes the *Car*. In the consecutive frame shown in Figure 5.4e the *Car* is again visible and both the CNN and the PMBM–filter includes the *Car*. When there is a missed detection the filter creates a missed detection hypothesis. This is the reason the PMBM filter is able to keep the estimate.

Note that there is a delay between the measurement and the filter output for newly detected objects, this is a result of the increased robustness to spurious false positives as can be seen in Section 5.2.1. This artifact is one of the reasons of the high number of false negatives as can be seen in Section 5.2.3, it is further discussed in Section 6.

(a) The CNN detects the *Car*.

(b) The CNN does not detect the *Car*.

(c) The CNN does not detect the *Car*.



(d) The CNN does not detect the *Car*.

(e) The CNN detects the *Car*.

Figure 5.4: Figure that for five consecutive time steps highlights the problem with missed detections and how the filter handles it. *Car* is the leftmost car with ID 234. *Note: Teal box is the measurement, Red box is the estimate from the filter.*

5.2.3 GOSPA Comparison with CNN detections

This section will summarize the results of the GOSPA evaluation metric [44], see Section 3.8.1 for information about the metric, and Section 4.9.5 for implementation details about GOSPA.

The evaluation of the CNN detections in comparison with the PBMB estimates (which are passed through the measurement model in order to be compared in the image plane) for each sequence in the KITTI tracking data set is presented in Figure 5.5.

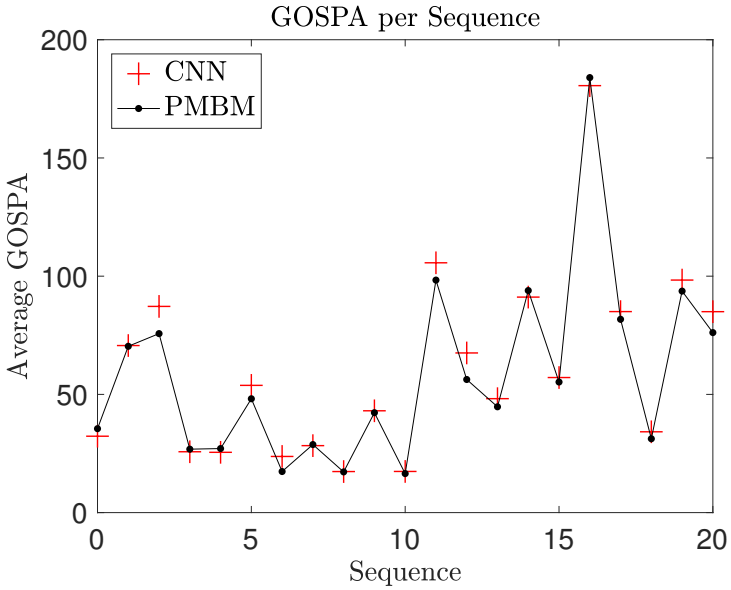


Figure 5.5: Image plane GOSPA evaluation of each sequence for both the CNN detections and the PMBM filter, averaged over all frames in each sequence.

During the GOSPA evaluation, the number of false positives and false negatives is counted, see Figure 5.6 and Figure 5.7 respectively for illustrations of these numbers. Also, the total number of false positives and false negatives is presented in Table 5.2, along with the recall and precision rates. The total number of ground truth objects through all sequences is 37910.

Figure 5.6 and Table 5.2 reveals some unexpected results, as the number of false positives is higher for the PMBM filter than for the CNN. This also leads to a lower precision rate for the PMBM compared with the CNN. As can be seen in Section 5.2.1, the filter should suppress false detections, not increase them. However by visualization one can see an unwanted artifact from the PMBM filter, which leads to an increase of false positives[1]. How the artifact arises and a discussion about it is presented in Section 6.2.3. In Table 5.2, one can also see that the number of false negatives is

[1]Note that the number of false positives corresponding to the CNN in Table 5.2 corresponds to pure false detections, while the same number for the PMBM filter corresponds mostly due to this artifact

lower for the PMBM filter than for the CNN. This is a consequence of the filter being able to keep the estimates even if the CNN fails to detect objects. As a result, the recall rate is also higher for the PMBM filter in comparison with the CNN. Note that, the PMBM filter has a delay in newly detected targets, leading to a false negative for each new target.

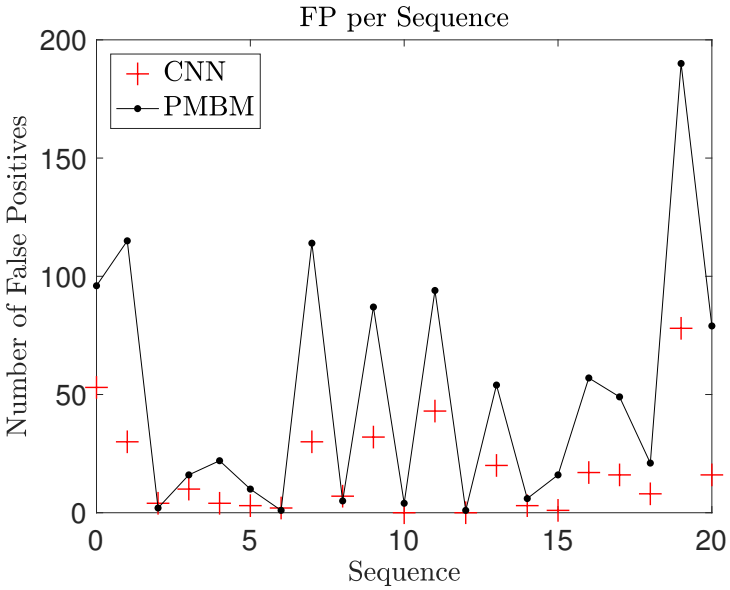


Figure 5.6: Number of false positives for each sequence for both the CNN detections and the PMBM filter.

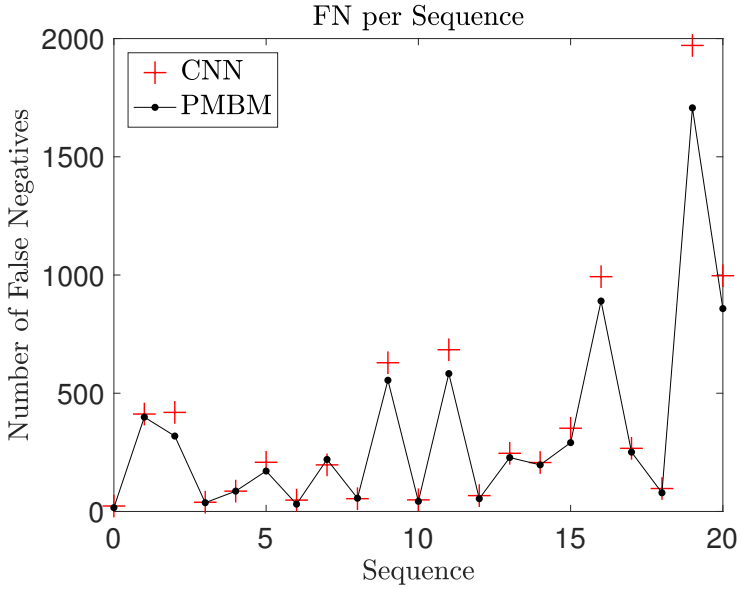


Figure 5.7: Number of false negatives for each sequence for both the CNN detections and the PMBM filter.

Table 5.2: Total number of false positives and false negatives, recall rate and precision through all the KITTI tracking sequences.

Source	Total number of false positives	Total number of false negatives	Recall rate	Precision
CNN	377	8045	0.7857	0.9874
PMBM	1039	7071	0.8082	0.9663

In addition, as the ground truth provided from KITTI contains the positions of objects (obtained by the Velodyne) in the Cam_2 coordinate system in addition to the pixel coordinates, we can also evaluate the 3D estimates of the system. Additionally, 3D measurements from CNN are computed by backward projection of the measured distance and bounding box center. The GOSPA for each sequence is shown in Figure 5.8.

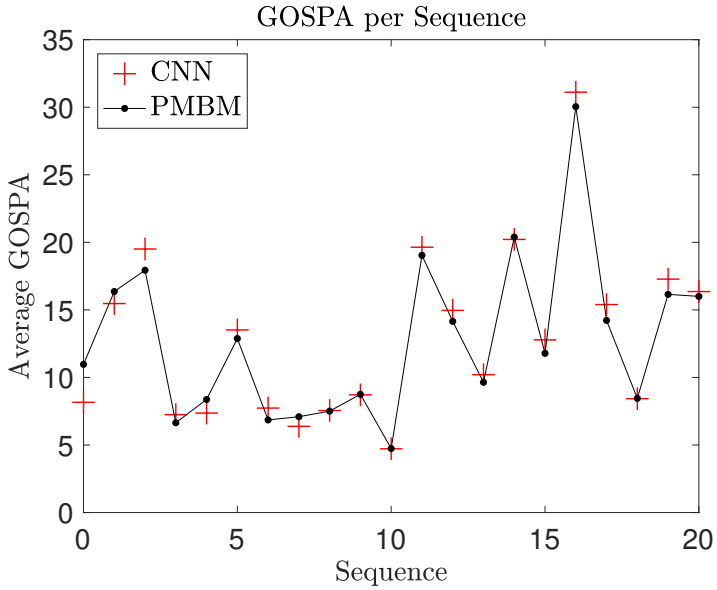


Figure 5.8: 3D GOSPA evaluation of each sequence for both the CNN detections and the PMBM filter, averaged over all frames in each sequence.

The average GOSPA score over all sequences for each of the evaluations are provided in Table 5.3. In Table 5.3 we can see that, although the unwanted artifact is present, the PMBM filter still provides a better GOSPA score than the CNN. If the unwanted phenomenon is to be removed, we would expect the GOSPA score to differ significantly more between the filter and the network.

Table 5.3: Average GOSPA score over all sequences, both in image plane and in 3D coordinates.

Source	Image plane GOSPA score	3D GOSPA score
CNN	60.85	12.99
PMBM	58.15	12.76

5.2.4 Similarity Score

In Table 5.4, the simulation results for comparing the filter using similarity score presented. For comparison, the GOSPA score for the CNN is also included. The simulations are based on two different filter setups, with and without using similarity score. GOSPA is evaluated in the image plane, hence the localization error is in pixel positions.

From Figure 5.9, it can be seen that the filter performs better than the CNN in terms of average GOSPA in both setups. Moreover, the difference between using the similarity score, or not is quite small. The largest difference occurs in sequence 12. Why sequence 12 performs better with similarity score is discussed in Section 6.2.4 However, using the similarity score produces slightly better GOSPA score on average, but this could be due to the larger difference in sequence 12. The filter is mostly better than the CNN, but in some specific sequences, for example sequence 4, the CNN performs slightly better than the filter.

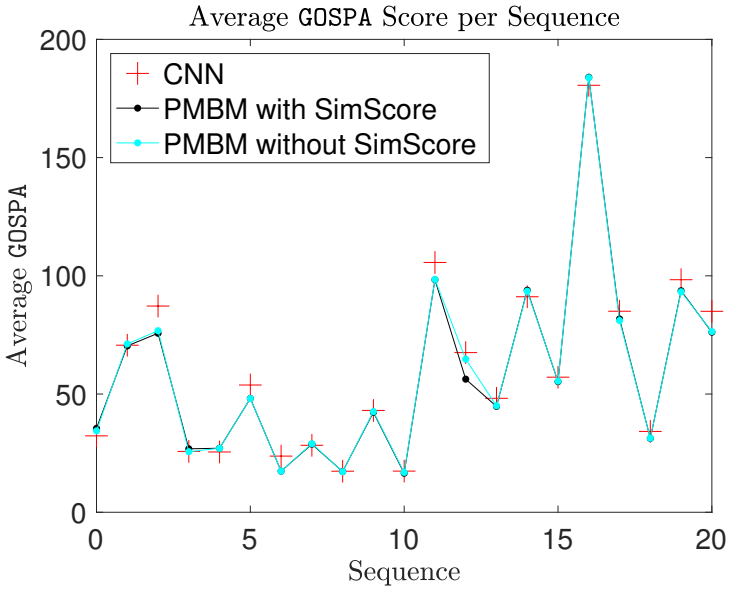


Figure 5.9: Average GOSPA in image plane, per sequence. It is noticeable that the similarity score ever so slightly improve the GOSPA score. Albeit, it is quite even between using the similarity score or not. However, compared to the CNN, it is often a higher performance in favour of the filter, for both setups.

Table 5.4: Comparison of GOSPA score averaged for all sequences between the PMBM–filter and CNN in the data set from KITTI tracking data set with and without similarity score.

Source	GOSPA Score averaged over all sequences
CNN	60.85
PMBM with similarity score	58.15
PMBM without similarity score	58.53

5.2.5 CLEAR–MOT

The CLEAR-MOT results are presented in Table 5.5, including the currently top results in the KITTI benchmark. It is noticeable that the filter scores higher for tracking *Cars* than tracking *Pedestrians*. With a total number of trajectories of 798 for *Car* and 309 for *Pedestrian*, the filter only suffers from 59 and 70 ID switches respectively.

According to the KITTI benchmark, the best trackers for the class *Car* has a MOTA and MOTP around 86% and 84%, respectively. For *Pedestrians*, MOTA and MOTP is around 58% and 72%, respectively. For the two classes, they suffer from 293 and 138 ID switches. Interestingly, the PMBM filter is not far from the top, for the class *Car* we score 71.75% and 75.75% in MOTA and MOTP, respectively. For *Pedestrians*, we score 56.35% and 73.93% MOTA and MOTP, respectively. In MOTA and MOTP there is still more ground to cover, but the ID switches in our system is lower. However, one reason for the low MOTA score we achieved, could be the large number of false positives and false negatives, that are dependent on the CNN. In Section 6.2.3 these results are further discussed and in Section 6.2.1, it is discussed why false positives arises and how false positives can be reduced.

Table 5.5: CLEAR-MOT score over all sequences for the filter. For the class *Car* the MOTA is almost 72 % and MOTP is almost 76 %. The ID switches are 59 and 70, respectively. The results shows that the filter is better at tracking *Cars* compared to tracking *Pedestrians*.

Class	MOTA	MOTP	ID switches	Recall	Precision	False Alarm Rate
Car_{PMBM}	0.7175	0.7575	59	0.7998	0.9369	0.1778
$Pedestrian_{PMBM}$	0.5635	0.7393	70	0.6674	0.8777	0.1300
Car_{KITTI}	0.8662	0.8397	293	0.9050	0.9799	0.0643
$Pedestrian_{KITTI}$	0.5815	0.7193	138	0.6412	0.9261	0.1072

6

Discussion

This chapter interprets and discusses the obtained results and evaluation methods, as well as discusses potential future work.

6.1 2D Image Tracking

During the early stages of this thesis, tracking was done in the image plane. However, it turned out quickly that tracking in the image plane was suboptimal when using common motion models such as `CV` and `CA`. Since the motion model can not predict the true motion accurately, the predicted estimate will not be associated with the latest measurement. The filter creates new objects in the location of the obtained measurements, instead of tracking the old target. Because of this, ID switches become a problem, and no continuous tracking through time is made.

Whenever a new object is created, information about the object's velocity is absent. This leads to a poor prediction from the filter, due the motion does not fit the true motion of the object. During the update step, the predicted estimate does not match the received measurement, which leads to the creation of a new object. This can be seen in Figure 5.2c. Because a new object was created, the direction of the velocity of the new object did not match the true direction, which again during the update step lead to creation of a new object since the prediction did yet again not fit with the observed state. It became clear that the motion of objects did not have a linear motion, especially for objects moving towards the ego–vehicle or objects in the edges of the `FOV`. Although note that an object may have a somewhat linear motion in specific situations, in which the filter works. Even though it is possible to understand that the performance of tracking was poor by only considering the visual evidence provided in Figure 5.2, the claim is supported by Figure 5.1 and the mean `GOSPA` score in Table 5.1.

Additionally, tracking objects in the image plane is useful for environment perception, but in order for the vehicle to make a decision to avoid an object for example, the states of each object needs to be transformed into a world coordinate system.

Some possible ways of solving the issue may be to include optical flow in order to

get some measurement of how the pixels have moved from one frame to an other. Alternatively, one could try to find a motion model describing the motion in images in a better way than standard linear models. An adaptive process noise covariance could also be considered, such that it is larger in cases where the motion is more nonlinear to compensate for the poorly fitting motion model. However as the solution is not obvious and the state still has to be transformed into a 3D world, a decision was made to instead focus on tracking directly in 3D.

6.2 3D World Coordinate Tracking

One important aspect with tracking is being able to predict trajectories of objects. In autonomous driving, it is important to safely plan the route of the car. With the information of trajectories, this will facilitate the route–planning for the ego–vehicle since there will be information where other objects will be in the near future. However, since it was difficult to model the motion of objects in the image plan, it strengthen the reason to track in 3D world coordinates as the motion model in that case is more accurate.

6.2.1 Suppressing False Positives

As previously mentioned, one issue with using a `CNN` for object detection is that it sometimes may include false positives. In Section 5.2.1 it is shown that the filter is able to filter out false positives. A reason for having multiple false positives on purpose is that you want to increase the recall rate, that is, you want to find all objects in a frame. This is done by letting the output of the `CNN` to be many bounding box proposals to hopefully include all objects. For example, the response required from the `CNN` to declare an object could be lowered. This implies that objects with a lower probability of being an object will still be a part of the output. Thus, this increases the false positive rate and might simultaneously decrease the false negative rate. Being able to suppress false positives will be an important aspect.

There are two possible ways of making the system more robust to these spurious detections. Which method to use is dependent on what you want to achieve with the filter. Either, the parameters can be set such that the probability of existence for potential targets detected for the first time is below the threshold Γ, leading to the fact that the target is not included in the estimation for the first detection of the object. This leads to a robustness against false positives occurring in a single frame. In order to increase robustness against false positives, one can apply the feature of confirmed targets mentioned in Section 4.5. Namely, a potential target is required to have a probability of existence above a threshold, n_{conf} number of times. The increase of robustness provided by the filter can be seen in Figure 5.3, as a false positive is generated by the `CNN` in the frame displayed in Figure 5.3b, but the `PMBM`–filter does not include it in its estimate. However as a penalty, setting

the filter parameters to obtain this feature also delays including true objects in the estimation. Thus, one has to choose what is desired from the filter and tune the parameters accordingly. One might consider using a range dependent value of n_{conf}, such that an object appearing close to the ego–vehicle is confirmed faster than an object further away. This might be useful since a decision to avoid collision is more crucial in the case of an object being close to the ego–vehicle.

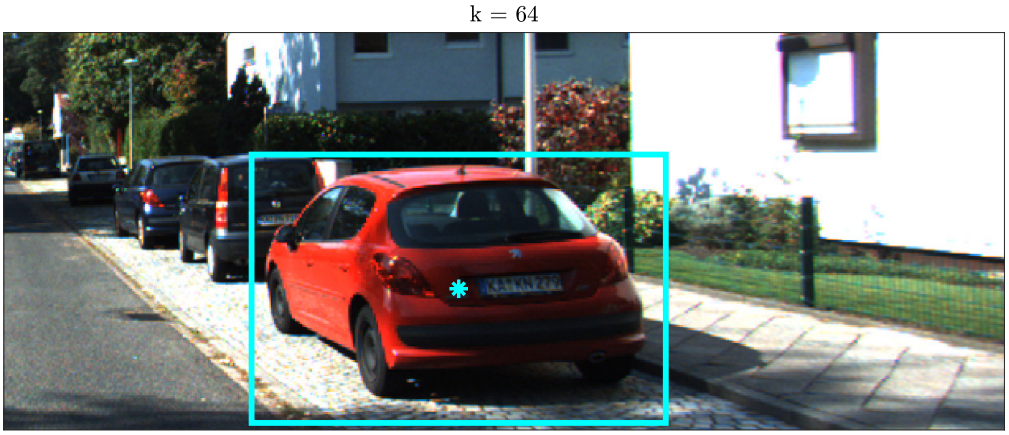
6.2.2 Keeping Estimates when Measurements are Missing

When driving in traffic, different problems are expected. Especially, problems with occlusions occur frequently, thus relying solely on the `CNN` for detecting objects may not be sufficient, as it will suffer from false negatives if the occlusion is severe enough. It can be seen in Section 5.2.2, that our filter manages to keep tracking objects when they are occluded, or when there is a missing measurement, by creating a missed detection hypothesis.

The reason for this is the probability of existence. When there is a lack of measurement for an object, the probability of existence decreases each time step, depending on the probability of detection, the probability of survival and its own probability of existence. Of course, changing p_d will enable the filter too keep tracks without measurement for different lengths of time, see Section 6.3. Furthermore, when a measurement from a missed object is received, the filter will in many cases associate the missed detection hypothesis with the returning measurement. If the motion model has managed to somewhat accurately predict the motion while there was no measurements.

6.2.3 GOSPA Comparison with `CNN` detections

Objects that are leaving the `FOV` of the camera are truncated. This entails that the center of the bounding box changes definition from what it is in a non–truncated case. For example, if only a half of a *Car* is included in the image, the network will be able to slightly stretch the bounding box to be outside the image boundaries. However, the bounding box will most likely not include the whole *Car* and thus the center of the box will be biased towards the center of the image and thus no longer represent the center of the *Car*. This phenomenon is illustrated in Figure 6.1, the car, denoted *Car*, of interest is the red, rightmost car. The definition of the bounding box center is correct for Figure 6.1a. However, it changes as the *Car* is truncated as in Figure 6.1b and in Figure 6.1c.



(a) Bounding box center well placed.

(b) Bounding box center slightly misleading for center of the `Car`.

(c) Bounding box center very misleading for center of the `Car`.

Figure 6.1: Figure illustrating the change of definition for the center of the bounding boxes of the `CNN` detections. The car of interest denoted `Car` in the sub–figure captions is the red, rightmost car. `CNN` detections of other objects are not displayed. *Note: Teal boxes are the `CNN` detections.*

If the center of a bounding box is not representing the center of an object, there will be a contradiction to the obtained measurements. The distance estimation from network [30] is trained from the `Velodyne` data, we obtain a distance measurement to the center of the *Car* and a bounding box center which is not the center of the *Car*. This change of definition and miss connection between the measured bounding box and distance leads to the previously mentioned unwanted artifact.

The predicted position of an object moving out of the `FOV` will correspond poorly with the incoming measurement, increasing the risk of creating a new object instead of connecting the measurement to the old target. Also either for an old or new target, the change of definition for the bounding box will confuse the filter to believe that the object is moving inwards the center of the `FOV` and lead to further faulty estimates. Note that this phenomenon arises before the ground truth of the object is set to truncated, this results in the estimate not to be seen as a *Don't care*.

Additionally, even if the object is marked as truncated in the ground truth, the overlap with the estimate may be to small as it moves inward the center of `FOV`. When these situations arise, we obtain a significant increase of false positives which origins from true targets. As these phenomenons are more and more likely to arise the closer the objects are to the ego–vehicle, we can see correlation in number of false positives with the scene of the sequence. As can be seen in Figure 5.6, the number of false positives is large for sequences 1, 7, 9, 11 and 19. In all these sequences, the ego–vehicle is traveling on narrow roads and it has many objects passing by close to it. Additionally note that this artifact also arise when objects are entering the `FOV` close to the ego–vehicle, as the change of definition of the bounding box center happens also in this case. Although as this is associated with an object birth, where the initial covariance is quite high and the velocity adapts faster, this is not as crucial as the previously stated situation.

As can be seen in Section 5.2.2, one benefit with adding the `PMBM` filter to the system is that we do not require having detections of each object in each frame. That is, if an object has been detected we can rely on the motion of the object if the `CNN` fails to detect the object in the next frame. This is further strengthened by Figure 5.7 and Table 5.2, where we can see a significant decrease in the number of false negatives. Note that even if false negatives naturally arise from the filter when new objects are detected due to the threshold of probability of existence, the total number of false negatives is still reduced. Also notice that the number of false negatives is high for both the CNN and the filter. This can be explained by the fact that the CNN fails to detect some objects, either by not getting a high enough response from the activation function or if the object is occluded. The filter can compensate for these cases but only if the CNN has previously detected the objects, which is not the case is all situations.

6.2.4 Similarity Score

It can be seen in Section 5.2.4, that using some sort of appearance comparison does improve the performance. Although, using similarity score is marginally better. The performance is basically the same except in one sequence, in which the similarity score provides a boost in GOSPA score. The CNN in that sequence in particular is having problems with detecting a *Pedestrian*, namely there are a lot of unstable detections. The filter is able to keep an estimate on the *Pedestrian* even though the detections are unstable. Hence, it is indicated from this behavior that using similarity score does provide some increased performance by keeping the estimates stable. In this case the similarity score provides the necessary weight increase for the filter to keep its estimate of the object. The similarity score implemented in this project is rather simple, and a discussion about more advanced solution is carried out in Section 6.3.4.

6.2.5 CLEAR–MOT

Although CLEAR-MOT is not an intuitive measure, nor is MOTP a metric due to inconsistency, we still decided to evaluate the filter using CLEAR-MOT. This enables a comparison to other tracking algorithm, as they are evaluated by following CLEAR-MOT. However, note that the score also relies on the performance of the detector. Additionally, the tracking scores presented on the KITTI website is for the *Testing* set, while we have evaluated on the *Training* set since the *Testing* set does not contain ground truth trajectories. Remark, the CNN is trained on the KITTI object detection *Training* set, which is different from the tracking *Training* set.

Since the network misses objects, it entails that the score is penalized. In Figure 5.7 and Figure 5.6 it can be seen that the number of false negatives and false positives are quite high. Thus, if those could be decrease, an overall performance increase would be achieved. However, if single test sequences or a few sequences are evaluated, in which the CNN performs well, the CLEAR-MOT score is much higher, reaching beyond the top in the KITTI benchmark. It is not fair to change the evaluation set, but the scores indicates that a CNN with better performance might entail better score.

Interestingly, in Table 5.5, the scores are lower for the class *Pedestrians* compared to the class *Car*. This is probably due to the fact that it is more difficult to detect *Pedestrians*, rather than cars. This also supports the hypothesis that a better detection system would boost the scores, for both classes. In addition, note that our method is closer to the top performing trackers when it comes to tracking *Pedestrians* than tracking of *Cars*. This may be because the artifact mentioned in Section 6.2.3 is more present for cars than pedestrians.

Furthermore, recall the discussion in Section 6.2.3, solving the problem with the artifacts would decrease the number of false positives and ID switches. This would increase the CLEAR-MOT score.

6.2.6 Delay in Newly Detected Targets and Dying Trajectories

In Figure 5.3 and Figure 5.4, it is shown in the image–sequences that there is a delay for newly detected targets, this is both wanted and not. In the case where there is a false positive, as in Figure 5.3, this is desired. Having a delay in estimating objects from the filter ensures that false positives will be removed. Additionally, this delay entails that there is a delay in true positives as well, see Figure 5.4, which is less desired. The delay exists due to the property of the filter. Each target has a probability of existence. For a new target, this probability is lower than a threshold, thus it is omitted as an output until it reaches the threshold. The threshold is changeable, in the sense that it can either accept new estimates easier, or the opposite, be more selective regarding its output. Using a low threshold would result in being more likely to output both false and true positives, as using a high threshold would cause an increased risk of false negatives but being more robust to false positive.

However, an implementation of *Confirmed targets* where an object has to be detected a number of times until it is seen as an object, entails the same problem. The trade–off seems to be between having fast, less reliable estimates or delayed but more reliable estimates. Perhaps, using the camera, distances and semantic information in each situation could determine how long the delay should be. This is further discussed in Section 6.3.3.

Furthermore, there is a delay of keeping the estimate when an object leaves the `FOV`. This happens because the filter sees this as a missing measurement, hence a missed detection hypothesis is created. As long as the probability of existence is higher than a threshold, the object will be kept as an estimate. Although, if it is possible to separate objects that are leaving the `FOV` and objects that are truly missed, for example by occlusion, the delay of keeping an estimate for an object leaving the `FOV` could be removed.

6.3 Future Work

This section aims at discussing some ideas for future work in order to improve the system. One adjustment that could be implemented in order to be less reliant on the position of the ego–vehicle, could be to instead define the coordinate system for time k in the position of the ego–vehicle in time $k-1$. If either semantic segmentation or a map is available, it would also be possible to increase the weight of new objects in areas where there are roads entering the camera `FOV`. Similarly, the weight of new objects could be slightly decreased in the center of the `FOV`. This would probably make the filter better at both keeping old targets and generating new objects.

6.3.1 Objects Moving Out of Field Of View

As has been discussed in Section 6.2.3, there is an unwanted artifact in the filter as objects are moving out of the camera's FOV. There may be some solutions to this. One could consider having a varying measurement covariance of the bounding box center obtained from the network, which is dependant on the either the bounding box size or the distance to the object. That is, the covariance for the center could increase as the bounding box size increases or the distance decreases. This may help the filter to connect the old target to the new detected even if the definition of the bounding box center changes. If the connection is made, the estimate would also move less towards the new measurement and instead rely more on the motion model, thus the filter may also be more robust against being tricked that the object is moving inwards the center of the FOV. A changing definition of the bounding box center raises the question if a bounding box is the best choice to represent objects in the image plane, see Section 6.3.2 for further discussion.

6.3.2 Bounding Box Representation

It is interesting to discuss how an object should be represented in the image plane. In Section 6.3.1, it is described that the definition of the bounding box center changes for objects leaving the FOV due to the center point changes, implying that a bounding box might not be the ideal representation.

First, when the object is far away, the bounding box covers the object, with the center of the box as the 2D–center of the object. In this scenario, the bounding box represent the front or the back of the object, depending on its orientation. If the object is seen from the side, the bounding box in some sense represent the length of the object. Furthermore, when the objects are close to end of the FOV, they are getting truncated which entails that the bounding box is only representing the visible part of the object, implying that the center point of the bounding box is no longer the 2D–center point of the object. This further implies that the definition of a bounding box is changing for different scenarios. While we do not offer a solution on how an object should be represented, it is still interesting to voice a concern regarding how objects could be represented.

6.3.3 Confirming Targets

There are both pros and cons for having confirmed targets from the filter output. A direct consequence by confirming targets is that the filter will not output an estimate until a number of measurement are obtained from the same object. A possible solution for removing the delay of newly detected objects, would be to use the confidence score the CNN outputs along with its detections. That is, if the CNN is certain that the detected object is a *Car*, *Cyclist* or *Pedestrian*, this information

could be taken into account in the filter. This way, the filter could output an estimate during the same time step as the measurement is obtained. Less confident measurements would have to be confirmed after a number of time steps. An other aspect is to use the distance to the object, such that objects appearing close to the ego–vehicle are confirmed faster than targets appear further away.

Furthermore, there might be a way to handle the delay of objects exiting the FOV. By using the velocity, and semantic information about the situation and the object, a decision if the object will return into the FOV shortly after exiting could be made. If the orientation and velocity of the object (for example an oncoming *Car* with high speed) indicates that the object will be exiting the FOV, and there is a small chance that the same object will return, then there less reason to keep the estimate when there is no measurements from that object anymore.

6.3.4 Similarity Score

In this project, a similarity score based on color distribution was used. With the advances of deep learning another similarity score using feature vectors could be implemented, with an increased complexity in the sense of how detailed features could be described. Then, not only the color similarity would be considered, but also the geometric shape, orientation and other features that makes an object unique. By training a deep network to recognize how similar two objects are, the filter could use this information in the same sense the color distribution is used by us, but with possible a higher accuracy. Intuitively, with an advance method the data association could become less computational complex, if it would be possible to say with high certainty that two measurement from two consecutive frames belongs to the same object. However, given the slight increase of performance seen in Section 5.2.4, a too complex solution might have to be reconsidered. Obviously, the similarity score is not limited to using deep networks, one could also consider using some other image processing techniques to differentiate different objects from each other.

6.3.5 Probability Map of Missed Detection

One idea for future extensions of the filter is to create a map of probability of detection, using the estimates of the filter. Namely, objects appearing in the scene may occlude other objects, that is, the object being in the way of the sensors view. If we would estimate the occupancy of objects and estimate the angle to the objects, we could create a probability map showing how likely objects are to be detected in certain areas, see Figure 6.2. In case of occlusion, the probability of detection should be decreased, leading to the fact that object appearing there does not get removed from the estimates as quickly. Objects that are not assigned to a measurement have a missed detection hypothesis, where the probability of existence is given by (4.4). The object will be included in the estimate for a longer time as it has a probability of existence above the threshold Γ for a longer time. Additionally, it will take longer

time for it to be discarded during the pruning and we will be more likely to reconnect to the old target whenever the object is detected again.

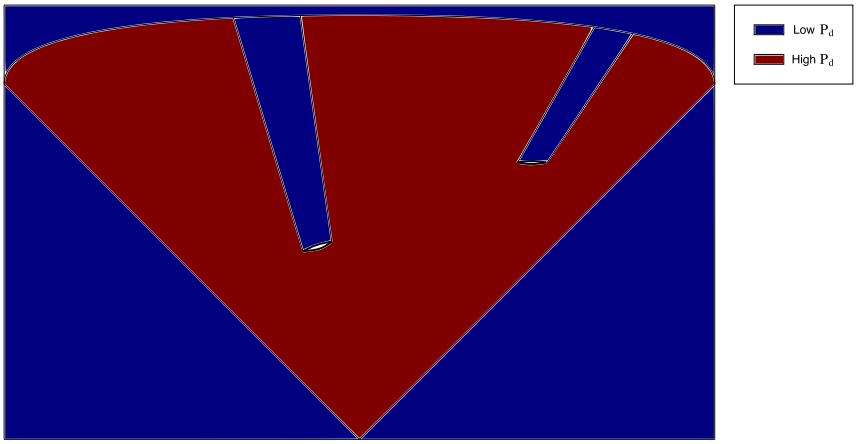


Figure 6.2: Probability map of missed detection probability. There are two objects in the camera FOV, which results in occluded areas seen from the camera. Thus, the probability of detecting objects outside the FOV or inside the occluded areas is low (blue). Similarly, the probability of detection inside the FOV but outside the occluded areas is high (red).

6.3.6 Motion Models

As it was seen in the case where tracking took place in the image plane, the problem was that the motion models did not fit the true motion of the objects, hence performing poorly. Moreover, in 3D the tracking performed well and main point being that the motion model (CV) fit the true motion of the objects more accurately. However, there exists more accurate motion models (but they are also more complex) than the CV model. Therefore, based on the results in Section 5, it is indicated that to increase the performance of the tracker a motion model that is able to describe the exact movements would possibly increase the performance. Such motion model could be the CA or a *Bicycle motion model*[48].

A Bicycle motion model takes into account the steering angle and the length of the objects from a point of rotation to the front of the object. Assuming that the point of rotation is on the back axis of the object, the Bicycle motion model will be able to describe complex movement such as a turning motion more smoothly, especially compared to the CV model. This motion model fits well with how both cyclist and cars are moving. However, the Bicycle model might not be suitable for all objects, for example pedestrians.

As said, the Bicycle model assumes a length of the object and a rotation point, but for a *Pedestrian* the length from the point of rotation to its front would essentially be zero. Therefore, motion models depending on the object classes might be necessary to fully be able to describe all motions accurately. Additionally, this approach

requires an assumption about the objects lengths, which is not always an accurate assumptions. *Cars*, for example, come in different shapes and sizes. Perhaps, extended objects where the shape is also measured would be a good starting point. This of course implies other difficulties, such as the computational complexity might get too severe when many objects are present.

6.3.7 Sensor Fusion

One crucial measurement in the 3D setup is the estimate of distance to each object. As it turns out, the distance error from the `CNN` grows fast as the distance increases as can be seen in Figure 6.3. An improvement of the distance measurement would be a fusion between the camera and a sensor that can measure the distance with a higher accuracy, for example a radar. Given measurements with higher accuracy, estimates from the filter would become more accurate and precise, thus the `GOSPA`–score would be lower due to a lower localization error.

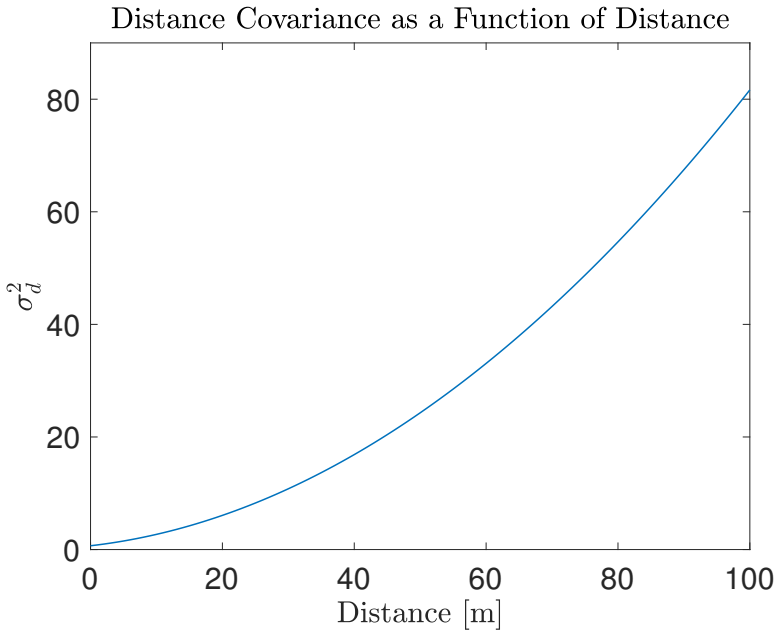


Figure 6.3: The covariance of the distance estimation from the `CNN` as a function of distance.

Another sensor that could be of interest would be an ultrasonic sensors for close range. By detecting objects that lie next to, or slightly behind the ego–vehicle, the Poisson component for undetected objects could take these measurement into account and generate a probability map for objects behind the *Car*. This would help initiating tracks for *Cars* that are about to overtake the ego–vehicle.

7

Conclusion

In this thesis, we have implemented and evaluated the PMBM filter using deep learning detectors on camera data. Method and details about the implementations are discussed. Tracking in both the 2D image plane and in a 3D coordinate system are evaluated, but 3D is the superior domain when using traditional motion models for our setting. Furthermore, the results based on CLEAR-MOT and GOSPA have been presented and it has been shown that the performance is good and in some cases solves problems that a CNN introduces. However, different artifacts have been discovered which decreases the performance of the full system, solving those may lead to a tracking solution with great potential. A discussion about said artifacts have been carried out, along with suggestions for future improvements, such as sensor fusion.

8

Bibliography

[1] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3D Object Detection for Autonomous Driving", in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 2147–2156. DOI: 10.1109/CVPR.2016.236.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[3] B. Ristic, B. T. Vo, B. N. Vo, and A. Farina, "A Tutorial on Bernoulli Filters: Theory, Implementation and Applications", *IEEE Transactions on Signal Processing*, vol. 61, no. 13, pp. 3406–3430, Jul. 2013. DOI: 10.1109/TSP.2013.2257765.

[4] K. Granström and M. Baum, "Extended Object Tracking: Introduction, Overview and Applications", *CoRR*, vol. abs/1604.00970, 2016. [Online]. Available: http://arxiv.org/abs/1604.00970.

[5] Á. F. García-Fernández, J. L. Williams, K. Granström, and L. Svensson, "Poisson multi-Bernoulli mixture filter: direct derivation and implementation", *ArXiv e-prints*, Mar. 2017. "arXiv": 1703.04264 ("cs.CV").

[6] K. Granström, M. Fatemi, and L. Svensson, "Poisson multi-Bernoulli conjugate prior for multiple extended object estimation", *ArXiv e-prints*, May 2016. arXiv: 1605.06311 [stat.CO].

[7] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite", in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[8] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets Robotics: The KITTI Dataset", *International Journal of Robotics Research (IJRR)*, 2013.

[9] Y. Xiang, A. Alahi, and S. Savarese, "Learning to Track: Online Multi-Object Tracking by Decision Making", in *The IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015.

[10] L. D. Stone, R. L. Streit, T. L. Corwin, K. L. Bell, and E. C. (.-b. collection), *Bayesian multiple target tracking*, English, Second. Boston: Artech House, 2014.

[11] D. Reid, "An algorithm for tracking multiple targets", *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, Dec. 1979. DOI: 10.1109/TAC.1979.1102177.

[12] M. Betke, Z. Wu, M. C̃. .-.-. S. D. L. of Engineering, and C. S. (.-b. collection), *Data association for multi-object visual tracking*, English. San Rafael, California: Morgan & Claypool Publishers, 2017, vol. Lecture Number 9.

[13] Y. Bar-Shalom, F. Daum, and J. Huang, "The probabilistic data association filter", *IEEE Control Systems*, vol. 29, no. 6, pp. 82–100, Dec. 2009. DOI: 10.1109/MCS.2009.934469.

[14] S. Challa, B. (.-b. collection), K. (.-b. collection), and I. Books24x7, *Fundamentals of object tracking*, English. Cambridge;New York; Cambridge University Press, 2011.

[15] L. Jiang and S. S. Singh, "Tracking multiple moving objects in images using Markov Chain Monte Carlo", *ArXiv e-prints*, Mar. 2016. arXiv: 1603.05522 [stat.AP].

[16] B. N. Vo, B. T. Vo, N. T. Pham, and D. Suter, "Joint Detection and Estimation of Multiple Objects From Image Observations", *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5129–5141, Oct. 2010. DOI: 10.1109/TSP.2010.2050482.

[17] L. Wang, T. Liu, G. Wang, K. L. Chan, and Q. Yang, "Video Tracking Using Learned Hierarchical Features", *IEEE Transactions on Image Processing*, vol. 24, no. 4, pp. 1424–1435, Apr. 2015. DOI: 10.1109/TIP.2015.2403231.

[18] X. Jia, H. Lu, and M. H. Yang, "Visual tracking via adaptive structural local sparse appearance model", in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 1822–1829. DOI: 10.1109/CVPR.2012.6247880.

[19] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking The Untrackable: Learning To Track Multiple Cues with Long-Term Dependencies", *ArXiv e-prints*, Jan. 2017. arXiv: 1701.01909 [cs.CV].

[20] Y. Xiang, A. Alahi, and S. Savarese, "Learning to Track: Online Multi-object Tracking by Decision Making", in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 4705–4713. DOI: 10.1109/ICCV.2015.534.

[21] Y. m. Song and M. Jeon, "Online multiple object tracking with the hierarchically adopted GM-PHD filter using motion and appearance", in *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, Oct. 2016, pp. 1–4. DOI: 10.1109/ICCE-Asia.2016.7804800.

[22] H. Guan, X. Xue, and A. Zhiyong, "Online video tracking using collaborative convolutional networks", in *2016 IEEE International Conference on Multimedia and Expo (ICME)*, Jul. 2016, pp. 1–6. DOI: 10.1109/ICME.2016.7552897.

[23] R. Smith, M. Self, and P. C. Cheeseman, "Estimating Uncertain Spatial Relationships in Robotics", *CoRR*, vol. abs/1304.3111, 2013. [Online]. Available: http://arxiv.org/abs/1304.3111.

[24] O. Pink, F. Moosmann, and A. Bachmann, "Visual features for vehicle localization and ego-motion estimation", in *2009 IEEE Intelligent Vehicles Symposium*, Jun. 2009, pp. 254–260. DOI: 10.1109/IVS.2009.5164287.

[25] A. Sanna, B. Pralio, F. Lamberti, and G. Paravati, "A Novel Ego-Motion Compensation Strategy for Automatic Target Tracking in FLIR Video Sequences taken from UAVs", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 45, no. 2, pp. 723–734, Apr. 2009. DOI: 10.1109/TAES.2009.5089552.

[26] K. R. J. Mukherjee; "Image Similarity Measure using Color Histogram, Color Coherence Vector, and Sobel Method", *International Journal of Science and Research (IJSR)*, vol. 2, no. 1, pp. 538–543, Jan. 2013.

[27] O. Pele and M. Werman, "The Quadratic-Chi Histogram Distance Family", English, in. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 6312, ch. 2, pp. 749–762.

[28] X. Chen, L. An, and B. Bhanu, "Multitarget Tracking in Nonoverlapping Cameras Using a Reference Set", *IEEE Sensors Journal*, vol. 15, no. 5, pp. 2692–2704, May 2015. DOI: 10.1109/JSEN.2015.2392781.

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.

[30] A. Krishnan and J. Larsson, "Vehicle detection and road scene segmentation using deep learning", Master's thesis, 2016.

[31] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional Neural Networks for Speech Recognition", *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 22, no. 10, pp. 1533–1545, Oct. 2014. DOI: 10.1109/TASLP.2014.2339736. [Online]. Available: http://dx.doi.org.proxy.lib.chalmers.se/10.1109/TASLP.2014.2339736.

[32] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection", *CoRR*, vol. abs/1607.07155, 2016. [Online]. Available: http://arxiv.org/abs/1607.07155.

[33] F. G. K. Granström C. Lundquist and U. Orguner, "Random Set Methods: Estimation of Multiple Extended Objects", *IEEE Robotics Automation Magazine*, vol. 21, no. 2, pp. 73–82, Jun. 2014. DOI: 10.1109/MRA.2013.2283185.

[34] S. Särkkä, *Bayesian Filtering and Smoothing*, ser. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2013. DOI: 10.1017/CBO9781139344203.

[35] M. S. Grewal and dawsonera (e-book collection), *Kalman filtering: theory and practice using MATLAB*, English, 3rd. Oxford: Wiley-Blackwell, 2008.

[36] J. L. Williams, "Marginal multi-bernoulli filters: RFS derivation of MHT, JIPDA, and association-based member", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 3, pp. 1664–1687, Jul. 2015. DOI: 10.1109/TAES.2015.130550.

[37] R. Mahler, "Statistics 102; for Multisource-Multitarget Detection and Tracking", *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 3, pp. 376–389, Jun. 2013. DOI: 10.1109/JSTSP.2013.2253084.

[38] J. L. Williams, "An Efficient, Variational Approximation of the Best Fitting Multi-Bernoulli Filter", *IEEE Transactions on Signal Processing*, vol. 63, no. 1, pp. 258–273, Jan. 2015. DOI: 10.1109/TSP.2014.2370946.

[39] B. T. Vo and B. N. Vo, "Labeled Random Finite Sets and Multi-Object Conjugate Priors", *IEEE Transactions on Signal Processing*, vol. 61, no. 13, pp. 3460–3475, Jul. 2013. DOI: 10.1109/TSP.2013.2259822.

[40] M. Beard, B. T. Vo, B.-N. Vo, and S. Arulampalam, "A Partially Uniform Target Birth Model for Gaussian Mixture PHD/CPHD Filtering", English, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 4, pp. 2835–2844, 2013.

[41] V. D. Nguyen and T. Claussen, "Reducing computational complexity of gating procedures using sorting algorithms", in *Proceedings of the 16th International Conference on Information Fusion*, Jul. 2013, pp. 1707–1713.

[42] *Hungarian algorithm*, May 2017. [Online]. Available: https://en.wikipedia.org/wiki/Hungarian_algorithm.

[43] K. G. Murty, "An Algorithm for Ranking all the Assignments in Order of Increasing Cost", *Operations Research*, vol. 16, no. 3, pp. 682–687, 1968. [Online]. Available: http://www.jstor.org/stable/168595.

[44] A. Sajana Rahmathullah, Á. F. García-Fernández, and L. Svensson, "Generalized optimal sub-pattern assignment metric", *ArXiv e-prints*, Jan. 2016. arXiv: 1601.05585.

[45] K. Bernardin and R. Stiefelhagen, "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics", *EURASIP Journal on Image and Video Processing*, vol. 2008, no. 1, p. 246 309, May 2008. DOI: 10.1155/2008/246309. [Online]. Available: http://dx.doi.org/10.1155/2008/246309.

[46] J. Bento, "A metric for sets of trajectories that is practical and mathematically consistent", English, 2016.

[47] P. L. Mazzeo, L. Giove, G. M. Moramarco, P. Spagnolo, and M. Leo, "HSV and RGB color histograms comparing for objects tracking among non overlapping FOVs, using CBTF", English, 2011, pp. 498–503.

[48] D. Schramm, M. Hiller, and R. Bardini, "Single Track Models", in *Vehicle Dynamics: Modeling and Simulation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 223–253. DOI: 10.1007/978-3-540-36045-2_10. [Online]. Available: https://doi.org/10.1007/978-3-540-36045-2_10.