DGK Ausschuss Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 953

Christian Bader

Multi-Sensor Multi-Object Detection and Tracking for ADAS, Applied to Trucks

München 2025

Verlag der Bayerischen Akademie der Wissenschaften, München

ISSN 0065-5325 ISBN 978-3-7696-5365-6

DGK Ausschuss Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften

Reihe C Dissertationen Heft Nr. 953

Multi-Sensor Multi-Object Detection and Tracking for ADAS, Applied to Trucks

A thesis accepted by the Faculty of Aerospace Engineering and Geodesy
of the University of Stuttgart
in fulfillment of the requirements for the degree of
Doctor of Engineering Sciences (Dr.-Ing.)

by

Christian Bader
born in Pforzheim

München 2025

Verlag der Bayerischen Akademie der Wissenschaften, München

ISSN 0065-5325 ISBN 978-3-7696-5365-6

Adresse des Ausschusses Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften:

Ausschuss Geodäsie (DGK) der Bayerischen Akademie der Wissenschaften
Alfons-Goppel-Straße 11 ● D – 80 539 München
Telefon +49 – 89 – 23 031 1113 ● Telefax +49 – 89 – 23 031 - 1283 / - 1100
e-mail post@dgk.badw.de ● http://www.dgk.badw.de

Main referee: Prof. Dr.-Ing. habil. Volker Schwieger
Co-referee: Prof. Dr.-Ing. Hans-Christian Reuss

Day of the exam: 11.10.2024

Diese Dissertation ist auf dem Server des Ausschusses Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften unter <http://dgk.badw.de/>
sowie auf dem Publikationsserver der Universität Stuttgart unter
<https://elib.uni-stuttgart.de/items/9b598f15-2988-4e8d-83e7-14b1937e6cd1> elektronisch publiziert

ISSN 0065-5325 ISBN 978-3-7696-5365-6

Contents

List of Tables VII

List of Figures IX

Zusammenfassung XI

Abstract XIII

1 Introduction 1
- 1.1 Advanced driver assistance systems . . . 1
- 1.2 Challenges of multi-sensor multi-object detection and tracking for trucks . . . 2
- 1.3 Thesis outline . . . 4
- 1.4 Contributions . . . 6

2 Environment perception basics and sensors 8
- 2.1 Overview of proposed perception pipeline . . . 8
- 2.2 Environment representations . . . 9
  - 2.2.1 Occupancy grid . . . 9
  - 2.2.2 Object map . . . 10
- 2.3 Object representations . . . 11
- 2.4 Perception sensors . . . 12
  - 2.4.1 Lidar sensors . . . 13
  - 2.4.2 Cameras and artificial vision sensors . . . 15
  - 2.4.3 Radar sensors . . . 16
  - 2.4.4 Comparison . . . 17
- 2.5 Coordinate systems . . . 17

3 Object detection 20
- 3.1 Object detection overview . . . 20
  - 3.1.1 Deep learning in object detection . . . 21
- 3.2 Lidar-based object detection . . . 24
  - 3.2.1 Traditional approaches . . . 24
    - 3.2.1.1 Ground extraction and object segmentation . . . 25
    - 3.2.1.2 Oriented bounding box generation . . . 26
    - 3.2.1.3 Point cloud classification . . . 26
  - 3.2.2 Deep learning approaches . . . 27
    - 3.2.2.1 Projection approaches . . . 29
    - 3.2.2.2 Volumetric approaches . . . 29
    - 3.2.2.3 Point-based approaches . . . 29
    - 3.2.2.4 Performance comparison of Lidar-based deep learning detection . . . 30
  - 3.2.3 Two-stage detection approach for safety critical Lidar object detection . . . 30
    - 3.2.3.1 First stage: Traditional object detection . . . 32
    - 3.2.3.2 Second stage: Classification and OBB regression . . . 35
- 3.3 Camera-based object detection . . . 40
  - 3.3.1 Monocular CNN-based 3D detection algorithms . . . 40
  - 3.3.2 Distance estimation using ground projection . . . 41
  - 3.3.3 Multi-stage object detection for close ranges at trucks . . . 43

3.4 Radar-based object detection . . . 47

4 Basics of multi-object tracking and multi-sensor fusion 48
4.1 Tracking approaches . . . 49
4.2 Bayesian filters . . . 49
4.3 Kalman filter . . . 50
4.3.1 Kalman filter extensions . . . 52
4.3.2 Kalman filter in MOT . . . 52
4.3.2.1 Data association . . . 53
4.3.2.2 Track management . . . 55
4.4 Random finite set filters . . . 56
4.4.1 Random finite sets . . . 57
4.4.2 Multi target Bayes filter . . . 58
4.4.3 PHD filter . . . 59
4.4.3.1 The GM-PHD filter equations . . . 60
4.4.4 PHD filter in MOT . . . 61
4.4.4.1 Adaptive birth intensity . . . 62
4.4.4.2 Labeled tracking . . . 62
4.4.4.3 Typical implementation . . . 62
4.4.4.4 State-dependent detection probability . . . 63
4.5 Classification tracking . . . 64
4.5.1 Bayesian framework for classification tracking . . . 64
4.5.2 Dempster-Shafer framework for classification tracking . . . 65
4.6 Sensor fusion for MOT . . . 66
4.6.1 State of the art and overview . . . 66
4.6.1.1 Levels of fusion . . . 66
4.6.1.2 Fusion frameworks . . . 70
4.6.2 Temporal and spatial alignment . . . 70
4.6.2.1 Spatial alignment . . . 70
4.6.2.2 Temporal alignment . . . 71

5 Proposed multi-sensor multi-object tracking pipeline 73
5.1 Fusion framework overview . . . 73
5.2 Temporal management system . . . 76
5.3 Track definition and motion model . . . 77
5.4 Spatial measurement mapping using turning functions . . . 79
5.4.1 Spatial measurement mapping and pseudo measurement generation . . . 79
5.5 Sensor-based tracking parameter models . . . 84
5.5.1 Detection probability . . . 86
5.5.1.1 Detection probability model . . . 87
5.5.1.2 Detection probability implementation . . . 91
5.5.2 Clutter density . . . 92
5.6 Classification fusion . . . 95
5.7 GM-PHD filter implementation . . . 98
5.7.1 Tags and classification extensions . . . 98
5.7.2 Step 0: Initialization . . . 98
5.7.3 Step 1: Prediction . . . 99
5.7.4 Step 2: Update . . . 100
5.7.5 Step 3: Pruning . . . 102
5.7.6 Step 4: Merging . . . 102
5.7.7 Step 5: Track extraction . . . 102

5.8 Kalman filter implementation . . . 103
5.9 Track confirmation strategy . . . 103

6 Evaluation of multi-object detection and tracking 106
6.1 Datasets . . . 106
6.1.1 KITTI dataset . . . 107
6.1.2 Truck dataset . . . 108
6.2 Metrics for object detection . . . 109
6.3 Evaluation of the two-stage Lidar detection algorithm . . . 110
6.4 Evaluation of the camera detection algorithm for downward looking cameras at trucks . . . 113
6.5 Evaluation metrics for object tracking . . . 114
6.5.1 The OSPA metric . . . 116
6.5.2 The $OSPA^{(2)}$ metric . . . 117
6.5.3 The HOTA metric . . . 117
6.5.4 MOT metric summary . . . 119
6.6 Simple multi-object filter comparison . . . 119
6.7 Fusion framework evaluation . . . 124
6.7.1 Evaluation areas . . . 125
6.7.2 Comparison of GM-PHD and Kalman filter using real-world data . . . 126
6.7.3 Evaluation of sensor-based parameter models . . . 129
6.7.4 Comparison of the proposed GM-PHD implementation to original implementation . . . 131
6.7.5 Sensor failure analysis . . . 132
6.7.6 Comparison of performance in different operational domains . . . 133
6.7.7 Comparison of classification approaches . . . 133

7 Conclusion and future work 135
7.1 Conclusion . . . 135
7.2 Future work . . . 136

8 Acronyms

Bibliography II

List of Tables

2.1 Characteristics of different Lidar sensors. 15
2.2 Characteristics of different radar sensors. 17
2.3 Comparison of the capabilities of different sensor types. 17

3.1 Overview of object detection approaches using different sensors. 21
3.2 Comparison of categories of deep learning Lidar object detection approaches. 28
3.3 Comparison of AP of different Lidar detection methods on the KITTI 3D detection test set. . . 30

4.1 Comparison of DA filters. 54
4.2 Advantages and disadvantages of low-level fusion approaches. 68
4.3 Advantages and disadvantages of feature-level fusion approaches. 69
4.4 Advantages and disadvantages of high-level fusion approaches. 69

5.1 Operational domain parameters for detection probability of class "Car". 89
5.2 Operational domain parameters for clutter density of class "Car". 95

6.1 Comparison of KITTI MOT dataset and truck dataset. 106
6.2 Overview of sensor interfaces for the KITTI dataset. 107
6.3 Overview of sensor interfaces for the truck dataset. 108
6.4 Overview of individual scenes of the truck dataset. 109
6.5 AP and unclassified recall results of the two-stage approach on the KITTI Dataset. 112
6.6 Comparison of classification accuracy and inference time using the ModelNet40. 112
6.7 Comparison of classification accuracy and inference time using the STC. 113
6.8 Comparison of mAP scores of detection approaches (KITTI and truck data). 114
6.9 Overall mean results of filter comparison on KITTI dataset. 121
6.10 Overview of the experiments on KITTI and truck dataset, alongside with their goals. 125
6.11 Comparison of runtime and HOTA for different gating distances. 132
6.12 Comparison of runtime and HOTA for detection probability models. 132

List of Figures

1.1 Different types of tucks that can share parts of the electronic components. 3
1.2 Visualization of separately sprung cabin. 4
1.3 Simplified scheme of perception processing steps assigned to the chapters of this thesis. 5

2.1 General pipeline used for this thesis. 9
2.2 Example of occupancy grid and inverse sensor model. 10
2.3 Examples of different object representations. 11
2.4 Example for truck sensor setup with SRR in red, LRR in blue and camera in green. 13
2.5 Magnetic spectrum including the ranges of different sensing principles. 13
2.6 Basic working principle of a Lidar sensor. 14
2.7 Examples of different measurement types of Lidar sensors. 14
2.8 Comparison of different Lidar scanning principles. 15
2.9 Maximum detection distance over camera resolutions of different horizontal field of views. . . 16
2.10 Vehicle and cabin coordinate system definition. 18
2.11 Sensor coordinate system examples for Camera and Lidar. 18

3.1 Schematic representation of a perceptron. 22
3.2 Visualization of different activation functions. 22
3.3 Examples for fully connected layers in NN. 23
3.4 Example for CNN with one hidden convolutional layer and one hidden max-pooling layer. . . 23
3.5 Four steps of a traditional Lidar object detection pipeline. 25
3.6 Example of a point cloud segment with OBB. 31
3.7 Overview of two-stage Lidar detection approach. 32
3.8 Ground extraction process of two-stage Lidar detection approach. 33
3.9 Segmentation of two-stage Lidar detection approach. 34
3.10 OBB generation of two-stage Lidar detection approach. 35
3.11 NN architecture of two-stage Lidar detection approach. 35
3.12 MVFE layer and encoder network description of two-stage Lidar detection approach. 37
3.13 NN backbone of two-stage Lidar detection approach. 37
3.14 NN head branches of two-stage Lidar detection approach. 38
3.15 Visualization of NN calculated OBB features. 38
3.16 Visualization of different groups of monocular 3D camera detection. 41
3.17 Distance estimation using the projection to the ground at trucks. 42
3.18 FoV areas of mirror classes for trucks. 43
3.19 Front mirror replacement camera position and rectified example image. 44
3.20 Simulation of distance estimation error of downward looking camera. 44
3.21 Scheme of proposed multi-stage 3D OBB generation approach. 45
3.22 OBB fitting of proposed 3D OBB generation approach. 46
3.23 Example of radar and Lidar detections from various sensors. 47

4.1 Scheme of Kalman filter calculation steps. 51
4.2 Scheme of a KF in an MOT framework including track management and data association. . . . 53
4.3 Example of data association and gating. 54
4.4 Intuition of GM-PHD filter shown by visualization of intensity. 61
4.5 Scheme of high-level, feature-level and low-level fusion concepts. 67
4.6 Scheme of direct update sensor fusion without delay. 71
4.7 Scheme of sensor fusion with different sensor delays. 72

5.1 Overview of the proposed fusion framework. 75
5.2 Scheme of the temporal management, that is responsible for the correct buffering. 77
5.3 Visualization of ego vehicle and states of track. 78
5.4 Examples of faulty real-world measurement. 80
5.5 Scheme of OBB pseudo measurement generation approach. 82
5.6 Example mappings of proposed pseudo measurement generation approach. 84
5.7 Examples of influences on the measurement model. 85
5.8 Example of two sensors s_1 and s_2 covering different FoVs and two tracked objects. 86
5.9 Example situations showing benefits of sensor-based parameter models. 87
5.10 Detection probability distance model for class "Car". 90
5.11 Models of detection probability for camera detector. 91
5.12 Comparison of detection probability model with mixture model. 92
5.13 Clutter density distance model for class "Car". 93
5.14 Models of clutter density for camera detector. 94
5.15 Example situation with track confirmation strategy. 105

6.1 Comparison of the sensor setups and FoV of the KITTI and the truck dataset. 107
6.2 IoU for a 2D bounding box in image coordinates. 110
6.3 Interpolated precision-recall curve. 111
6.4 Distance error of camera detections to GT from Lidar over the object distance. 113
6.5 Visualization of different base distances for 2D bounding boxes. 115
6.6 Comparison of cardinality estimation of GM-CPHD and GM-PHD filter. 122
6.7 Comparison of simple filter approaches on the KITTI dataset using objects of class "Car". . . . 123
6.8 Visualization of the evaluation areas. 126
6.9 Comparison of GM-PHD and KF using HOTA metric on the KITTI dataset. 127
6.10 Comparison of GM-PHD and KF using the HOTA submetrics on the KITTI dataset. 127
6.11 Comparison of GM-PHD and KF using the mean OSPA$^{(2)}$ score on the KITTI dataset. 128
6.12 Comparison of GM-PHD and KF using the OSPA$^{(2)}$ metric on the truck dataset. 129
6.13 Comparison of detection probability model impact on HOTA results of the KITTI dataset. . . . 129
6.15 Influence of combined parameter models on HOTA results of the KITTI dataset. 130
6.14 Comparison of clutter density model impact on HOTA results of the KITTI dataset. 130
6.16 Influence of distance-depending detection probability model on the truck dataset. 131
6.17 Sensor failure analysis with different sensors missing. 133
6.18 Comparison of OSPA$^{(2)}$ results in different ODDs of the truck dataset. 134
6.19 Comparison of different classification approaches on Fused Truck dataset with GM-PHD filter. 134

Zusammenfassung

Moderne Lkws sind mit einer Vielzahl von Fahrassistenzfunktionen ausgestattet, welche sich durch steigende Gesetzesanforderungen noch weiter erhöhen. Damit diese Systeme zuverlässige Reaktionen auf Verkehrssituationen erzeugen können, wird eine gute Umgebungserfassung benötigt. Moderne Fahrzeuge haben dafür verschiedene Sensoren, welche Informationen über die Objekte um das Ego-Fahrzeug herum erfassen, die anschließend fusioniert werden. In dieser Arbeit werden alle Schritte einer solchen Verarbeitungskette zur Detektion, Verfolgung und Fusion von Objekten erläutert und diskutiert.

Die Arbeit stellt einen Ansatz zur Objekterkennung mit Lidar Sensoren vor, welcher sowohl sicherheitskritische Aspekte als auch die Detektionsgüte beachtet. Dabei wird eine Kombination aus klassischer Punktwolkenverarbeitung mit neuronalen Netzen eingesetzt. Außerdem wird ein Ansatz zur Objektdetektion direkt vor dem Lkw mithilfe von Kameras entwickelt, wobei die hohe Position der Kamera an der Fahrerkabine im Fokus steht.

In State-of-the-Art Systemen werden detektierte Objekte häufig mithilfe eines Kalman Filters (KF) fusioniert. Solche Ansätze haben Vorteile durch eine geringe Komplexität und Laufzeit. Sie sind jedoch eine Erweiterung des Single-Target-Trackings und benötigen ein Datenassoziationsverfahren, welches die neuen Messungen den bereits verfolgten Objekten zuordnet, und ein Track-Management, welches das Auftauchen und Verschwinden von Objekten behandelt. Dabei können Fehler entstehen, welche die Zustands-, Existenz- und Klassifikationsschätzung negativ beeinflussen und somit ein falsches Verhalten der Fahrassistenzfunktionen hervorrufen können. Multi-Objekt-Bayes Filter schätzen hingegen sowohl die Zustände als auch die Anzahl der Zustände mithilfe von Random Finite Sets, wodurch direktes Multi-Objekt-Tracking ohne explizite Datenassoziation und Track-Management möglich wird. In den vergangenen Jahren wurden mehrere Approximationen wie der Probability Hypothesis Density Filter vorgestellt, welche aufgrund einer relativ geringen Laufzeit auch für die Anwendung in integrierten Steuergeräten infrage kommen.

Im Rahmen dieser Arbeit wird ein Framework zur Objektverfolgung und -fusion von Messungen verschiedener Sensoren vorgestellt, welches verschiedene Multi-Objekt Filter zur Zustandsschätzung verwenden kann. Hier werden sowohl der Kalman Filter als auch den Gaussian Mixture Probability Hypothesis Density (GM-PHD) Filter eingesetzt. Somit wird eine Plattform geschaffen, welche einen direkten Vergleich der Ansätze ermöglicht. Die Plattform ist dabei flexibel gestaltet, wodurch sie für eine Vielzahl verschiedener Sensortypen und Sensorpositionen eingesetzt werden kann. Um diese Flexibilität mit dem GM-PHD Filter zu erreichen, werden sensorbasierte Parametermodelle zur Darstellung der Detektionswahrscheinlichkeit und Clutter-Dichte entwickelt. Außerdem werden eine Methode zur Integration der Fusion der Klassifikation im GM-PHD Filter sowie ein Algorithmus zur Fusion verschiedener geometrischer Repräsentationen von Detektionen vorgestellt. Durch den Einsatz von Gating kann für beide Filter eine Laufzeit von unter einer Millisekunde pro Zeitschritt erreicht werden.

Das entwickelte Framework zur Objektverfolgung und -fusion ermöglicht einen direkten Vergleich des KF mit dem GM-PHD Filter auf Basis von verschiedenen Datensätzen. Durch die Evaluation der Filter mit dem KITTI Datensatz und einem neu entwickelten Lkw-Datensatz wird ein umfangreicher Vergleich mit verschiedenen Sensor-Setups und deren Teilmengen gezeigt. Hier zeigt die entwickelte Implementierung des GM-PHD Filters mit einem erreichten HOTA Ergebnis von 77,1 % einen höheren Wert als der Kalman Filter mit 73,6 % für fusionierte Daten des KITTI Datensatzes. Auch bei den meisten der anderen Versuche erreicht der GM-PHD Filter bessere Ergebnisse. Die Evaluation untersucht zudem den Einfluss der vorgestellten Optimierungen, wie der sensorbasierten Parametermodelle und des Gatings. Beim Lkw-Datensatz lässt sich somit eine Verbesserung des 1-OSPA$^{(2)}$ Ergebnis von 0,36 auf 0,4 für fusionierte Daten erreichen. Des Weiteren werden Versuche zum Einfluss eines ausgefallenen Sensors sowie ein Vergleich der verschiedenen Methoden der Klassifikationsfusion gezeigt.

Abstract

Modern trucks are equipped with various Advanced Driver Assistance Systems (ADAS) functions, which is increasing due to new legal requirements. Good environmental sensing is required in order for these systems to behave reliable. For this purpose, modern vehicles fuse the detections of different sensors, which measure the objects around the ego-vehicle. In this thesis, all steps of such a pipeline for detection, tracking, and fusion of objects are explained and discussed.

This thesis presents an approach to object detection with Lidar sensors that considers both safety-critical aspects and detection quality. A combination of classic point cloud processing with neural networks is used. In addition, an approach for object detection directly in front of the truck using a camera is developed, focusing on the high position of the camera on the driver's cab.

In state-of-the-art systems, detected objects are often fused using a Kalman Filter (KF). These approaches have the advantage of low complexity and runtime. However, they are an extension of single-target tracking and require a data association approach that assigns new measurements to the already tracked objects and a track management handling the appearing and disappearing of objects. This can introduce errors that can negatively affect the state, existence, and classification estimation, causing incorrect behavior of the ADAS functions. Multi-object Bayes filters, on the other hand, estimate both the states and the number of states using random finite sets, allowing direct multi-object tracking without explicit data association and track management. In recent years, several approximations, such as the Probability Hypothesis Density filter, have been presented which are also suitable for use in integrated control units, due to their relatively low runtime.

In this thesis, a framework for object tracking and fusion of measurements from different sensors is presented, which can use different multi-object filters for state estimation. Here, both the Kalman filter and the Gaussian Mixture Probability Hypothesis Density (GM-PHD) filter are applied. Thus, a framework is created which allows a direct comparison of the approaches. The framework is designed to be flexible, allowing the number and type of sensor used for environment perception to be easily changed. In order to achieve this flexibility with the GM-PHD filter, sensor-based parameter models are developed to represent the detection probability and clutter density. In addition, a method for integrating the fusion of the classification in the GM-PHD filter and an algorithm for fusing different geometric representations of detections are presented. By using gating, a runtime of less than one millisecond can be achieved for both filters used.

The developed framework for object tracking and fusion enables a direct comparison of the KF with the GM-PHD filter based on datasets. By evaluating the filters with the KITTI dataset and a newly developed truck dataset, a comprehensive comparison with different sensor setups and their subsets is shown. Here, the developed implementation of the GM-PHD filter with an achieved HOTA result of 77.1 % shows a higher performance than the Kalman filter with 73.6 % for fused data of the KITTI data set. The GM-PHD filter also performs better in most of the other tests. In addition, the evaluation shows the influence of the developed optimizations, such as the sensor-based parameter models and the gating. For the truck dataset, this results in an improvement of the 1-OSPA$^{(2)}$ score from 0.36 to 0.4 for fused data. Furthermore, experiments on the influence of a failed sensor and a comparison of the different methods of classification fusion are shown.

1 Introduction

In recent years, the automotive industry is undergoing a major change. Electronics and software define the purchasing criteria of today and tomorrow (Haas et al. 2020), which also includes Advanced Driver Assistance Systems (ADAS). This is also supported by the "zero-fatalities" strategy of the European Union (European Commission and Directorate-General for Mobility and Transport 2020; European Commission 2020) to achieve zero fatalities and serious injuries on European roads by 2050. This goal is supported by regulations like the General Safety Regulation (GSR) (European Parliament and Council 2019) and promoted in public projects such as TransSec[1].

Driver assistance systems can save lives, especially in the field of heavy commercial vehicles. This is exemplified by the Advanced Emergency Braking System (AEBS), although other systems are also effective. The AEBS for trucks, which has been declared mandatory by EU regulation since the end of 2015, is also proven to be very effective (Unger 2011). The effect of this system is shown by Petersen et al. (2018) in an analysis of truck accidents with serious personal injury on highways in Niedersachsen. The involvement of trucks with AEBS is under-proportional compared to the number of trucks equipped with such systems (Petersen et al. 2018). This suggests a reduction in accident frequency due to AEBS. Other systems, like the Blind Spot Information System (BSIS) have similar potentials in reducing the amount of critical accidents.

1.1 Advanced driver assistance systems

ADAS are active systems designed to improve vehicle safety and improve driving comfort by warning the driver or actively engaging the vehicle control. This encompasses emergency systems that react to hazardous situations by warnings or executing emergency maneuvers, as well as systems for partial driving automation, such as actively maintaining the current lane through steering.

For risk assessment and classification of automated driving and ADAS functions, the Society of Automotive Engineers (SAE) defined different levels of automation (On-Road Automated Driving (ORAD) committee 2021):

- Level 0: No Driving Automation - The driver is always in full control of the vehicle. Level 0 covers passive warning systems, that do not actively engage.
- Level 1: Driver Assistance - The system is in control of either longitudinal or lateral dynamics. The driver has to constantly monitor the system and be prepared to take over full vehicle control.
- Level 2: Partial Driving Automation - The system is in control of longitudinal and lateral dynamics for defined situations and for a limited amount of time. The driver has to constantly monitor the system and be prepared to take over full control any time.
- Level 3: Conditional Driving Automation - The system is in control of longitudinal and lateral dynamics for defined situations and for a limited amount of time. In contrast to level 2, the driver does not have to constantly monitor the system, but needs to prepared to take over control as a fallback solution.
- Level 4: High Driving Automation - The system is in control of longitudinal and lateral dynamics for defined situations and for a limited amount of time. If the driver does not take over control in case of a failure, the system automatically maneuvers to a state of minimum risk.
- Level 5: Full Driving Automation - The system completely takes over control in all situations and no interaction by the driver is necessary.

[1] http://www.transsec.eu/

Currently, the majority of ADAS are classified in levels one to three. Although a detailed environment perception is necessary for all levels, this thesis focuses on the application of perception for ADAS systems of level 2.

Haas et al. (2020) show an overview of the currently available ADAS functionalities. The following list gives a subset of available features for trucks where an advanced environment perception is beneficial:

- Advanced Cruise Control (ACC): Automatic speed control, while keeping a minimum distance to the preceding vehicle. Mainly controlled by a Long Range Radar (LRR).
- Lane departure warning and lane keeping assist: Warns the driver in case of lane departure or actively steers to keep the lane. Mainly controlled by front-facing camera.
- Advanced Emergency Braking System (AEBS): In the event of an imminent collision, a warning is first generated and then emergency braking is carried out. Controlled by LRR and front-facing camera.
- Blind Spot Information System (BSIS) (or side guard assist): Monitors the side area of the truck (especially blind spots) and warns of collisions with road users. Controlled by Short Range Radar (SRR) or ultrasonic sensors in the side area.
- Moving Off Information System (MOIS): Monitors the front area of the truck, especially the direct blind spot in front of the front bumper, and warns of collision with weak targets. Can be controlled by SRRs, ultrasonic sensors or cameras.

Previous ADAS are mostly based on single sensors and designed to work in specific and simple situations. For example, the early emergency braking assistants in many vehicles were usually based on radar detections and limited to warning and partial braking. By combining more sensor domains and increasing the number of sensors, the perception quality improves. This fused perception is able to cover more complex situations, which can increase the ADAS capabilities to carry out advanced maneuvers. This implies the need for modern detection sensors and modern multi-sensor multi-object tracking methods to fuse the detections. Such approaches can provide a stable full surround perception, which is key to develop new level 2 ADAS function and to enable level 4 and level 5 automated systems.

1.2 Challenges of multi-sensor multi-object detection and tracking for trucks

In general, many principles and methods for multi-sensor multi-object tracking are similar for passenger cars and trucks. However, there are some differences that need to be considered when developing perception approaches for trucks.

The variety of chassis variants, that can share parts of the electronic components, is enormous, which can result in differences in the sensor setup. Figure 1.1 shows examples of different truck types that have different mechanical platforms, and within each of the models shown there is a wide variety of bodies and configurations. On the one hand, the installation positions of the sensors can be different, as can the number and types of perception sensors. There are different types of cabins available, that have the forward facing camera installed at different heights, which can influence the detection capabilities for high distances, as a lower mounted camera may not be able to look over preceding cars. Similarly, radar sensors might be attached at different positions in the chassis, creating differences in the overlap of the Field of View (FoV). The aim is to be able to handle the highest possible variance of sensor setups with the same perception framework.

Due to the high mileage and long distances covered by trucks, the sensor set is often prone to wear and damage. In addition, the sensors can easily be damaged during maneuvering. The approaches should therefore be able

Figure 1.1: Different types of tucks that can share parts of the electronic components (Daimler Truck 2023).

to cope as well as possible with suddenly changing sensor sets. Tracking and fusion should continue to work even if a sensor fails.

Another important point is the estimation of the class of all surrounding objects. Depending on the class, the ADAS functions react differently: Particular attention is paid to Vulnerable Road User (VRU)s, while infrastructure should not be reacted to. This is particularly critical in the case of trucks, as false positive warnings and interventions are often triggered for incorrectly classified infrastructure objects due to the large ego dimensions and trailing curves. On the other hand, incorrect classification of VRUs must be avoided to be able to trigger alerts in all relevant situations. Therefore, estimating the class using a fusion of all available information is critical for the functions.

There are additional differences in the area of object detection, which are mainly linked to the mechanical properties of trucks. Most types of truck have a separately suspended cab. Cameras are usually located behind the windshield in the cab area so that they can be cleaned by the windshield wiper. Compared to cars, they are therefore mounted much higher and are also subject to greater movement. Figure 1.2 shows the separately sprung area for a European truck, with typical mounting positions for a camera and radar sensor. As shown, radar sensors are mounted at a lower position on the chassis.

In addition, the ADAS functions itself need to consider other masses, dimensions and dynamic behaviors compared to cars. In particular, the large variety of the mentioned parameters needs to be considered, since the same systems have to work for single tractors as well as truck-trailer combinations. However, the functions themselves are not discussed in this thesis.

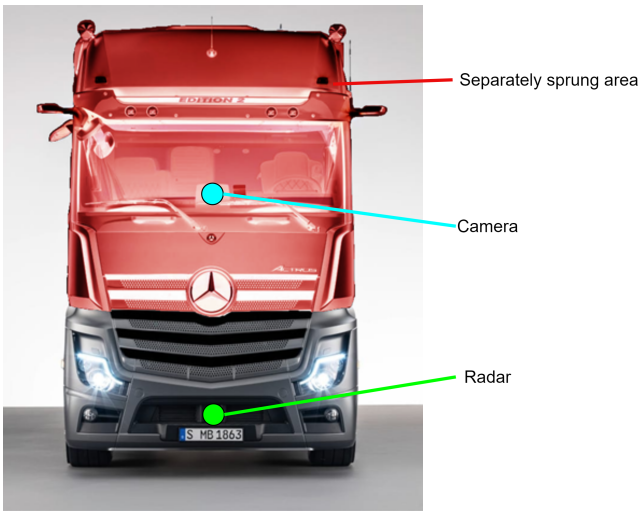


Figure 1.2: Red area marks separately sprung cabin are with camera sensor mounted behind windshield. The radar is mounted on the chassis.

1.3 Thesis outline

The overall goal of the thesis is the development of a general object-based multi-sensor multi-object tracking and fusion framework, that is universally applicable and is not tailored to specific functions. The framework is not limited to a specific set of sensors (e.g. series radars), but is designed to support all types of perception sensors with object interface, regardless of the sensing domain. This means that all combinations of series sensors and modern sensors, like new cameras or Lidar sensors, can be applied to the framework. In order to be able to use such new sensors, detection algorithms for Lidar sensors and camera sensors in new positions are developed.

Various filter algorithms can be used within such a framework. An important goal of this work is the application of the Gaussian Mixture Probability Hypothesis Density (GM-PHD) filter from the Random Finite Set (RFS) filter family in the developed framework and a comparison with the commonly used Kalman Filter (KF). As both filters are supported, the framework is unique in offering a direct comparison of both methods with the same boundary conditions and parameters, allowing recommendations to be made on their use. An extensive comparison provides a good decision basis for the usage of future filter algorithms.

In addition, adjustments and optimizations of the GM-PHD filter for automotive applications are developed, since most existing work does not focus on a generic application of this filter for sensor fusion in real-world automotive systems, but typically on specific setups or scenarios. All developed approaches are evaluated using both public datasets and a truck dataset to make general, as well as truck-specific statements about performance.

Throughout the thesis, there are some main considerations for the usability of all contributions in real-world trucks. These are used for evaluation of existing approaches and are boundary conditions for the development of new systems. In particular, these boundary conditions meet the specific needs for truck that were described

in the previous chapter. They are a differentiator to other works, since these boundary conditions enable a fast integration to widespread real-world applications.

- Real-time capability is mandatory to be able to run in real-world trucks.
- The framework should support different sensor numbers, types and mounting positions to be invariant of chassis and vehicle variants.
- Compatibility with current sensor systems using object-based interfaces should be given. However, different types of geometric object representations should be supported at the same time to allow combinations of sensors with different capabilities.
- The developments should be universally applicable and work in different Operational Design Domains (ODD), such as highways, rural and urban areas. Therefore, the dataset should also include these domains.

These basic considerations are applied to all developments of the perception pipeline covered by this thesis. Since an object-based framework is being developed, the individual chapters of the thesis contain individual processing steps of such a pipeline, which are shown in the simplified overview of Figure 1.3. First, the basics that are necessary for all further chapters are explained. Chapter 2 therefore explains the basics of common data representations, common perception sensors and an object-based perception pipeline. Next, chapter 3 discusses different object detection approaches and describes the developed detection methods for Lidar and camera sensors. The next processing step is the multi-sensor multi-object tracking, so chapter 4 describes the basics of multi-object-tracking with the KF and RFS filters, as well as the basics and an overview of multi-sensor fusion. Chapter 5 describes the developed multi-sensor multi-object tracking and fusion framework, as well as the adaptations to the GM-PHD filter for automotive applications. The developed detection approaches and the multi-sensor multi-object tracking framework are then evaluated in chapter 6, that also shows an overview of the used datasets, including the newly created truck dataset.

Several parts and methods, in particular the KF-based developments, are based on the approaches developed in the preceding master thesis "System for Detection, Tracking and Classification of multiple Objects for Collision Detection based on Lidar Sensors" (Bader 2019).

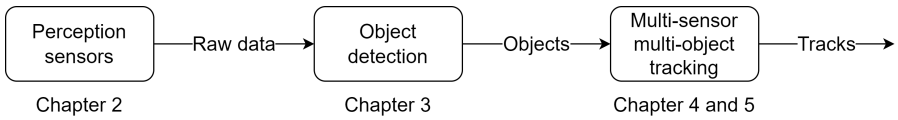


Figure 1.3: Simplified scheme of perception processing steps assigned to the chapters of this thesis.

1.4 Contributions

The following list contains the major contributions achieved within the thesis, arranged by the corresponding chapter. The detailed contributions are shown in the respective chapters, while the chapters 2 and 4 only provide theoretical background.

- Chapter 3:
  - A comparison of state-of-the-art detection approaches for Lidar and camera sensors is shown.
  - A new two-stage approach for detection and classification using Lidar sensors is proposed. In contrast to existing approaches, it combines traditional approaches with deep learning to achieve a state-of-the-art performance while still providing detections in untrained edge cases.
  - The development, application, and evaluation of a new camera detection pipeline for downward looking cameras mounted on truck cabins is proposed. Existing detection approaches are combined with distance estimation on a new position and perspective to provide 3D bounding boxes in close range.
- Chapter 5:
  - The proposal of a multi-sensor multi-object framework capable of working with multiple Multi-Object Tracking (MOT) filters, including the KF and GM-PHD filters, which is a unique differentiator and the prerequisite for a comprehensive comparison.
  - A new spatial measurement matching approach to handle different geometric types of measurements, like Oriented Bounding Box (OBB) or L-shapes, in a single framework is proposed. This unique approach allows to combine sensors of different capabilities in a general way and is not depending on certain type of sensor output.
  - Development of new GM-PHD filter adaptations for real-world automotive applications. These include sensor-based parameter models for the detection probability and clutter density, that allow a proper fusion of sensors with partially overlapping FoVs and at high distances and generally improve the tracking quality. The addition of a classification tag is developed, allowing an integrated propagation of the class with the GM-PHD filter. A gating process is proposed to improve the computation speed and a track confirmation strategy is developed to improve the robustness against multiple MOT errors, like ID switches during occlusion. All developments are optimized for the usage in automotive applications and are subject to the boundary conditions stated in the previous section.
  - The proposal of multiple classification fusion schemes, which can be integrated into the framework. This allows a comparison on real-world data. The proposed integration of the classification to the GM-PHD filter provides a unique possibility to implicitly propagate this property within the filter.
  - The implementation of a KF-based MOT approach for comparison to the GM-PHD approach.

- Chapter 6:
  - Evaluation of the two-stage Lidar detection approach on the KITTI detection dataset.
  - Evaluation of the camera detection approach for downward looking cameras mounted on truck cabins. The evaluation is based on a truck data set and is carried out for both detection and distance estimation.
  - The creation of a new truck dataset with sensors close to series for evaluation of the developed detection and tracking algorithms is proposed. The annotation of the three scenes is based on Lidar data and follows a similar structure like the KITTI dataset.
  - An empirical comparison of the KF and several RFS-based multi-object filter approaches using the Optimal Sub-Pattern Assignment (OSPA) metric with point objects on the KITTI dataset compares the potential of different RFS filters. This comparison only evaluates the filter performance without track continuity.
  - A comparison of the GM-PHD filter to the KF is carried out for both the KITTI and the truck dataset comparing the MOT performance for different sensor combinations.
  - The influence of the developed sensor-based parameter models for the GM-PHD filter are evaluated on both the KITTI and the truck dataset to show the potential.
  - The runtime differences for the GM-PHD gating and the proposed detection probability model are evaluated and compared to the KF and a mixture implementation for the detection probability model.
  - The developed classification fusion schemes are compared to each other on the truck dataset with the GM-PHD filter.

2 Environment perception basics and sensors

The environment of a truck can be measured by a variety of sensors with varying strengths and weaknesses. These measurements are then represented by one of several possible models that have different advantages and disadvantages, as well as different levels of abstraction. Depending on the model, they are also suitable for other types of sensors and data fusion. This chapter first provides an introduction to the representations used throughout the thesis and then presents the basics of all types of sensors used. First, however, an overview of the proposed perception pipeline is given to allow a better understanding of the relationships between the described sensors, detection, tracking and fusion approaches.

2.1 Overview of proposed perception pipeline

The overall goal of the perception pipeline shown in Figure 2.1, is to create robust tracks that can be used by ADAS functions such as the ones discussed in the previous chapter. In order to provide these tracks, a multi-sensor multi-object tracking approach is used to fuse the individual data from different sensors and track the objects over time.

Previous generations of ADAS functions often relied on single sensors, like automotive radars, without a central tracking and fusion stage, so many current generation sensors already have integrated object detection functionalities, that sometimes also cover tracking. These sensors have an object interface, but the provided object attributes may vary between different sensors, which must be taken into account.

On the other hand, modern sensors like Lidars often provide raw data, which are sensor measurements (e.g. point clouds) without further processing, like integrated detection or tracking. Therefore, object detector modules have to be applied to all sensors that deliver raw data, which are in particular raw-data cameras and Lidars. This ensures compatibility with object-based perception sensors, while raw data sensors can be included through the use of downstream object detectors.

The modules, which are marked in gray in Figure 2.1 are developed and described in detail in this thesis. In particular, object detector modules for camera and Lidar are proposed in chapter 3, while approaches for multi-sensor multi-object tracking are proposed and discussed in chapters 4 and 5. This chapter describes the basics of environment representation and the individual sensors to give an overview of their capabilities when used as input for the developed approaches in this thesis.

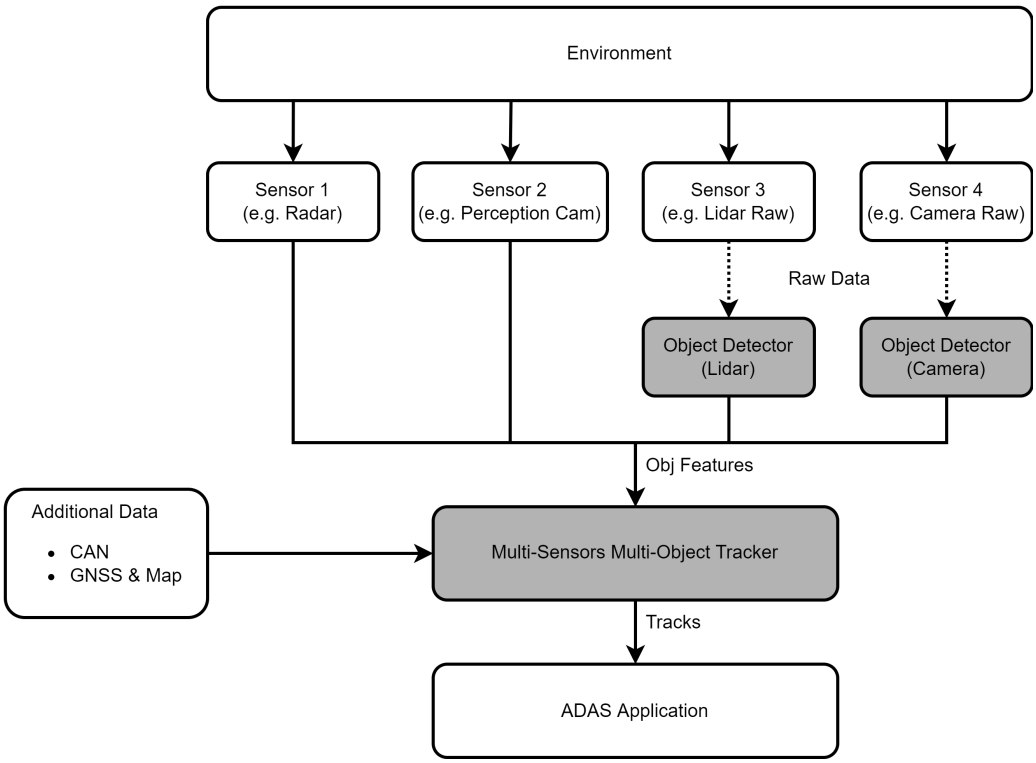


Figure 2.1: General pipeline used for this thesis. The environment is captured by sensors, which either directly process the data to object features or provide their raw output (e.g. image or point cloud) to detection algorithms. Next, the tracking and fusion is performed. Dark gray modules are developed in this thesis.

2.2 Environment representations

Generally, it can be said that low abstraction levels of the environment offer a higher level of detail, but also require more resources. Higher abstraction levels, on the other hand, make more assumptions and thus reduce the required resources (Schreier 2018). Therefore, a suitable compromise must be found for respective application. It must also be taken into account that some environment models are only suitable for certain sensor modalities. Schreier (2018) gives a good overview of the typical environment representations like interval maps (Weiherer et al. 2012), elevation maps (Herbert et al. 1989), Stixel worlds (Pfeiffer and Franke 2011) and more. Map data, especially high definition maps can also be considered as a static environment model. However, since the focus of this thesis is more on the perception of objects and road users surrounding the ego vehicle, only the occupancy grid map and the object map are described in more detail. Both representations are very different in terms of abstraction level and capabilities.

2.2.1 Occupancy grid

A common low-level representation is the occupancy grid, where the environment is represented by a discrete 2D grid and the probability of occupancy is assigned to each cell. This type of map was introduced by Elfes (1989) and was used in various applications in recent time. Thrun et al. (2005) give a detailed introduction to this topic. An occupancy grid is suitable for low-level data fusion since the raw data from all sensors with depth information (e.g. Lidar, radar, stereo-camera) can be used for an update. Another advantage is the explicit representation of free-space information. However, compared to other representations, an occupancy

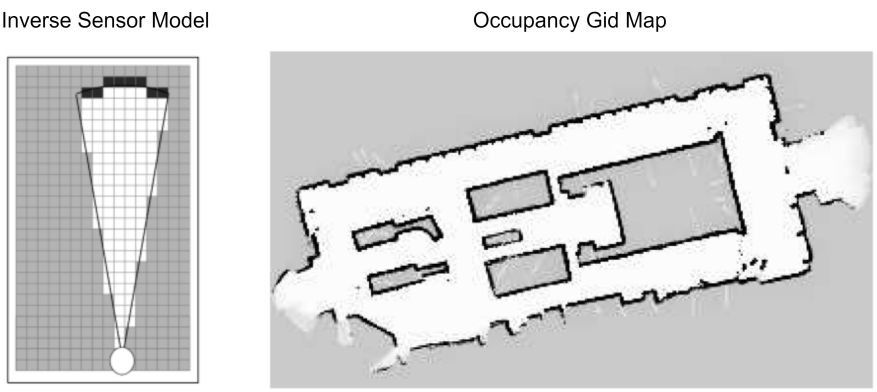


Figure 2.2: Example of occupancy grid and inverse sensor model (Thrun et al. 2005).

grid has high memory and bandwidth requirements (Grewe et al. 2012). Additionally, a standard occupancy grid is not capable of handling dynamic situations. This can only be done with hybrid models, where a list of dynamic objects is modeled in parallel to the static occupancy grid map (Grewe et al. 2012), or by creating a dynamic occupancy grid map (Nuss et al. 2018). For both options, a significantly higher computational effort is necessary. Another drawback is the occurrence of discretization effects (Schreier 2018), which can decrease the accuracy in important areas and the modelling of details in unimportant areas, which leads to inefficient processing. However, Grewe et al. (2012) argues that future ADAS systems will need representations like occupancy grid maps, that also model free-space to work in complex scenarios.

The goal of an occupancy grid map is the calculation of the occupancy of each cell over multiple time-steps from noisy measurement data using an inverse sensor model. Figure 2.2 shows both a map and an inverse sensor model, where the darkness represents the probability of occlusion. Each cell is updated in every time-step using a binary Bayes filter (Thrun et al. 2005). The update can also be solved with the Dempster-Shafer theory (Grewe et al. 2012) to model multiple conflicting sensors.

2.2.2 Object map

The object map representation consists of a list of spatially distributed geometric objects and corresponds to the group of feature maps in the overview of Schreier (2018). Such object-based representations have been used for many years in ADAS functions (Grewe et al. 2012), like the ones mentioned in chapter 1.1. A drawback of this approach is the sparse environment model, which only models the space covered by the features (objects) without additional information, like free-space or similar. On the other hand, they usually require low amounts of memory and bandwidth and can represent the environment in a continuous probabilistic way with uncertainties (Schreier 2018). Therefore, they can be more accurate if the object representation accurately describes the actual objects (Lakemeyer 2003) and the dynamics can be easily represented by assigning a motion state to each object. Additionally, for most of the sensors, like radar, camera or Lidar, there are methods for the generation of objects with similar state representation which allows the feature-level fusion of those components. Many current multi-object tracking pipelines work with object maps, too. Due to the lower complexity and focus on specific parts of the environment, this representation is suitable for ADAS functions that focus on a specific task, which is typically the case for functions up to SAE level two.

2.3 Object representations

In object map representations, all objects of the object list have a geometric description. Depending on the application, sensor capabilities and used algorithms, different types of geometric object descriptions may be used, which mainly differentiate in the degree of spatial details and the level of abstraction. In contrast to the proposed methods here, the shape estimation can also be directly integrated to the tracker using Bayesian models, which is known as extended object tracking (Granström et al. 2017) and does not depend on sensor-level shape estimation. The following list gives an overview of possible object representations. Here, the differences are discussed with a focus on spatial properties and shape and visualized in Figure 2.3. Note, that the dynamics of an object, like velocity and acceleration, are described by the motion model that is independent of the geometric description and discussed in chapter 5.3.

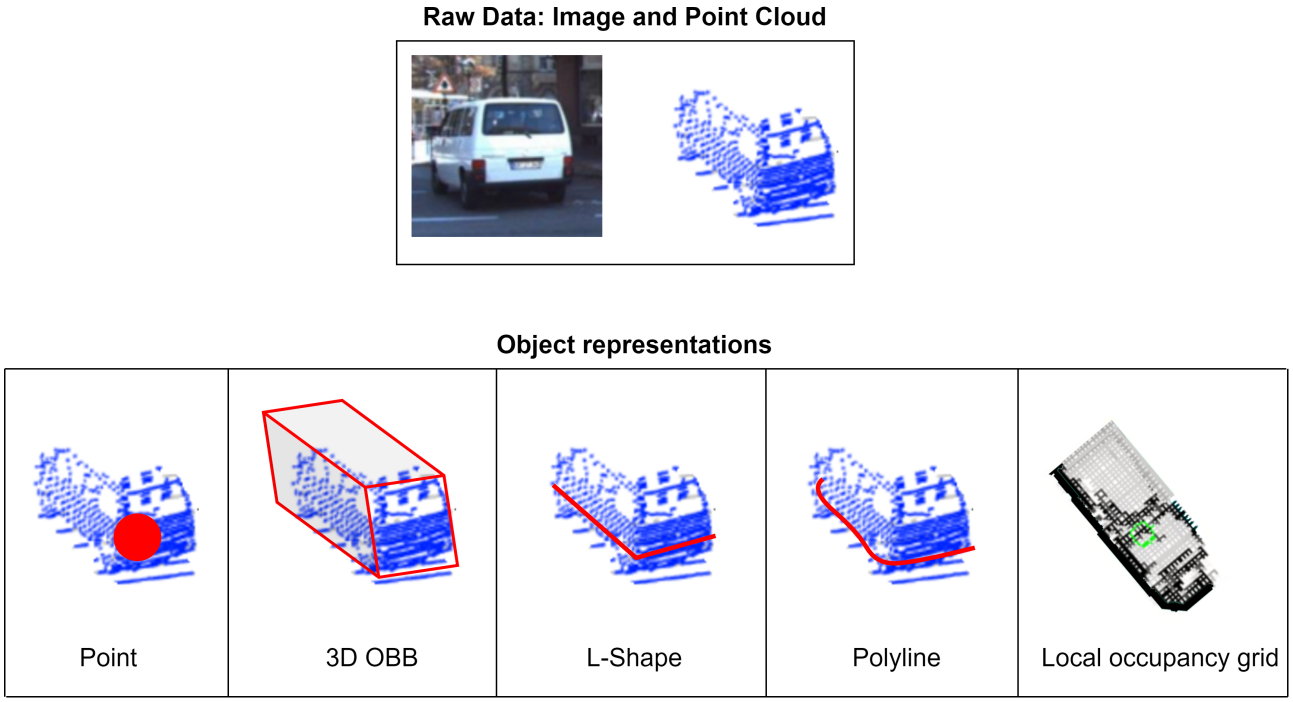


Figure 2.3: Examples of different object representations.

- Point: The point model represents each object as a point in space without spatial expansion. This results in a very compact and simple solution, that does not require complex matching algorithms. However, the geometry of the objects is not modeled.
- Basic geometric shapes: This group of models describes each object as a basic shape. The advantage is a compact geometric representation with few parameters, which includes enough geometric information for most level two ADAS functions.
  - OBB: The oriented bounding box consists of the object's position, the dimensions, and the orientation as a rectangular box. It is frequently used in public datasets, like KITTI (Geiger et al. 2012) or nuScenes (Caesar et al. 2019), since it can represent most road users with adequate accuracy. In addition, it is relatively compact, especially when the orientation is reduced to the yaw-Angle.
  - L-Shape: The L-Shape representation models the sensor-facing geometry of an object using two connected lines in Bird's Eye View (BEV). Therefore, it has some similarities to the OBB in practice, since the L-shape often consists of a close to orthogonal angle. The advantage is that it can

also represent other angles, but the drawback is that it only represents a portion of the outline of an object in 2D and not its volume.

- Polyline: A polyline representation (Kraemer et al. 2018) can represent the outline of an object more precisely and in a more general way compared to basic geometric shapes. Depending on the type of polyline, the amount of data required is relatively small, despite the high accuracy possible due to the parametric formulation. The state estimation, in particular the yaw rate estimation, can also improve compared to an OBB model (Kraemer et al. 2018). However, this type usually requires complex algorithms for matching and updating the shape over time. In addition, those algorithms may fail in complex scenarios when measurements within the object occur (Schütz et al. 2014), like a Lidar measurement through a window.
- Local Occupancy Grid: The idea of the occupancy grid map can be applied to the object level to deliver precise and robust shape estimation and the integration of local free space (Schütz et al. 2014; Quehl et al. 2019), where each object has a separate small occupancy grid. This can result in very accurate spatial representation at the cost of increase memory and processing consumption. Similar to the polyline model, a local occupancy grid-based approach can lead to better state estimations while tracking, as shown by Schütz et al. (2014).

The different representations are used depending on the required spatial accuracy, available bandwidth and computing power. In addition, the accuracy of spatial dimensions is limited by some sensing principles. For example, it is more difficult to measure accurate object geometries with cameras than with high-resolution Lidar sensors, which can create local occupancy grids. Local occupancy grid maps have high spatial accuracy, which may be required for tight maneuvering situations, but require a lot of bandwidth and computing power for tracking. Point objects, on the other hand, have no spatial extent, but are sufficient for some applications, such as ACC.

For other ADAS, like collision prevention on crossing or nearby targets, the object dimensions are important. These are described by the OBB for 3D objects, which is also the most common object representation in literature. The rectangular shape can be applied for most road users and is a good compromise between compute power and accurate environment representation. Therefore, it is also mainly used for the approaches in this thesis.

Multi-sensor systems operating on object maps usually use the same representation for each of its sensors for simplicity. However, this is not always given. If sensors with different representations are used in the same system, algorithms capable of matching and updating objects from multiple types of representations are required. This is addressed in chapter 5.4, where such an algorithm is proposed.

2.4 Perception sensors

The foundation to unlock highest potential for ADAS and Autonomous Driving (AD) is a good environment perception with high performance sensors. According to Marti et al. (2019), the key challenges for environment perception of AD and ADAS systems are:

- Complex and dynamic environment. This accounts especially for dense traffic and populated cities.
- Reliability under different conditions is necessary. This includes weather, light conditions, operation domains.

Currently, the most used sensors for ADAS are cameras and radars, with Lidar sensors emerging for higher level automation systems. All these sensors have strengths and weaknesses in different areas due to the application of different measurement principles. By combining the sensors in the form of sensor fusion, the weaknesses can be compensated, and the strengths combined, resulting in a better overall perception.

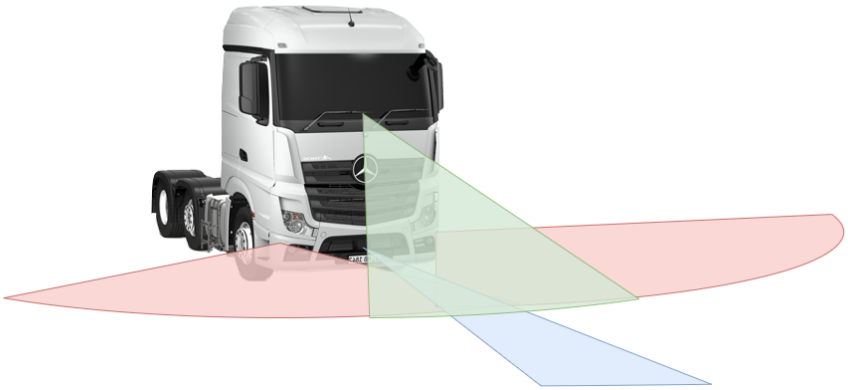


Figure 2.4: Example for truck sensor setup with SRR in red, LRR in blue and camera in green.

Current trucks already have several perception sensors, like a front facing camera and a LRR, as well as SRR sensors, which is exemplary shown in Figure 2.4. Each of these sensors individually detects objects surrounding the ego vehicle. The sensing principles of these sensors are described in the following sections. The sensors cover different areas of the electromagnetic spectrum and thus create redundancy, as shown in Figure 2.5.

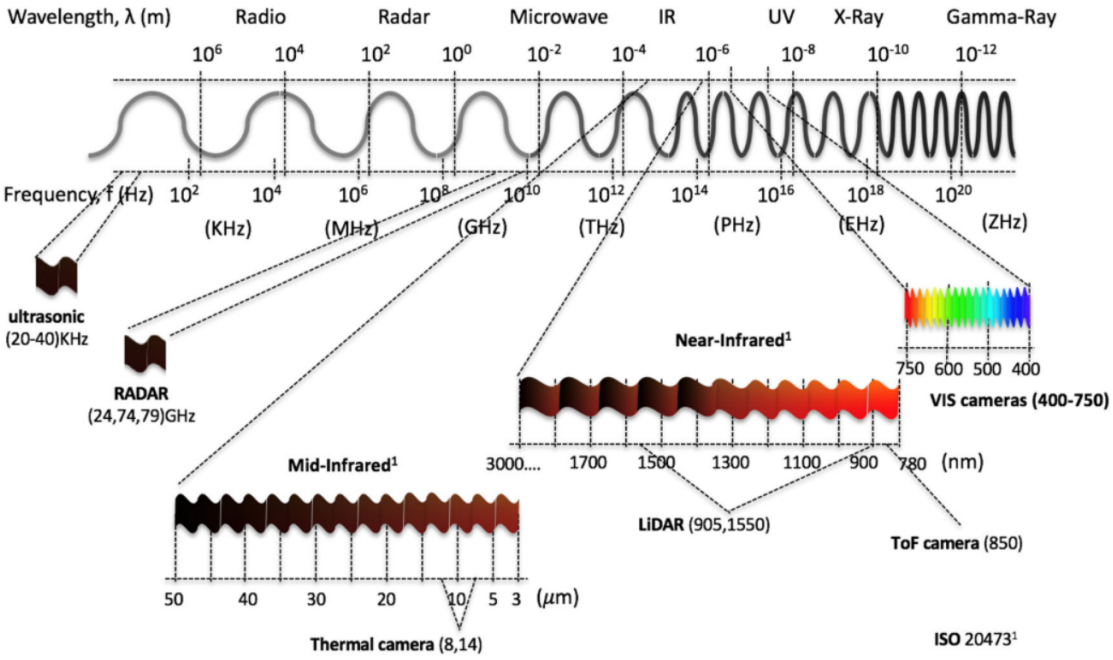


Figure 2.5: Magnetic spectrum including the ranges of different sensing principles, like camera, Lidar and radar (Rosique et al. 2019).

2.4.1 Lidar sensors

Light detection and ranging (Lidar) sensors are a fast-growing area of interest for ADAS and AD applications, since many successful participants of the DARPA grand challenge introduced these systems (Thrun et al. 2007). These sensors provide high precision 3D representations of the surrounding in the form of point clouds. A Lidar can therefore generate dense data of high spatial accuracy. Since it uses active illumination, the sensor is capable

of working in any light conditions and is less affected by weather than other optical sensor systems. Due to this combination of properties, Lidar sensors are stated to be a key technology for the future of AD (Roriz et al. 2021).

The basic working principle of Lidar sensors is to send out light signals and measuring the Time of Flight (ToF) to receive a reflection, which allows calculating the distance, as shown in Figure 2.6. Automotive Lidar sensors typically use wavelengths of 905 nm or 1550 nm. The transmission power, that also limits the maximum detection range, is limited by eye safety regulations, which allows current sensors to reach up to 500 m. This also depends on the signal type used for the detection. The two principles mainly used in automotive sensors are the pulsed ToF and the Frequency Modulated Continuous Wave (FMCW), while others like the Amplitude Modulated Continuous Wave (AMCW) are used less. The ToF and the FMCW principles are shown in Figure 2.7. ToF sensors emit short pulses of light, that are reflected by objects and then received by sensor, which allows the calculation of the distance. FMCW sensors, in contrast, continously emit light with the optical frequency modulated in a defined pattern over time. Doppler analysis of the return signals allows the calculation of the target's relative velocity in addition to the distance. Due to the velocity measurement, the higher robustness against interference and the higher possible signal-to-noise ratios, the latest developments tend to favor the FMCW Lidar sensors.

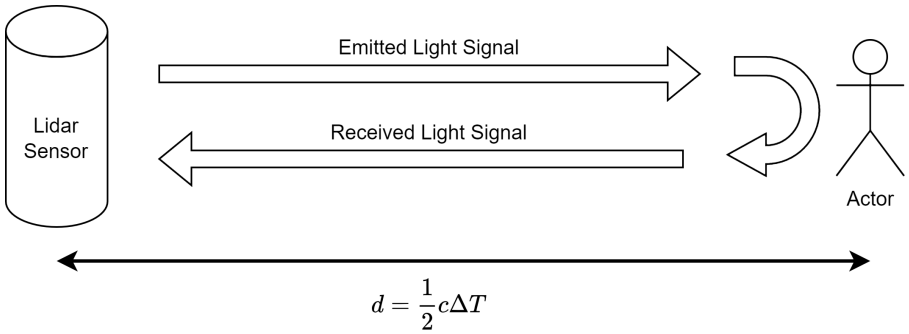


Figure 2.6: Basic working principle of a Lidar sensor. c is the speed of light in air, ΔT is the ToF of the light signal.

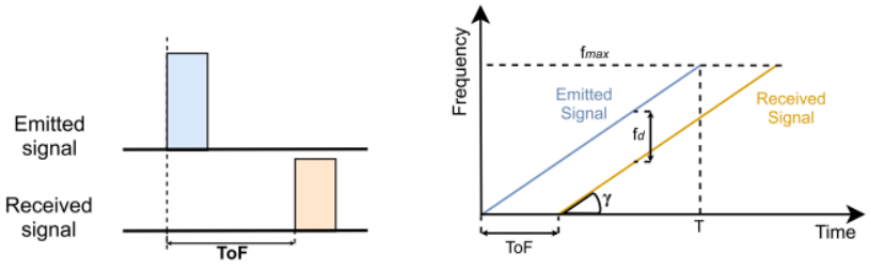


Figure 2.7: Examples of different measurement types of Lidar sensors by showing the emitted and received signals of ToF (left side) and FMCW (right side) Lidar sensors. For FMCW, the emitted and received frequency is shown which is used to calculate the target's velocity, composed from (Roriz et al. 2021).

In order to create a 3D point cloud, an image generating system is necessary to steer the laser beams in order to capture the distances of multiple directions. This is either done by a mechanical or a solid state system, as shown in Figure 2.8. In mechanical systems, either the sensor itself rotates on a platform or a spinning mirror is used to scan an area, which achieves a FoV of up to 360°. Solid state systems do not have mechanical parts and either use beam steering based on Micro-Electromechanical Systems (MEMS) controlled mirrors, optical

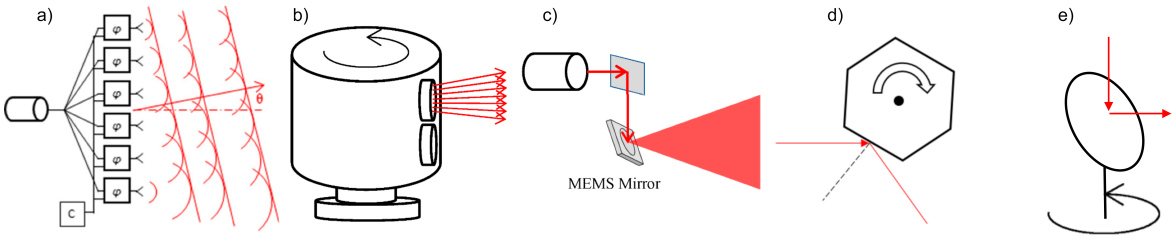


Figure 2.8: Comparison of different Lidar scanning principles: a) Optical Phase Array, b) rotating sensor platform, c) MEMS Mirror beam steering, d) rotating polygon mirror, e) rotating flat mirror. The principles shown in a) and c) are considered solid state, composed from Raj et al. (2020) and Wang et al. (2020).

phased arrays or use flash systems without any moving parts. Raj et al. (2020) show a more detailed overview of scanning mechanisms, while Roriz et al. (2021) give a general overview of Lidar technology for automotive applications.

The sensors used in this thesis use pulsed ToF measurements in combination with a mechanical spinning sensor platform and therefore have 360° horizontal FoV with vertical FoV of $-25\,\mathrm{deg}$ to $+15\,\mathrm{deg}$. Table 2.1 gives an overview of the capabilities of some state-of-the-art Lidar sensors. As shown, there is a large deviation in FoV, maximum range and accuracy depending on the used technology and application. In addition, the sensors vary greatly in price.

Table 2.1: Characteristics of Blickfeld Cube 1, Velodyne VLP-32c, Aeva Aeris II and Velodyne Alpha Puck.

	Blickfeld Cube 1	Velodyne VLP-32c	Aeva Aeris II	Velodyne Alpha Puck
Technology	ToF MEMS	ToF rotating	FMCW	ToF rotating
Range	< 75 m	< 200 m	< 500 m	< 300 m
HFoV	70°	360°	-	360°
VFoV	30°	40°	-	40°
Distance Accuracy	-	0.03 m	-	0.03 m
Application	Short-Range	Midrange	Long-Range	Long-Range

2.4.2 Cameras and artificial vision sensors

Cameras are a well established technology for ADAS, since they are able to sense all the traffic signs and lane markings that are originally designed for humans. They are today available in a huge variety of configurations in terms of resolution, frame rate, bandwidth, dynamic range, FoV and technology (Marti et al. 2019). As a result, cameras can be used for a wide range of tasks, from simple rear-view cameras to interior driver monitoring systems (Punke et al. 2016) and complex perception functions. One major advantage compared to other sensors is the low cost for most configurations.

Cameras are usually passive sensors that capture a 2-dimensional image of the world within the FoV. Most of them work in the visible spectral range and therefore can achieve similar detection capabilities like the human eye (Punke et al. 2016). The main challenges are varying light and weather conditions, as well as the data processing for 3-dimensional reconstruction. Some challenges of difficult conditions can be addressed by taking the near and far infrared spectrum into account, as shown by Pinchon et al. (2019). The 3D reconstruction capability can be improved by using stereo cameras (Hamzah and Ibrahim 2016). The quality of vision-based environment perception also massively increased in the last years due to the latest improvements in deep learning. This is also supported by the higher availability of computational performance.

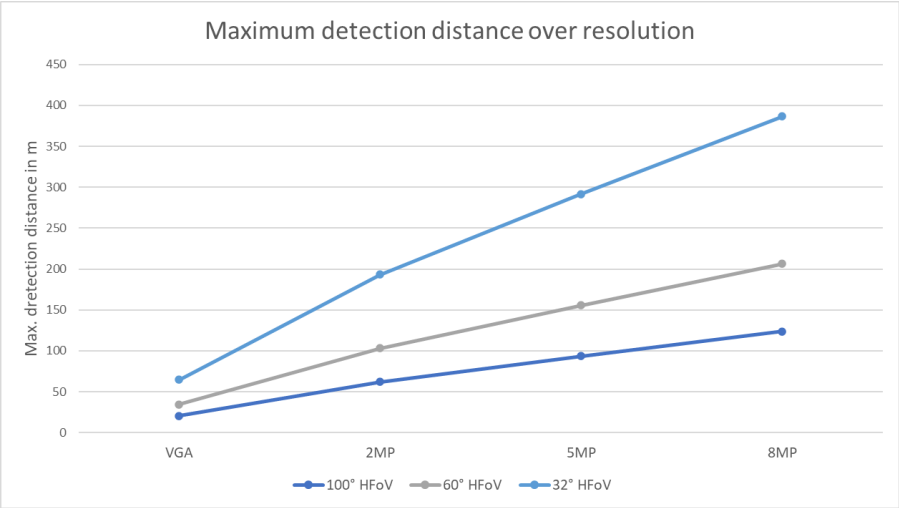


Figure 2.9: Maximum detection distance over camera resolutions of different horizontal field of views.

Automotive cameras with object detection are available in a huge variety of resolution and lense configurations. These two parameters are also the main variables influencing the maximum detection distance, since typical detection algorithms need a minimum amount of pixels on a target to be detected. Assuming a minimum height of 32 pixels for a 1.8 m pedestrian to be detected, Figure 2.9 shows the maximum detection distance for different lense configurations and resolutions. However, the minimum amount of pixels varies from algorithm to algorithm and also depends on the image quality in the given scenario. The accuracy of the distance estimation relies to a great amount on the used algorithm, but is typically in the range of < 10 % of the object's distance.

2.4.3 Radar sensors

Radar is derived from radio detection and ranging and describes a sensing technology based on the usage of electromagnetic waves in the radio frequency range. This type of perception sensor has been used for ADAS systems since 1998, where it was introduced for ACC (Winner 2016). Current automotive sensors use frequencies in the range of 24-100 GHz, with a tendency to use higher ones in future (Gamba 2020). The sensing principle has similarities to that of the FMCW Lidar sensor. Frequency modulated electromagnetic waves are sent out, reflected by targets and received. This enables radar sensors to detect targets like cars in distances > 250 m. Depending on the waveform type, the distance and receive angle, the relative velocity of reflections can be calculated.

Various interfering factors must be taken into account for radar measurements: The reflectivity of different targets varies greatly, with pedestrian typically having a low one compared to vehicles. However, this can also vary between different vehicles due to geometry: The reflections of flat vehicles or those with concave surfaces are harder to be detected (Winner 2016). Some weather conditions, such as heavy rain, can also negatively affect the signal-to-noise ratio and thus reduce the detection capabilities (Winner 2016). Another interfering factor are multipath reflections, where a target is reached by multiple radar waves on different reflection paths, thus generating multiple detections.

Following Patole et al. (2017) and Hasch et al. (2012), radar sensors can be divided by their maximum range into the categories of Short Range Radar (SRR), Medium Range Radar (MRR) and Long Range Radar (LRR), which are designed to support different ADAS functions. The radar sensors used for this thesis have integrated object detection functionalities. Therefore, they are treated as functional systems, without discussing the details of radar object detection. However, Gamba (2020) and Winner (2016) are referred for a detailed introduction to automotive radar technology.

Table 2.2: Characteristics of SRR, MRR and LRR, composed from Patole et al. (2017) and Hasch et al. (2012).

	SRR	MRR	LRR
Range	0.15-30 m	1-100 m	< 10-250 m
Vehicle speed	< 150 km/h	< 250 km/h	
FoV	$< \pm 80°$	$< \pm 40°$	$< \pm 15°$
Distance Accuracy	0.1 m	0.1 m	0.02 m
Velocity Accuracy	0.1 m/s	0.1 m/s	0.1 m/s
Applications	Parking Assistant Impact Warning VRU Detection	Blind Spot Detection Cross Traffic Assistant Side Impact Warning VRU Detection	Automatic Cruise Control Forward Collision Warning

2.4.4 Comparison

Table 2.3 shows an overview of the capabilities of each of the discussed sensor types. Each of them has other strengths, that can be combined in a sensor fusion. In particular, cameras are usually strong in classification of objects and the perception of non-geometric attributes, like signs and road marks. Radar and Lidar sensors have strengths in the localization of objects and measuring their geometric attributes, as well as velocities. Therefore, the fusion of these sensor types is desired to combine the strengths and create redundancy.

Table 2.3: Comparison of the capabilities of different sensor types.

	Camera	Radar	Lidar
Object location	-	++	++
Object size	+	+	++
Object shape	+	-	++
Fine features	++	-	+
Motion	-	++	++
Classification	++	-	+
Signs and road marks	++	-	-
Robustness light conditions	-	++	+
Robustness weather conditions	-	++	+

2.5 Coordinate systems

The vehicle coordinate system is cartesian and anchored to the ego vehicle. Different definitions exist throughout literature and applications, however, for this thesis the vehicle coordinate system is located at the center of the truck's front axle on ground level. The X-axis points in driving direction, the Y-axis points to the left when viewed in driving direction and the Z-axis points upwards, as shown in Figure 2.10. Since the cabin is suspended separately, it is able to move independently from the vehicle coordinate system. Therefore, the cabin coordinate system shown in blue in Figure 2.10 is introduced.

In addition to the vehicle coordinates, each perception sensor has its own sensor coordinate system relative to a fixed point of the sensor. For cameras, the sensor coordinate system is typically located at the optical center, while the X and Y axes are aligned with the pixel coordinate system, as shown in Figure 2.11. The other sensors, like Lidar or radar, have sensor coordinate system, where the origin is located at a defined point with the X-axis pointing in the sensing direction (see Figure 2.11).

In order to fuse information from multiple sensors, they need to be transformed from the sensor coordinate system to the common vehicle coordinate system. The measurement of this transformation is also called extrinsic calibration. The transformation from the sensor to the vehicle coordinate system is usually done with a

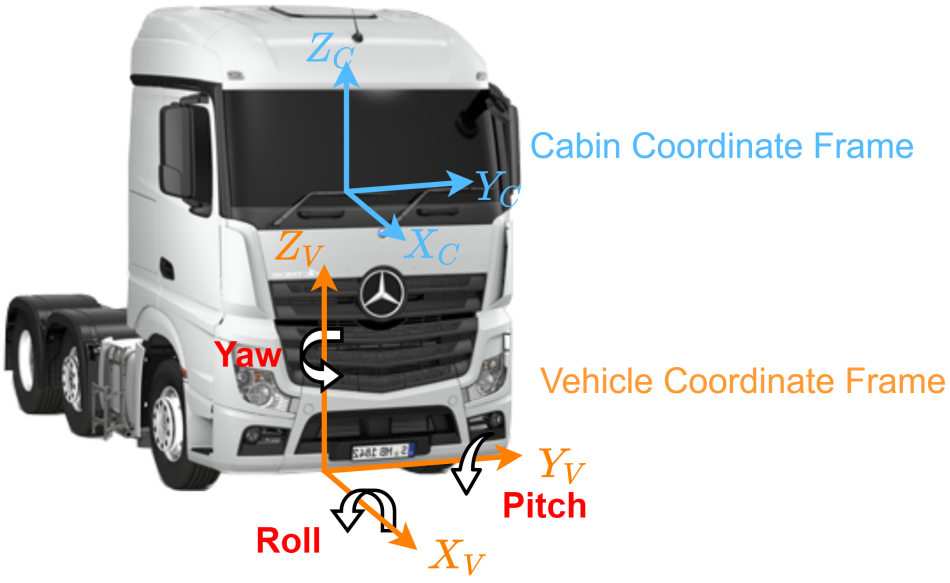


Figure 2.10: Vehicle and cabin coordinate system definition.

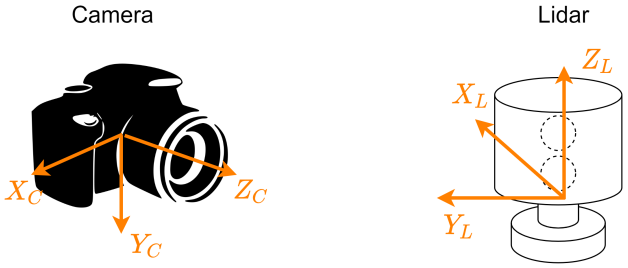


Figure 2.11: Sensor coordinate system examples for Camera and Lidar.

transformation matrix $\boldsymbol{T}_{s\rightarrow v}$, that transforms the state vector $\boldsymbol{x}_{Sensor} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ from sensor coordinates to vehicle coordinates $\boldsymbol{x}_{Vehicle}$:

$$\begin{bmatrix} \boldsymbol{x}_{Vehicle} \\ 1 \end{bmatrix} = \boldsymbol{T}_{s\rightarrow v} \begin{bmatrix} \boldsymbol{x}_{Sensor} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{Sensor} \\ 1 \end{bmatrix}. \tag{2.1}$$

The transformation matrix consists of the rotation matrix $\boldsymbol{R}^{(3\times 3)}$, describing the rotation between both coordinate systems, and the translation vector $\boldsymbol{t} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$, describing the translation between both coordinate systems. If the sensor state vector $\boldsymbol{x}_{Sensor}$ consists of additional information, like relative velocities, the transformation matrix $\boldsymbol{T}_{s\rightarrow v}$ may be more complex.

The raw data captured by camera systems is typically given in pixel coordinates $[u, v]^T$ and underlies distortion effects, which need to be corrected using an intrinsic calibration. The intrinsic calibration is often done using a checkerboard to derive a pinhole camera model and the lens distortion parameters (Punke et al. 2016). With the derived camera intrinsic matrix $\boldsymbol{K}$, a transformation from camera coordinates $\boldsymbol{x}_{Cam} = \begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^T$ to pixel coordinates $[u, v]^T$ with aspect ratio scaling s can be calculated by

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{K}\boldsymbol{x}_{Cam} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}, \tag{2.2}$$

where the camera matrix $\boldsymbol{K}$ consists of the focal lengths f_x and f_y and the principal point offsets u_0 and v_0.

Note that a vice versa calculation from pixel to camera coordinates is not possible without additional assumptions. Applying these assumptions is usually a key element in 3D detection algorithms based on monocular cameras, which is further discussed in chapter 3.3.1.

An important consideration for applying perception sensors to trucks is the mounting position, as the cabin is suspended separately from the chassis. This introduces additional degrees of freedom and results in constantly changing transformation matrices between the cabin coordinate system and the vehicle coordinate system. This issue can be addressed by continuously measuring the transformation with sensors, like Inertial Measurement Units (IMU). Another possibility is the compensation of the motion directly during the perception on sensor level.

3 Object detection

Over the last years, object detection is a rapidly evolving research field with a wide variety of approaches. It is the task of identifying and locating multiple objects of specific classes in scenes provided by sensors like cameras or Lidars. The outputs, performances and capabilities differ a lot depending on the sensor technology and the type of approach. Usually, the detection outputs include pose information, like the location, scale, bounding box or more detailed information like segmentation masks (Amit et al. 2020, p. 1).

This thesis focuses on 3D object detection methods that provide the pose of the detected objects in vehicle coordinates alongside with a classification and additional information. Detection methods that rely on detection-by-tracking are not considered in order to produce independent detections at each time-step. This is consistent with the Markov property used in many tracking approaches that states that the stochastic process is memoryless defined by $p\left(\boldsymbol{z}_k|\boldsymbol{x}_0,\ldots,\boldsymbol{x}_k\right) = p\left(\boldsymbol{z}_k|\boldsymbol{x}_k\right)$.

This chapter gives a general overview on state-of-the-art object detection methods for Lidar and camera sensors, as well as a short introduction to radar object detection. Since most state-of-the-art detection algorithms rely at least partially on deep learning, a short introduction is given as well. However, the main contributions are the proposal of a new Lidar-based object detection approach in chapter 3.2.3, which combines geometric and deep learning techniques for safety critical applications and the proposal of a new camera based detection algorithm for high sensor positions at the roof of a truck's cabin in section 3.3.3.

3.1 Object detection overview

In recent years, most published literature is using deep learning on camera and Lidar sensors for object detection, since these methods offer high accuracies. The surveys from Arnold et al. (2019) and Qian et al. (2021) give an overview on the current state-of-the-art for such 3D detection methods. Table 3.1 shows a short overview and comparison of different approaches, which are described in more detail in this section.

As shown by the table, fusion approaches have also been researched in recent years, in which the raw data from several perception sensors are fused before or during object detection. These approaches can reach high mean Average Precision (mAP) scores of over 90 % on the KITTI Dataset[1], especially when using low-level fusion of Lidar and camera like shown by Zhu et al. (2022) or Wang et al. (2021). However, in order to maintain a modular framework, this thesis focuses on separate object detection for each sensor and therefore is not using these approaches.

[1] referring to the KITTI "3D Object Detection Evaluation 2017" Benchmark (https://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d) for category "Car, easy", where cars with minimum bounding box height of 40 pixels in camera coordinates are evaluated.

Table 3.1: Overview of object detection approaches using different sensors. Composed from Arnold et al. (2019); Qian et al. (2021); Wu et al. (2021).

Sensor	Method	Description	Publications
Camera	Monocular	The detection is done based on single RGB images. The range information therefore has to be estimated. This can either be done by reprojection techniques, utilizing estimated 3D bounding box dimensions or directly estimating the range within the detection process.	Chabot et al. (2017); He and Soatto (2019); Li et al. (2019); Liu et al. (2021); Mousavian et al. (2017)
	Stereo	The detection is improved by providing range estimates from the stereo camera. This improves the location and dimension estimates to the cost of high computation requirements.	Liu et al. (2021)
	Pseudo Lidar	A depth value is estimated for each pixel using a neural network, which leads to a RGB-D point cloud. This can be used for object detection based on point cloud approaches.	Wang et al. (2019)
Lidar	Projection	The point clouds are projected to an image frame and processed using computer vision approaches. These can be extended using 3D processing approaches.	Barrera et al. (2020); Beltrán et al. (2018); Chen et al. (2017); Li et al. (2016); Simon et al. (2019b); Yang et al. (2018); Zeng et al. (2018); Zhou et al. (2019)
	Volumetric	The point clouds are converted into a volumetric representation structure, like voxels, which is the basis for the detection.	Chen et al. (2019); Kuang et al. (2020); Lang et al. (2019); Li (2017); Simon et al. (2019a); Yan et al. (2018); Zhou and Tuzel (2018)
	PC processing	The detection is done directly on the point cloud data, which can be done by both geometrical and deep learning approaches.	Charles et al. (2017a); Charles et al. (2017b); Shi et al. (2019); Yang et al. (2020); Burger and Wuensche (2018); Himmelsbach et al. (2010); Himmelsbach et al. (2009); Bogoslavskyi and Stachniss (2017)
Radar	Traditional radar	Objects are detected based on the 2D radar scan. Additional features, like the 3D shape are estimated using heuristics and signal processing.	
	4D imaging radar	Measures elevation angle in addition to azimuth, distance and velocity compared to traditional radars. Combined with increasing resolutions, this allows the application of point cloud based detection methods similar to Lidar sensors.	
Fusion	Low-level	Feeds the raw data from multiple sensors, like Lidar and camera, into a detection algorithm, which is usually a neural network. The performance can be high due to the exploitation of multiple modalities.	Wang et al. (2021); Roth et al. (2019)
	High-level	Individual detection approaches are executed for each sensor and their results are fused together. Allows the combination of strengths, like the classification from images combined with the range estimation of Lidar. During the fusion process, deep learning features from the individual approaches may be used.	Pang et al. (2020)

3.1.1 Deep learning in object detection

This chapter provides a short introduction to Neural Network (NN) and deep learning, since this field forms the basis for many modern object detection approaches. NN are machine learning techniques, which are based on connected artificial neurons. These structures are able to learn functions of an input by using training data and can be used in the fields of image processing, text- and speech recognition, control systems and applications in autonomous driving. Due to new NN techniques and architectures, deep learning is nowadays able to outperform humans on specific tasks. This is achieved with optimizations, like the increasing number of layers (increasing depth) or the utilization of convolutional layers in image processing. Because of the scope and complexity of the topic, only the basics of deep learning are covered in this thesis. For a greater overview, reference is made to the work of Aggarwal (2018).

The structure of NNs typically has several layers of artificial neurons interconnected, which is sometimes compared to the structure of a human brain. The neurons in the first layer receive the input information, perform calculations with it and pass it on to the next layer. With each layer, the processed information can

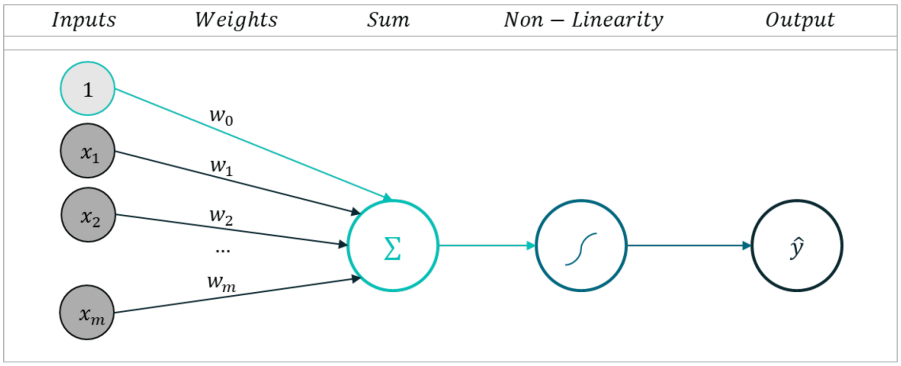


Figure 3.1: Schematic representation of a perceptron (Holz 2023).

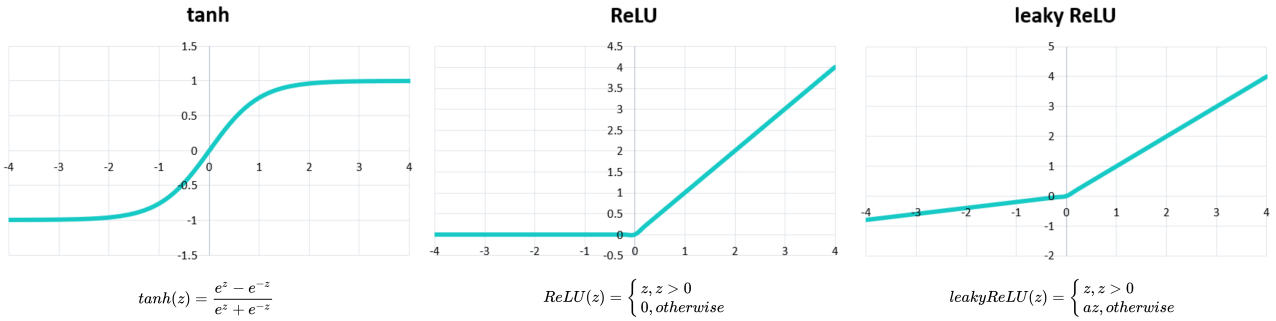


Figure 3.2: Visualization of tanh, ReLU and leaky ReLU activation functions with furmulas. a for leaky ReLU is a constant factor.

become more complex and in combination achieve complex tasks such as object detection on image data. A single neuron, also known as a perceptron, is the basic computational unit in a neural network and shown in Figure 3.1. Each of the input values $x_1, \ldots, x_m$ is multiplied with the corresponding weight $w_1, \ldots, w_m$ and summed up together with the bias weight w_0. Next, the output $\hat{y}$ with

$$\hat{y} = g(z) = g\left(w_0 + \sum_{i=1}^{m} x_i w_i\right) \tag{3.1}$$

is calculated, where $g()$ is the non-linear activation function.

Common activation functions include the Rectified Linear Unit (ReLU), leaky ReLU, and the hyperbolic tangent function, which are shown in Figure 3.2. The choice of activation function depends on the specific task and the architecture of the neural network. The ReLU activation function is a typical choice for modern deep learning architectures due to the ease in training (Aggarwal 2018) and the low computational effort.

A layer in a neural network is composed of multiple neurons, and each neuron in the layer performs the same operations as described above. In a single layer, each neuron has its unique set of weights and biases, which allows them to learn and recognize different features or patterns in the input data. This diversity of weights and biases across neurons in a layer enables the network to capture a broad range of features.

The simplest structure of a layer is called fully connected, where the inputs of each neuron are connected to the outputs of all neurons in the previous layer. In deep neural networks, information is processed through multiple layers, each containing numerous neurons. The layer structure is shown in Figure 3.3 with a shallow network on the left side and a deeper one on the right side. All layers that do not act as input or output of the network are called hidden layer.

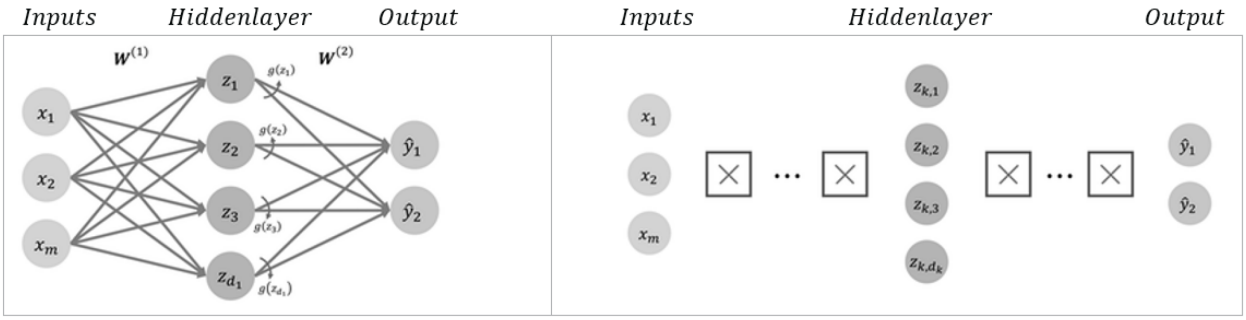


Figure 3.3: Example for fully connected layer with single input and output layer on left side, deep version on right side (Holz 2023). W_1 and W_1 are the sets of weights for the hidden and the output layer.

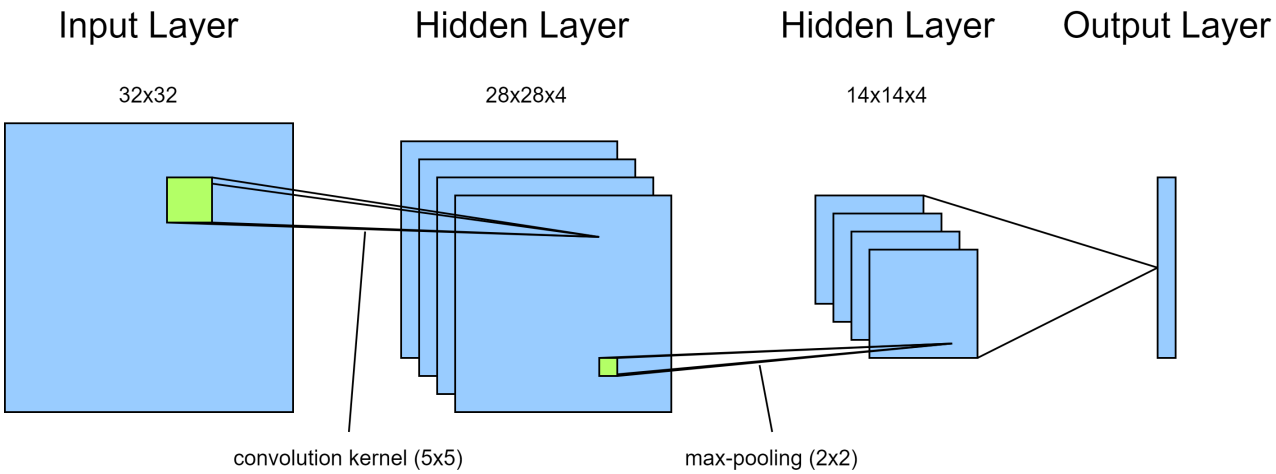


Figure 3.4: Example for CNN with one hidden convolutional layer and one hidden max-pooling layer.

Convolutional layers are a different type of layer and are the fundamental building blocks in many Convolutional Neural Network (CNN)s for image processing. These layers use filters (also known as kernels) to scan the input image in a systematic way, extracting local features. With each layer, increasingly complex and abstract features are created. In image processing tasks, the kernels usually work on 2-dimensional data structures, as shown in Figure 3.4, but can be applied to 3-dimensional data in a similar way.

Max-pooling layers are an important building block for CNNs as well. They can downsample the features obtained from the convolutional layers, as shown in Figure 3.4, by only keeping the maximum value of the kernel. This helps to reduce the spatial dimensions of the feature maps while retaining the most important information. The downsampling process reduces computation while improving the model's ability to recognize features in different scales and positions within the image.

The output layer of a NN in object detection applications typically performs either a classification or regression task. The classification assigns a label to a detected object, usually represented as a probability distribution over all possible classes. This is achieved by a softmax layer that calculates the probability $\sigma(z)_i$ of each output neuron i with $\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$ with k being the number of output neurons. A regression output layer on the other side estimates numerical values like the dimensions and orientation of a bounding box of a detected object. Therefore, both output layer types are used for typical 3D object detection approaches, since both the classification and 3D features are necessary.

The processing of data through a network is called feedforward process. To get meaningful results from this feedforward process, NNs are trained using backpropagation to calculate the gradients in combination with an

iterative optimization technique. The optimizer adjusts the model's weights and biases during training to minimize a loss-function, which is shown by Aggarwal (2018) in detail. The Adam optimizer is a popular choice due to its ability to effectively handle different learning rates for various parameters to effectively minimizing the loss-function.

The loss-function models the difference between prediction and ground truth and is chosen to match the output type. For classification tasks, a typical loss-function is the cross entropy loss $\mathcal{L}_{CE}$ with

$$\mathcal{L}_{CE} = -\sum_{i=1}^{k} y_i \log(\hat{y}_i), \tag{3.2}$$

where $\hat{y}_i$ is the prediction of the network, y_i the ground truth and k the number of exclusive classes. For regression tasks the mean square error loss $\mathcal{L}_{MSE}$ with

$$\mathcal{L}_{MSE} = -\sum_{i=1}^{k} (y_i - \hat{y}_i)^2 \tag{3.3}$$

is often used. Here, k is the number of output neurons with y_i being the ground truth values and $\hat{y}$ the regression outputs.

The building blocks described above are the basis of NN-based object detection for all sensor types described in this thesis. Depending on the application and type of data, the network architecture, training parameters, types and amount of layers and loss-functions will differ and need to be tuned.

3.2 Lidar-based object detection

The goal of object detection with Lidar sensors is to extract individual objects of different classes from a large point cloud with several hundred thousand points. This reduces complexity, which simplifies further processing steps. The output of the presented approaches are 3D OBBs and point cloud segments, which are either determined during the detection process or can be determined based on the OBB dimensions.

Object detection with Lidar sensors is currently being researched very intensively, as new deep learning techniques show increasingly strong results. Such approaches use a huge variety of different network architectures to extract 3D OBBs from the point cloud data. Traditional detection approaches on the other hand try to exploit the geometric relationships between points based on heuristic rules. This usually leads to lower Average Precision (AP) results, but they do not rely on training. Therefore, they can detect objects of previously unknown classes and can be adapted to new environments (e.g. sensors, mounting positions, ...) without new training data.

This thesis gives an overview of both types of approaches. In some cases, a combination of deep learning methods and traditional approaches can also be beneficial. Zhao et al. (2021) show an example, where a traditional clustering method can extend a NN-based semantic segmentation method to perform panoptic segmentation. This thesis introduces a new hybrid detection algorithm in chapter 3.2.3, that combines both types of approaches with a focus on safety relevant applications.

3.2.1 Traditional approaches

In contrast to deep learning approaches, the traditional object detection pipeline for Lidar sensors in automotive applications relies on the geometric relationship between points. Therefore, they follow a fixed algorithm and rules and do not require any training. Many of the point cloud processing techniques used for automotive

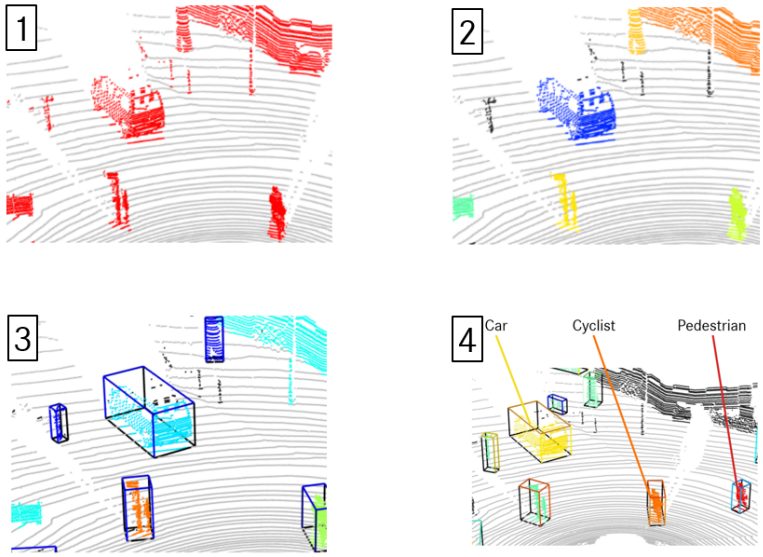


Figure 3.5: Four steps of a traditional Lidar object detection pipeline: 1. ground extraction, 2. object segmentation, 3. OBB generation, 4. classification of objects.

applications were originally developed for other domains, like engineering geodesy or monitoring. An overview of such techniques that have also been used for mobile and automotive applications is given by Che et al. (2019).

The methods used for ADAS typically associate each point with an object and thus perform a segmentation task while removing non-relevant points. Many of the algorithms are divided into multiple steps, as shown in Figure 3.5. Some steps may be combined depending on the approach.

0. Point cloud preprocessing
1. Ground extraction: All points, that correspond to the ground plane, are removed.
2. Object segmentation: The points are segmented, where each segment represents one object.
3. OBB generation: An OBB enclosing all points is generated based on the points positions.
4. Classification: The class of the object is estimated.

3.2.1.1 Ground extraction and object segmentation

Ground extraction is usually a preprocessing step for the object segmentation, because this can improve performance in sparse point clouds, as shown by Douillard et al. (2011). One way to separate the ground points is the application of a ground model, like a plane. This model is fitted using the Random Sample Consensus (RANSAC) approach or advanced models based on RANSAC, as shown by Choi et al. (2014), Asvadi et al. (2016) and Zermas et al. (2017).

Other methods, like shown by Chu et al. (2017) and Burger and Wuensche (2018), utilize the intrinsic properties of Lidar sensors, such as scan lines, to create graph structures. Similar methods can be used on range images created from point clouds, which also create a graph structure, as shown by Douillard et al. (2011). Based on height and angle thresholds within this graph, the ground points are extracted. The graph structures can also be used for the segmentation process, as shown by Burger and Wuensche (2018). Grid-based approaches divide the XY-plane into a rectangular (Himmelsbach et al. 2009; Thrun et al. 2007) or a polar (Himmelsbach et al. 2010; Kampker et al. 2018) grid and estimate the ground height of each cell. Based on this height, the object

points can be determined. Grid or graph structures can also be used for the segmentation process which can lead to an advantage in efficiency.

Object segmentation is the task of dividing the point cloud into segments, where each segment represents one physical object. Therefore, each point is assigned to one of these objects. There are a lot of traditional approaches facing this task, that are not specifically developed for ADAS applications, which are summarized by Nguyen and Le Bac (2013). However, some of these ideas, like the Euclidean clustering (Cao et al. 2022) directly work on automotive Lidar data and are the basis for modern techniques.

Segmentation algorithms try to find points which correspond to the same object. Most approaches rely on procedures, that connect neighboring points based on a set of homogeneity criteria. If a neighboring point of the current one meets some specific criteria, it is added to the segment, resulting in a growth. Neighboring points can be found by applying a grid in 2D/2.5D (Himmelsbach et al. 2009; Kampker et al. 2018) or 3D (Himmelsbach et al. 2010; Asvadi et al. 2016). Intrinsic structures can also be utilized, as shown by Zermas et al. (2017) with scan lines or by Burger and Wuensche (2018) with graph structures. In order to reduce complexity for the search of neighboring points in 3D, advanced data representation structures like Octrees (Vo et al. 2015) or KD-Trees can be used. In the most trivial case, the Euclidean distance is used to determine if points are added to a segment, like used by Klasing et al. (2008) and Cao et al. (2022). However, additional criteria depending on the representation can also be used, like the angular criteria in graph and line structures (Bogoslavskyi and Stachniss 2017; Burger and Wuensche 2018) or combining neighboring occupied grid cells (Kampker et al. 2018). Advanced point-wise features, like normal vectors (Klasing et al. 2009) or local point statistics (Lalonde et al. 2006) may be used by providing additional information about each point's affiliation.

3.2.1.2 Oriented bounding box generation

The most common type of object representation is the oriented bounding box. This type of representation is a rectangular body, which encloses all points of an object and is oriented to properly match the outer shape. The deterministic calculation of an OBB is a non-trivial task, as in some cases the shape of an object does not match a rectangular shape. The calculation is often done in the XY-Plane, while estimating the height using the minimum and maximum z-position of all points. Road users, like cars or trucks, frequently generate L-shaped point clouds. Therefore, some algorithms, such as those of Qu et al. (2018) and Zhang et al. (2017), are optimized for this type of object. Another possibility to determine the orientation is the usage of RANSAC (Teichman et al. 2011) or principal component analysis to find the dominant line, which is then used as orientation. Naujoks and Wuensche (2018) present an approach based on the convex hull and line creation heuristic that can find an orientation for L-shape and arbitrary objects. For ADAS applications, the focus on L-shaped objects usually best fits the requirements to adequately represent other road users.

3.2.1.3 Point cloud classification

An accurate classification allows applying different class-depending motion models and class-depending reactions of ADAS functions. In contrast to deep learning approaches, the traditional pipelines do not provide a classification alongside the detection. Therefore, this task must be handled in a separate step.

The classification of objects usually requires the extraction of hand-crafted features, such as geometric, radiometric, or contextual ones (Che et al. 2019), that may be represented in compact format like voxels or histograms. Heuristic, or rule-based approaches, like shown by Yu et al. (2014) and Herrmann (2019), determine the class using these attributes and a set of heuristically determined rules, thus requiring no training (Che et al. 2019). The execution times are generally very fast, but those approaches are less able to represent complex relationships and are more prone to sparse data. Thus, the results from Herrmann (2019) show, that the accuracy decreases sharply for distances above 25 m.

Better results in these difficult conditions can be achieved with advanced methods, such as Support Vector Machine (SVM) (Himmelsbach et al. 2009; Lin et al. 2018; Chen et al. 2014) or boosting methods (Teichman et al. 2011; Asvadi et al. 2016; Arras et al. 2007; Yoshioka et al. 2017). The number and complexity of the computed features also affects the result. NNs are getting more and more attention in the field of classification as they promise better performances with modern network architectures. As shown by Bader et al. (2021), they can often outperform other such methods and heuristic approaches. However, they require higher computation power to do so.

3.2.2 Deep learning approaches

Due to the great success of deep learning in image processing, these approaches were transferred to three-dimensional point clouds (Fernandes et al. 2021). These architectures therefore often require a transformation of the point cloud into a different representation. However, many of the more recent studies focus on specialized approaches to point clouds.

Object detection with deep learning for Lidar sensors has some unique challenges that come along with the properties of the data structure. Unlike images, Lidar point cloud data are high dimensional, sparse and unstructured, which yields to challenges in deep learning applications (Fernandes et al. 2021). A point cloud $P = \left\{p^{(i)} \mid i = 1, \ldots, n\right\}$ is usually represented as sets of $n \in \mathbb{N}$ permutation invariant points $p^{(i)} = \left(x^{(i)}, y^{(i)}, z^{(i)}\right)$, which consist of the 3D coordinates x, y, z. Depending on the type of sensor, additional properties can exist per point, like the intensity or reflectivity. It is impossible to use typical deep learning methods like convolutional layers directly to point clouds, due to their unordered data structure (Guo et al. 2020). Therefore, the point cloud is often converted to some sort of structured representation, which usually leads to a loss of information or high computation effort.

Lidar sensors usually have scanning rates of 10 Hz to 20 Hz, why the approaches need to be real-time capable. In real-world driving scenarios, there are many objects in close proximity that are partly or completely occluded. This requires detection frameworks to generalize in the case of incomplete point clouds. The points are affected by noise and the intensities may vary a lot (Li et al. 2021), which are additional requirements for generalization. The generalization of detection models is difficult, due to small datasets and a high variety in possible driving domains (motorways, urban areas, ...), changing weather conditions and different sensor setups (sensor resolutions, sensor FoV, mounting positions, ...). This is a problem in particular, because most approaches try to maximize their performance on a specific dataset, with "[...] no understanding on the required detection performance levels for reliable driving applications (Arnold et al. 2019, p. 3793)". Therefore, they do not have variations in the setup, only specific driving scenes and limited training data, which leads to low significance outside the boundary conditions.

There are several surveys and reviews that give an overview and a comparison of the existing state-of-the-art approaches (see Li et al. (2021); Qian et al. (2021); Arnold et al. (2019); Fernandes et al. (2021); Wu et al. (2021)). Fernandes et al. (2021) show an overview of current object detection methods with a focus on deep learning for point clouds, which has seen a rapid increase in studies since 2019. In contrast, Guo et al. (2020) show a more general overview of deep learning on 3D point clouds, including the tasks of classification, segmentation, and tracking. Meanwhile, Li et al. (2021) focus on various tasks of Lidar point cloud processing for autonomous driving using deep learning. A recent and detailed survey on deep learning object detection methods using Lidar sensors is given by Wu et al. (2021). Most of these surveys divide the approaches into three categories: Projection, volumetric/voxel and point-based. Wu et al. (2021) show an overview of these categories, summarized in Table 3.2:

Table 3.2: Comparison of categories of deep learning Lidar object detection approaches, based on Wu et al. (2021).

Point cloud representation	Methodology	Highlights	Limitations
Projection-based	Project the 3D point cloud into BEV or front view to generate a 2D feature map, then apply 2D CNNs on it for object detection	• Allows the application of off-the-shelf approaches and CNNs known from computer vision. • Similar view across Lidar and camera image for forward view makes it easy to fuse the camera image. • The size of objects in BEV remains constant regardless of the range, providing strong prior information about size and making the network training easier.	• Projection to 2D results in the loss of information. • Objects may occlude for forward view projection. • Many cells of BEV representation do not include points, which creates memory and computation overhead. • BEV representation is influenced by ground plane fluctuations and vertical occlusion for pole-like objects.
Volumetric/voxel-based	Structure the 3D space into regular voxels to create 3D feature maps where 3D CNNs can be applied	• 3D CNN can directly learn local 3D dependencies. • Voxelization naturally normalizes the non-homogeneous points input for Lidars with various channels. • Ability to carry-over techniques from mature 2D Detectors.	• Loss of information during voxelization. • Many voxels do not include points, which creates memory and computation overhead. • Computational and memory cost increase cubically with increasing voxel resolution.
Point-based	Directly process the irregular point cloud by extracting point-wise features and create predictions based on each point.	• Ability to fully exploit 3D geometry without information loss. • Low memory consumption due to processing of sparse point cloud.	• Direct application of well-known CNN approaches not possible. • Computation requirements are less predictive since they vary with size of point cloud.

3.2.2.1 Projection approaches

Lidar point clouds are not suitable for direct processing of 2D detection methods, which require a grid-like input. However, this type of network architecture is attractive, because they are well researched and benchmarked in the domain of 2D image processing (Arnold et al. 2019, p. 3787). Therefore, several approaches use a projection of the point cloud to an image plane.

Next to the utilization of well known deep learning architectures for 2D images, the advantages of projection approaches are the efficiency due to the reduction in dimensionality and the possibility to pretrain the models based on images (Su et al. 2015, p. 945). The main limitation of these approaches is the potential loss of information during the projection (Li et al. 2021, p. 3420).

The point cloud can be projected cylindrically to create a cylindrical image of the forward view. Another possibility is the projection to a BEV image. The BEV is the most popular representation and has some advantages over the others, because it is proportional to the metric space, the objects represent their real sizes and the occlusion problem is solved directly (Fernandes et al. 2021). The channels of the generated images can include statistical properties of the points in each cell, like the maximum or mean height, distance and intensity.

The images are then fed into a neural network. Different types of networks can be used, like 2-stage detectors with region proposals or single shot detectors with direct bounding box estimation. Some architectures incorporate multiple views in order to improve the results (Su et al. 2015; Chen et al. 2017).

3.2.2.2 Volumetric approaches

In order to make convolutional layers applicable on 3-dimensional data, this category represents the scene in a 3D grid or voxel grid to allow 3D-convolutions. A voxel is here defined as one cell in a uniformly constructed three-dimensional grid that divides space. One main drawback is the high number of empty grid cells, that do not contain points in sparse regions, which leads to inefficient calculations (Arnold et al. 2019). Additionally, the discretization can lead to information loss. In general, 3D operations are computational much less efficient compared to 2D operations, like the projection approaches use.

A feature vector is calculated for each of the voxel cells. This features can consist of occupancy, density, mean position and other geometric properties, as well as results from feature encoder networks introduced by Zhou and Tuzel (2018). These encoder networks are often based on PointNet's (Charles et al. 2017a) concept and calculate complex features of each cell.

Subsequently, several approaches (Zhou and Tuzel 2018; Yan et al. 2018; Lang et al. 2019; Kuang et al. 2020) consist of three parts: Feature encoder network, CNN backbone and detection head. In SECOND (Yan et al. 2018), the 3D convolutions are replaced by more efficient sparse convolutions, which leads to improved speed and performance. PointPillars (Lang et al. 2019) increases the speed even more by using pseudo BEV voxels and can run at 62 fps, which makes it suitable for real-world applications. Kuang et al. (2020) use a voxel-based feature pyramid to improve the performance. Fast Point R-CNN (Chen et al. 2019) and related methods use a different approach based on a two-step detector with voxel region proposals and a point-based refiner network.

3.2.2.3 Point-based approaches

The unordered nature of point clouds that comes along with $n!$ permutations and a variable size is a challenge for the application of traditional neural networks, that expect a fixed input. The pioneer work PointNet from Charles

et al. (2017a) solves this problem by the application of Multi Layer Perceptrons (MLP), which create point-wise features. These features are then globally aggregated by a symmetry function, like a max-pooling layer, which generates permutation invariance. In PointNet++ (Charles et al. 2017b), this is extended by a hierarchical structure to locally aggregate features. Several object detectors emerged from this network structure, like the point-based single stage detector 3DSSD (Yang et al. 2020) and the two-stage detector PointRCNN (Shi et al. 2019). Point-based approaches have no loss of information due to preprocessing, and the computational requirements depend on the point cloud size.

3.2.2.4 Performance comparison of Lidar-based deep learning detection

Table 3.3 shows a performance comparison of numerous deep learning based detection approaches on the KITTI detection benchmarks. The difficulty level easy, moderate and hard refers to the minimum bounding box size of the GT object in pixel coordinates and the degree of occlusion[2]. The methods are divided into the groups of projection, volumetric and point-based approaches. None of these groups is in general better than the others, although there is a tendency of the research directing towards point-based and point voxel hybrid solutions. The highest scores achieved on car moderate in this comparison is 79.57 % from 3DSSD (Yang et al. 2020).

Table 3.3: Comparison of AP of different Lidar detection methods on the KITTI 3D detection test set.

Type	Method	Car AP			Pedestrian AP			Cyclist AP		
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
Projection	BirdNet+ (Barrera et al. 2020)	76.15	64.04	59.79	41.55	35.06	32.93	65.67	53.84	49.06
	BirdNet (Beltrán et al. 2018)	40.99	27.26	25.32	22.04	17.08	15.82	43.98	30.25	27.21
	MV3D (Lidar)(Chen et al. 2017)	68.35	54.54	49.16	-	-	-	-	-	-
	Complex Yolo[a] (Simon et al. 2019b)	67.72	64.00	63.01	41.79	39.70	35.92	68.17	58.32	54.30
	PIXOR[b] (Yang et al. 2018)	81.70	77.05	72.95	-	-	-	-	-	-
	RT3D (Zeng et al. 2018)	23.74	19.14	18.86	-	-	-	-	-	-
	FVNet[b] (Zhou et al. 2019)	65.43	57.34	51.85	42.01	34.02	28.43	38.03	24.58	22.10
Volumetric	VoxelNet(Zhou and Tuzel 2018)[b]	81.97	65.46	62.85	57.86	53.42	48.87	67.17	47.65	45.11
	SECOND (Yan et al. 2018)	84.65	75.96	68.71	-	-	-	-	-	-
	Fast Point R-CNN (Chen et al. 2019)	85.29	77.40	70.24	-	-	-	-	-	-
	Voxel-FPN (Kuang et al. 2020)	85.64	76.70	69.44	-	-	-	-	-	-
	PointPillars (Lang et al. 2019)	82.58	74.31	68.99	51.45	41.92	38.89	77.10	58.65	51.92
	Complexer-YOLO (Simon et al. 2019a)	55.93	47.34	42.60	17.60	13.96	12.70	24.27	18.53	17.31
Point-based	3D-FCN (Li 2017)	70.62	61.67	55.61	-	-	-	-	-	-
	PointRCNN (Shi et al. 2019)	86.96	75.64	70.70	47.98	39.37	36.01	74.96	58.82	52.53
	3DSSD (Yang et al. 2020)	88.36	79.57	74.55	54.64	44.27	40.23	82.48	64.10	56.90

[a]Results on splitted validation set instead of official test set
[b]Results not officially listed

3.2.3 Two-stage detection approach for safety critical Lidar object detection

Despite the good results on datasets like KITTI, deep learning approaches can randomly fail to detect objects, even if there is clearly a physical object in the point cloud. Reasons for that may be a lack of training data,

[2]Easy: Min. bounding box height: 40 Px, Max. occlusion level: Fully visible, Max. truncation: 15 %.
Moderate: Min. bounding box height: 25 Px, Max. occlusion level: Partly occluded, Max. truncation: 30%.
Hard: Min. bounding box height: 25 Px, Max. occlusion level: Difficult to see, Max. truncation: 50%.

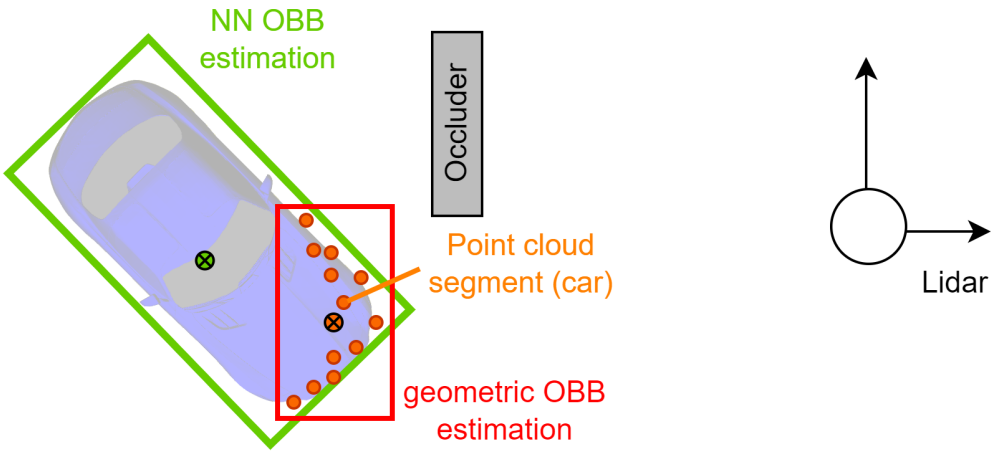


Figure 3.6: Point cloud segment of partially occluded car and example OBB estimations for NN and geometric approach. NN is able to estimate an OBB close to the GT, while geometric approaches are not.

overfitting, or too less training for objects of rare appearance. In safety critical systems, like ADAS, such false negatives might be a threat for individual situations, even though the overall performance of the approach is good. In contrast to most literature, which focuses on improving the overall performance, this thesis proposes an algorithm to decrease the number of non-detected objects to a minimum while still preserving a state-of-the-art performance on the KITTI 3D detection benchmark by splitting the detection into two stages:

The first stage applies an object segmentation based on traditional approaches, while the second stage performs the classification and OBB regression based on deep learning. The main advantage is a robust detection of all types of objects regardless of training, which is necessary for safe operation. Even if a special vehicle with unique appearance is missing in the training data, the traditional object segmentation is able to detect the physical existence, which might not be the case for a deep learning approach. The second stage of the approach refines the detection results to a state-of-the-art level performance by the deep learning classification and OBB regression. This brings a huge performance advantage over pure traditional approaches both in terms of classification and OBB accuracy.

Assume a car being partially occluded by some other object, as shown in Figure 3.6. In this case, it is possible to detect the remaining point cloud segment robustly using traditional approaches, which leads to the assumption that there is a solid object at the detection position. However, the classification is unlikely to work with a traditional approach. In addition, the Ground Truth (GT) OBB (visualized in green) is larger then the segment's geometric dimensions. Since traditional approaches are bound to the geometric properties, it will not be possible to calculate the correct OBB dimensions, position and orientation. A deep learning approach on the other hand can achieve this, if it has been trained sufficiently with data from similar situations and has learned that such a segment belongs to a car with specific OBB.

Authors like Schwalbe and Schels (2020) advocate such a splitting of NN-based functions into human interpretable sub-functions for argumentation within the ISO 26262 functional safety standard, which is another benefit of the two-stage approach. Instead of detecting objects with a large NN, it is done in two steps interpretable for humans and with separately measurable outputs. This makes it easier to prove the robustness of the approach. An overview of the two stages is illustrated in Figure 3.7, which are described in detail in the next sections.

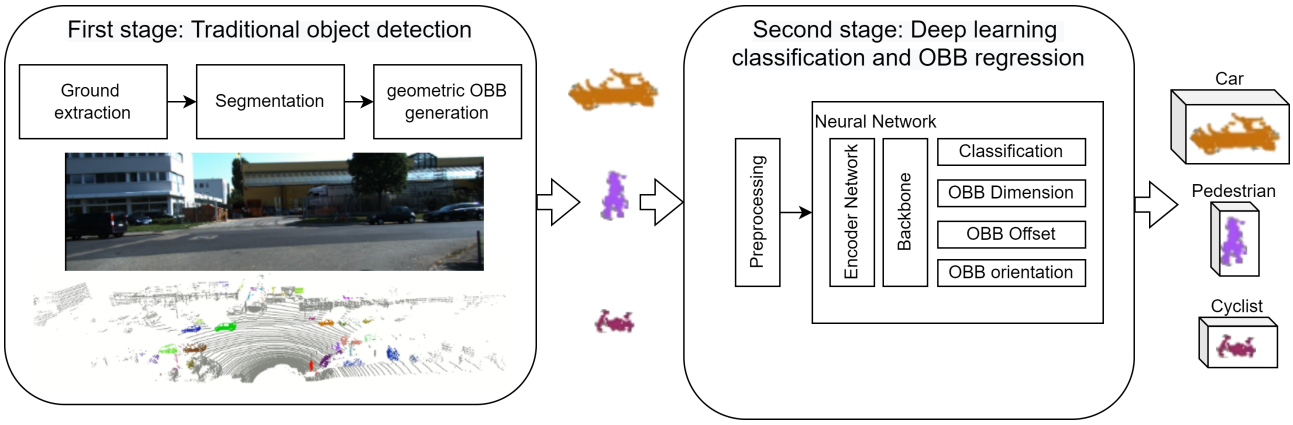


Figure 3.7: Overview of two-stage Lidar detection approach. The first stage consists of ground extraction, segmentation and geometric OBB generation. The point cloud segments are then passed to the second stage, which consists of a preprocessing and a neural network to estimate the class and OBB properties.

3.2.3.1 First stage: Traditional object detection

A traditional object detection pipeline is used, consisting of ground extraction, object segmentation and OBB calculation. The ground extraction and segmentation are simple approaches based on cartesian grids. This layout allows applying the approach to point clouds from multiple types of Lidars including different scan patterns and fused point clouds. It does not depend on the point cloud intrinsic and does not imply a single point of scanning, like the radial grid approaches (Himmelsbach et al. 2010; Kampker et al. 2018) do. Therefore, this approach is applicable to a wide range of sensors setups with few adaptations of the parameters.

Ground extraction: A rectangular, 2D grid of the dimensions $l_{grid,G}, w_{grid,G}$ with a cell length of $l_{cell,G}$ is used as representation in BEV, which results in $n_{l,G} \in \mathbb{N}, n_{w,G} \in \mathbb{N}$ grid cells. Each point $p^{(i)} \in P$ of a Lidar point cloud P with $p^{(i)} = \left(x^{(i)}, y^{(i)}, z^{(i)}\right)$ is then assigned to one of the grid cells $C^{(j,k)}$, where it lies in.

The ground extraction is performed in multiple steps: First, all outlier points below a negative slope of t_{ns} radians from an assumed flat ground plane are removed, to exclude points from reflection artifacts, as shown in the first step of Figure 3.8, where h_s is the sensor height above ground.

Next, for each cell $C^{(j,k)}$, the local ground level $z^{(j,k)}_{ground}$ is estimated. A simple assumption for the local ground level is to use the minimum z-value of all points in the cell. To gain robustness compared to this simple approach, the lowest z-values of all surrounding grid cells within a kernel size of $k \times k$ are taken into account, too. This is necessary, if a cell does not include a ground point while the neighboring cells do. The current cell's lowest z-value is compared to the median of the lowest z-values of all cells in the kernel. If it deviates more than the object height threshold t_{oh}, the local ground level $z^{(j,k)}_{ground}$ is set to the median value, otherwise to the lowest z-value of the current cell. In Figure 3.8, the second step shows this process, where the lowest points of all cells in the kernel are shown, with the kernel size being marked red. The dark blue point is the median of the kernel, while the orange-marked point has the lowest z-value of the current cell and deviates from the median. Therefore, the median value is chosen as local ground level in this example.

Next, all points, with $z^{(i)} - z^{(j,k)}_{ground} < t_{gh}$ with $z^{(i)} \in C^{(j,k)}$ are then defined as ground points and removed from the point cloud, where t_{gh} is the threshold for the ground height. All points in Figure 3.8 that lie in the green area are marked as ground points. Similar, all points with $z^{(i)} - z^{(j,k)}_{ground} > t_{mh}$ with $z^{(i)} \in C^{(j,k)}$ are defined as points with height out of reach, like high trees or bridges and therefore removed as well. t_{mh} is the maximum reachable height and is chosen greater than the vehicle height. In Figure 3.8 showing the points

inside a single cell, the point in the blue-marked area is defined as out of reach, while the dark blue points remain as object points.

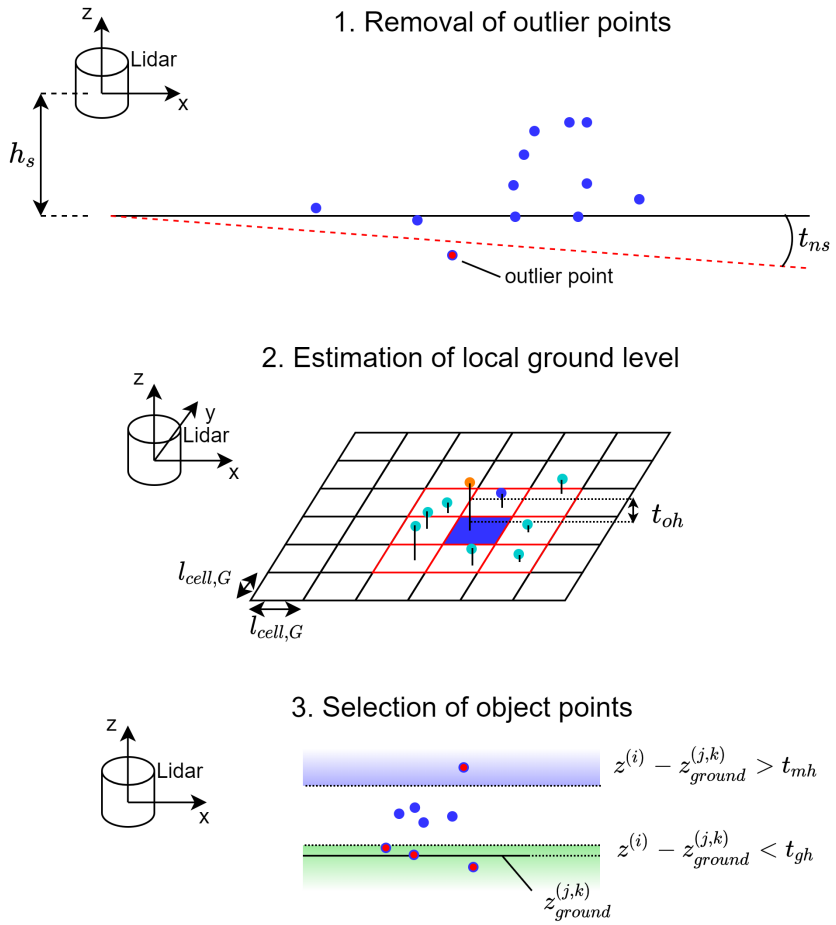


Figure 3.8: Scheme of ground extraction process in three steps. The first step shows the outlier removal based on a negative slope between an assumed flat ground plane and each point. The second step shows the estimation of the local ground height. The point with the lowest z-value of each cell is sketched. The dark blue-marked point has the median z-value, while the orange marked point is from the currently evaluated cell, which is outside the object height threshold t_{oh}. The third step shows the selection of the object points within one cell, where all points in the green area are ground points and all points in the blue area are out of reach.

Segmentation: The segmentation applies the remaining points on a similar grid, like the ground extraction, and connects occupied cells close to each other. The cell length $l_{cell,S}$ of the segmentation may vary from the one used for the ground extraction $l_{cell,G}$. Each cell that contains points is connected to all cells containing points within a segmentation range of t_{sr}. Due to the decreasing point density over distance, the segmentation range is calculated dynamically depending on the distance in the XY-Plane of the currently evaluated cell: $t_{sr} = t_{sr\_0} + min(d(C^{(i,j)})t_{sr\_f}, t_{sr\_max})$. The segmentation range is calculated based on the cells distance $d(C^{(i,j)})$ and a range factor t_{sr_f}, while being limited by a minimum search radius of t_{sr_0} and a maximum radius t_{sr_max}. All the created segments are now evaluated for the number of points. If the total number of points of a segment is below t_{np}, the segment is removed to get rid of small artifacts. In Figure 3.9, the search radius is sketched on the left side, while an example for the result of the segmentation is shown on the right side with segments marked in different colors. The small, red hatched segment is a removed artifact.

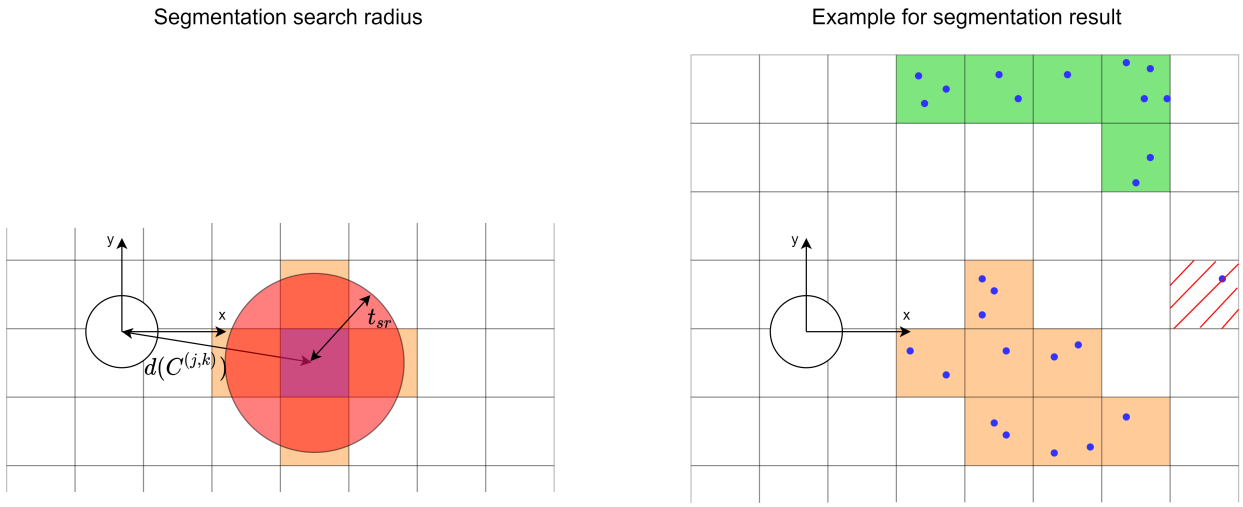


Figure 3.9: Scheme of the search radius of the current blue-marked cell on the left side. The orange-marked cells are connected if they contain points since their centers lie within the search radius. The right side shows an example result of the segmentation process with two segments marked in different colors. The red hatched segment gets removed since it's number of points is below t_{np}.

Geometric OBB generation: Although the deep learning based OBB generation is part of the second stage of the detection approach, a traditional fallback solution for objects of class "background" is developed. This is applied to objects of classes without learnable patterns, like plants, buildings or special vehicles, where the performance of a deep learning approach is expected to be worse. The goal of the OBB calculation algorithm is to generate bounding boxes for point cloud segments within a low time. It should also be able to calculate correct orientations for L-shapes, as well as arbitrary object shapes. The approach is based on the proposal from Bader (2019). Figure 3.10 shows the calculation steps. First, the convex hull of all points in the segment is calculated using the Graham scan algorithm (Graham 1972), which generates the blue points on the left sketch in Figure 3.10. Next, the two points with the highest distance of all convex hull points are selected to create the line $l_{diagonal}$. The line between these points is assumed to be the diagonal of an L-shape object. Now, the point with the highest orthogonal distance d_P to the diagonal is searched in a circular area d_c. The lines between the three selected points create a triangle. Other approaches, like the one from Naujoks and Wuensche (2018) also use triangles for the orientation estimation. In contrast to Naujoks and Wuensche (2018), the line selection is done counting the number of points in close proximity to the line with distances lower than d_{Lin}. Therefore, each of the lines has a number of corresponding points. If this number of points corresponding to the diagonal is higher than the combined number of points corresponding to the other lines, the orientation of the diagonal is used. Otherwise, the orientation of the longest of the other lines is used. The size of length l, width w and height h and center position P_{center} are now calculated using the maximum and minimum in the direction of the chosen orientation θ.

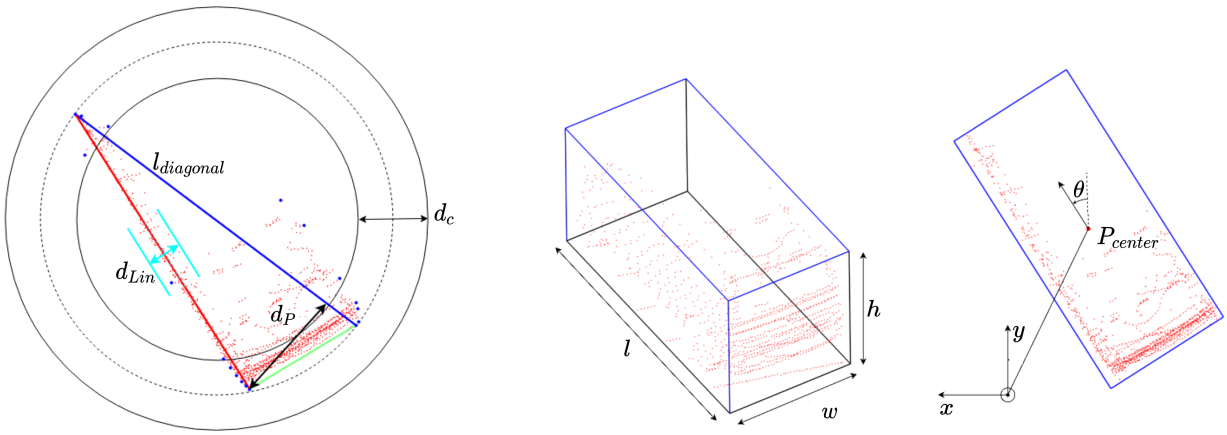


Figure 3.10: Calculation of the oriented bounding box. First, the three triangle lines are searched for. In this example, the diagonal line has a lower number of associated points, so the red line is chosen for orientation. Next, the length, width and height and position are calculated based on the orientation.

3.2.3.2 Second stage: Classification and OBB regression

The first stage of the detection pipeline is able to detect object segments correctly. However, there is no heuristic method for classifying these segments with an acceptable accuracy. In addition, geometrically calculated OBBs are prone to errors due to (self-)occlusion. As a result of the occlusion, the calculated bounding box of the same object may vary over time and can lead to reduced tracking performance. Machine learning techniques usually achieve better results in both of these domains. While there is a lot of research for the classification of segments, the combined regression of OBBs in parallel to the classification is unique for this approach. However, it enables a precise OBB estimation without a lot of additional computational overhead, since the encoder and backbone are shared with the classification.

Bader et al. (2021) describe an efficient deep learning approach for classification of point cloud segments with focus on efficiency and real-time capability. The approach classifies point cloud segments, which are created by a traditional detection method, into the classes background, car, cyclist, and pedestrian with an accuracy of 96.8 % and a runtime of 0.43 ms per segment on the Stanford Track Collection (STC). Based on the work of Bader et al. (2021), the approach is extended to estimate the OBB dimensions in addition. This allows to generate more precise OBBs, that are more consistent over time compared to pure traditional approaches. The deep learning pipeline consists of a preprocessing step and three main parts, similar to many examples for voxel-based object detection (Zhou and Tuzel 2018; Yan et al. 2018; Lang et al. 2019; Kuang et al. 2020).

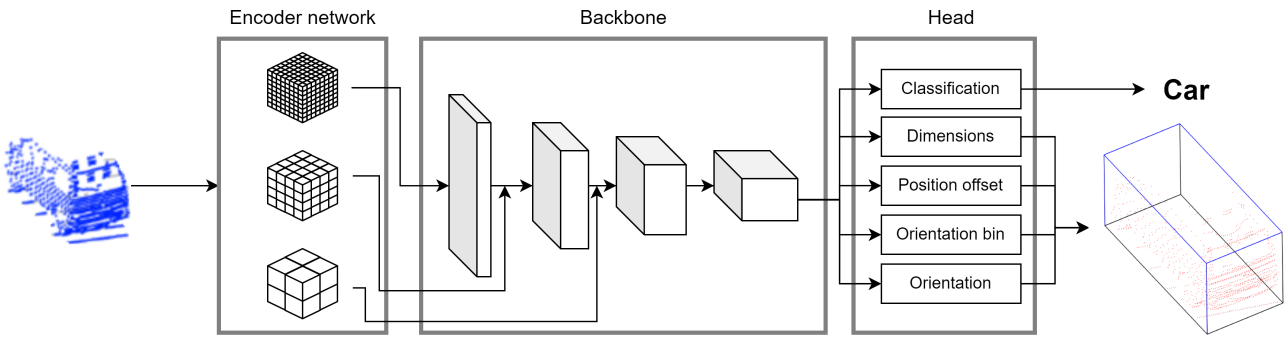


Figure 3.11: Architecture of network, which consists of an encoder network, a backbone and a several heads to output the classification and OBB regression.

The network is divided into the three parts encoder network, backbone and head, which are shown in Figure 3.11. The encoder network extracts features from the point cloud and provides them in a voxel structure on three scales. The backbone is built up of 3D convolutional and max-pool layers creating complex features. The head consists of multiple small sub-networks that perform the classification and OBB estimation based on the backbone's features. For all of the three parts, an overview is given in this thesis. However, since the encoder network and backbone is shared with the proposal from Bader et al. (2021), only the head is described in full detail.

Before the point cloud is processed by the encoder network, a preprocessing step is applied: An equally spaced voxel grid of size $M \times M \times M$ is fitted to the dimensions of the point cloud segment. Each point's position is then represented relatively to the center of its corresponding voxel in range $[-1, 1]$. This preprocessing is applied for each of the three scales of $M = 8$, $M = 4$ and $M = 2$ separately. These three grid sizes have been determined experimentally to optimize the accuracy (Bader et al. 2021).

Encoder Network: The three encoder networks directly operate on the point cloud and create features in form of a voxel grid. Therefore, the encoder networks convert the unordered point cloud data to a grid structure with features, that can be utilized by 3D convolutional layers. Each encoder network consists of three Modified Voxel Feature Encoder (MVFE) layers, which are the proposal of new modified versions of the Voxel Feature Encoder layer introduced by Zhou and Tuzel (2018).

The structures of both the encoder network and a single MVFE layer are shown in Figure 3.12, where M is the size of the voxel grid, C is the number of feature channels and N is the number of points. In the Figure, the process is exemplified with three points of different colors, which are located in two different voxels in different shades of gray. Each point contains the x-, y- and z-position and the Lidar intensity, resulting in an input of $N \times 4$ features. The features are extended to $N \times C$ by the successive MVFE layers. The subsequent max-pooling layer serves as a symmetry function, whereby a max-pooling operation is applied to all points that correspond to the same voxel. This creates output features in an $M \times M \times M \times C$ grid structure suitable for 3D CNNs. Within a single MVFE layer, an extended feature vector is first created for each point using a 1D CNN. The features of each voxel are then generated by max-pooling and concatenated with the original point features. In contrast to the original implementation from Zhou and Tuzel (2018), the concatenation step is done using the original input features instead of the transformed ones. This improvement allows a higher information content for voxels with a low number of points (Bader et al. 2021).

Overall, the encoder networks work directly on point-wise features and extract local features of each voxel. Three encoder networks are applied to the point cloud segment at grids of size $M = 8$, $M = 4$ and $M = 2$ to extract voxel features at different size levels. Compared to other approaches working with feature encoder layers, this allows to extract features of multiple spatial sizes directly during the encoder network. The implementation allows processing a different number of points per voxel, as shown by Bader et al. (2021). Therefore, no sub-sampling within the voxels is necessary while maintaining processing speed.

Backbone: A 3D CNN is used as backbone, as showed in Figure 3.13. It consists of multiple stages of two 3D convolutional and one max-pool layer, as these types of architectures are well established for voxel processing (Maturana and Scherer 2015; Zhou and Tuzel 2018; Lang et al. 2019). The encoder networks of size $M = 4$ and $M = 2$ are concatenated with the features after the first and second max-pool operation. All network hyperparameters shown in Figure 3.13, like the number of convolutional layers and feature channels C are experimentally determined. The implementation details of the encoder networks, the backbone, and the classification head are described by Bader et al. (2021).

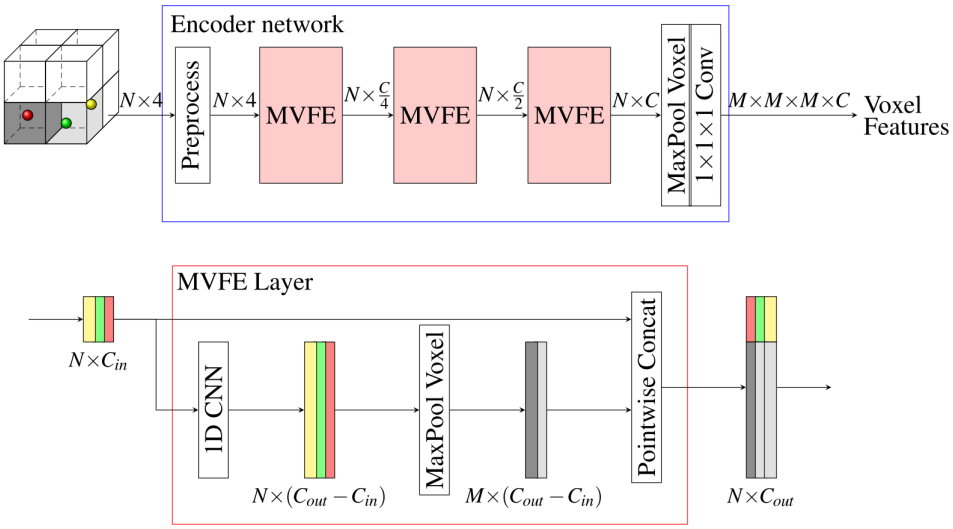


Figure 3.12: On top, the structure of the encoder network is shown, which consists of three MVFE layers and creates features for each voxel. The lower half shows the structure of the MVFE layer. In this visualization, an example with 3 points spread across two voxels is shown (Bader et al. 2021).

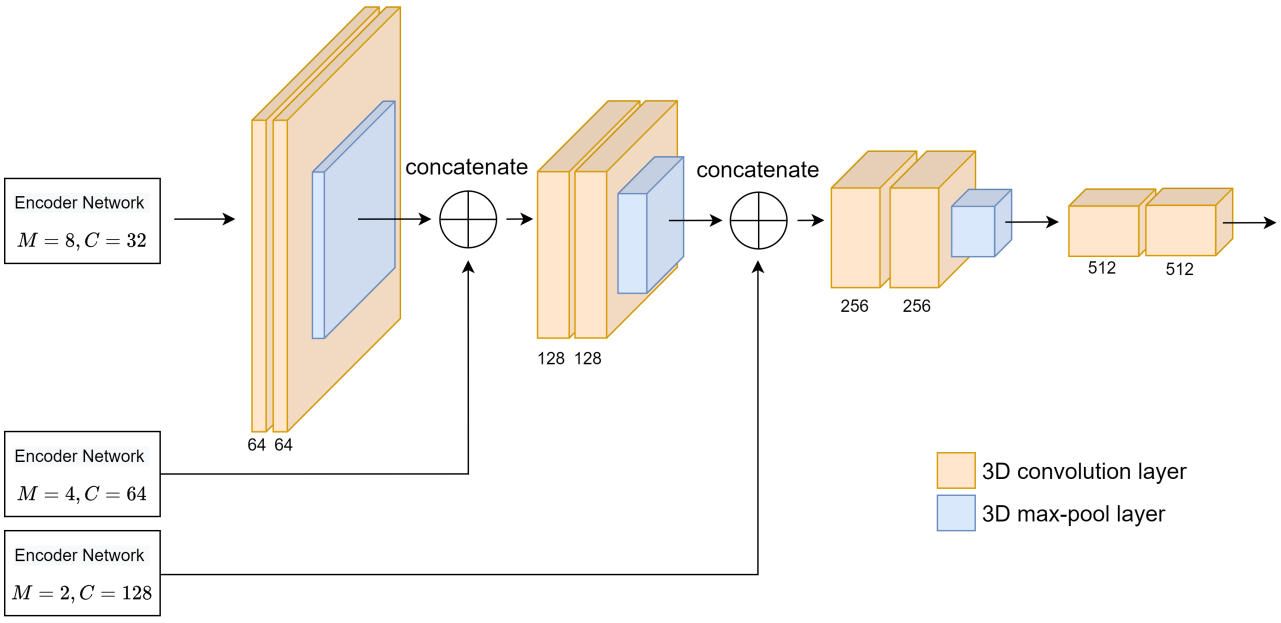


Figure 3.13: Backbone of NN consisting mainly of convolutional and max-pool layers. For simplicity, the 3-dimensional grid is reduced to 2D in the Figure. The number of feature channels is described below each convolutional layer.

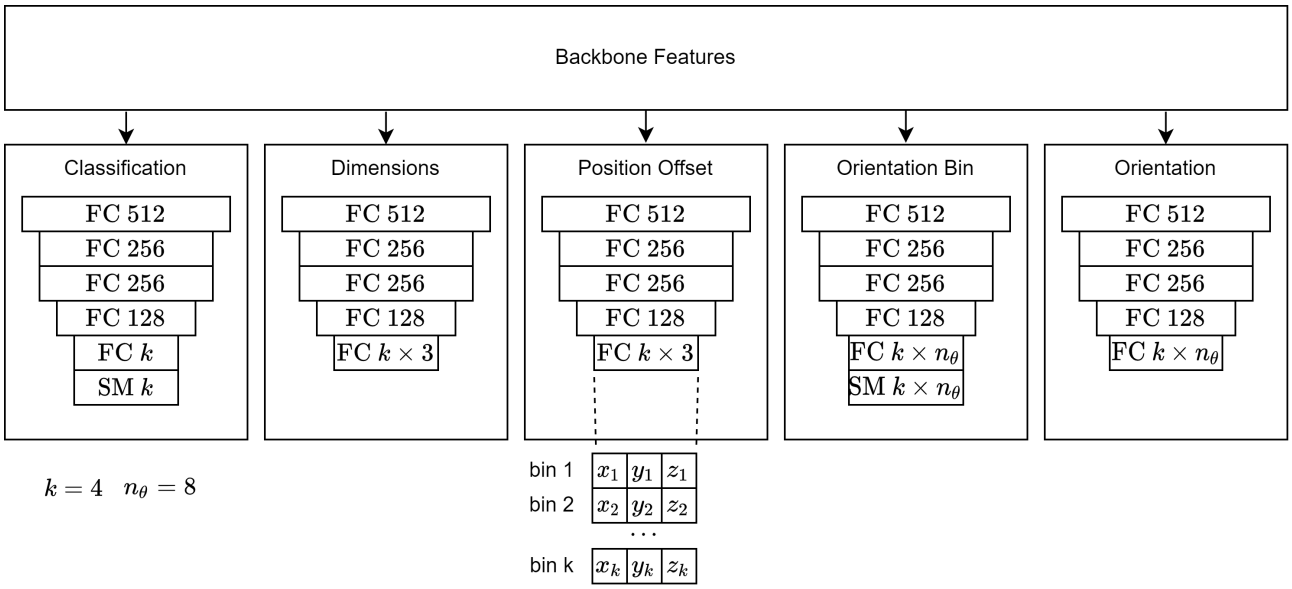


Figure 3.14: Architecture of head with multiple branches. Inside of each branch, the layers are described by the number of fully connected (FC) neurons . For classification and orientation bin branch, a softmax (SM) layer is added. The outputs of all branches except classification consist of k bins, where k is the number of classes. For the position offset branch, the structure of the bins in the output layer is shown.

Head: In contrast to the work of Bader et al. (2021), the head does not only consist of the classification. Instead, a dimension regression branch, a position offset regression branch, an orientation classification branch and an orientation regression branch, as shown in Figure 3.14. These branches are used to calculate the features visualized in Figure 3.15, where the combination of the orientation classification branch and the orientation regression branch is used to calculate the final orientation θ. All branches consist of five fully connected layers and work on the features extracted by the backbone, which is similar to the structure used by Mousavian et al. (2017) on monocular camera detection. The output layers use linear functions, while all other fully connected layers use leaky ReLU activation functions, with the layer sizes are shown in Figure 3.14. For each of the branches, a separate loss is calculated, which is summed up for training.

The actual output values of different classes may vary vastly. For example, the dimensions of a truck and the variance of the dimensions are different those of a pedestrian. Since it is difficult to represent a large range

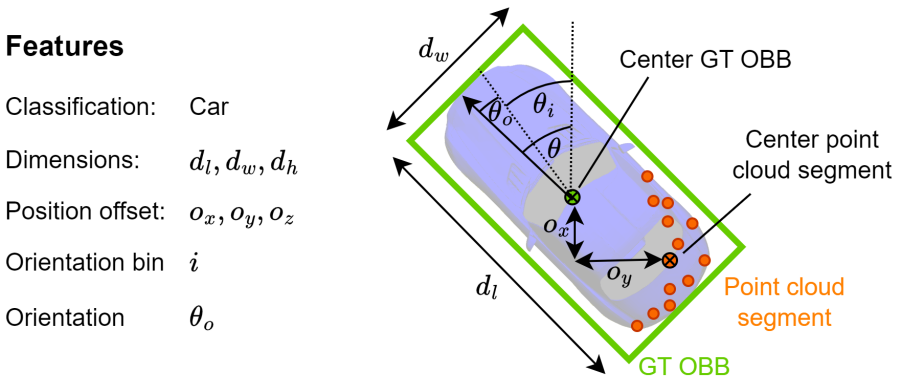


Figure 3.15: Features calculated by each branch visualized.

of values with a single regression output neuron, for each class a separate regression output is calculated in parallel. These parallel outputs are called bins and are applied to all branches except the classification branch itself. This allows the model to learn different values for each class separately. Figure 3.14 shows how the output layer is structured into bins for the position offset branch.

The dimension branch estimates the length, width, and height of the object and therefore regresses an output of size $k \times 3$, where k is the number of classes and thus the number of bins. The smooth L1 loss (Girshick 2015) function

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & if|x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \tag{3.4}$$

is used to calculate the dimension loss $\mathcal{L}_D$ defined by

$$\mathcal{L}_D = \sum_{i \in \{l,w,h\}} smooth_{L_1}(d_i^{(c)} - \hat{d}_i^{(c)}), \tag{3.5}$$

where $d_i^{(c)}$ is the ground truth dimension of the ground truth class c and $\hat{d}_i^{(c)}$ is the prediction of the bin of ground truth class c for length l, width w and height h. Consequently, only the loss of the bin of the ground truth class of each example is taken into account.

The center of the point cloud can be easily calculated, but usually does not match the center of the GT OBB. Therefore, the offset between the segments center and the ground truth center is regressed by the position offset branch, as shown in Figure 3.15. Similar to the dimension branch, the smooth L1 loss is used for offset loss $\mathcal{L}_O$ with the GT offset $o_i^{(c)}$ and the predicted offset $\hat{o}_i^{(c)}$ of GT class c:

$$\mathcal{L}_O = \sum_{i \in \{x,y,z\}} smooth_{L_1}(o_i^{(c)} - \hat{o}_i^{(c)}). \tag{3.6}$$

Inspired by Mousavian et al. (2017), the orientation θ with $-\pi \leq \theta \leq \pi$ is divided into n_θ orientation bins of size $\frac{2\pi}{n_\theta}$. The ground truth orientation is assigned to the orientation bin $i = \lfloor \frac{n_\theta(\theta+\pi)}{2\pi} \rfloor$. The orientation bin branch estimates the bin i, while the orientation regression branch estimates the orientation offset θ_o to the orientation bin θ_i. Based on the network's predictions, the orientation is calculated by

$$\theta = \theta_i + \theta_o = \frac{2i\pi}{n_\theta} + \theta_o. \tag{3.7}$$

This discretization into orientation bins improves the orientation accuracy. Both the orientation bin branch and the orientation regression branch therefore predict an output of size $k \times n_\theta$. The orientation loss is a combined loss of both branches with

$$\mathcal{L}_\theta = \mathcal{L}_{bin} + w_{rot}\mathcal{L}_{rot}, \tag{3.8}$$

where the bin loss $\mathcal{L}_{bin}$ is calculated using the cross entropy loss shown in equation (3.2). The rotation loss factor w_{rot} is used to balance the impact of both loss functions. For the rotation loss $\mathcal{L}_{rot}$, a cosine loss function is used:

$$\mathcal{L}_{rot} = 1 - cos(\theta_o^{(c,i)} - \hat{\theta_o}^{(c,i)}). \tag{3.9}$$

$\hat{\theta_o}^{(c,i)}$ is the predicted rotation offset and $\theta_o^{(c,i)}$ the GT rotation offset of ground truth class c and ground truth orientation bin i.

Processing: In order to process a batch of multiple segments in parallel, a fixed point cloud size is used. Therefore, all point cloud segments are either zero padded or randomly subsampled to match this size. Bader et al. (2021) show, that a fixed size of 512 points is a reasonable choice for point cloud segments extracted from the KITTI dataset.

The above described two-stage approach for Lidar object detection is tested on the KITTI dataset. The results of that can be found in chapter 6.3.

3.3 Camera-based object detection

Camera-based object detection has been a focus of research for decades. Modern cameras outperform other perception sensors easily in terms of resolution and production cost. In addition, latest developments in the area of deep learning massively improved the performance of image processing tasks, like semantic segmentation, lane or object detection.

In the beginning of image-based object detection, the algorithms were mainly based on handcrafted features (Zou et al. 2019), sometimes combined with machine learning techniques like AdaBoost. Examples of these early detection approaches are the Viola Jones Detector (Viola and Jones 2001), the Histogram of Oriented Gradients (HOG) detector (Dalal and Triggs 2005) and the Deformable Part-based Model (DPM) (Felzenszwalb et al. 2008). In automotive applications, such traditional algorithms are designed to work under clear constraints in specific situations, like the detection of vehicle rear-views for highway scenarios. As an example, Rezaei et al. (2015) show an approach with rear light detection and a virtual symmetry function for rear-view vehicle detection, which shows robust results even at night. However, approaches like this are limited to specific situations and perform poorly on general datasets, like KITTI.

Since RCNN (Girshick et al. 2014) was proposed in 2014, CNN-based methods started to clearly outperform the traditional ones. CNN-based detectors can be divided into two groups: One-stage detectors with the pioneer work YOLO (Redmon et al. 2016) and two-stage detectors with the pioneer work RCNN (Girshick et al. 2014). In two-stage detectors, the first stage suggests object candidates, which are extracted. During the second stage, these proposals are fed into a CNN to predict features like the class and the bounding box. One stage detectors divide the image into regions and calculate the bounding boxes and probabilities for existing objects for each region at the same time. This leads to a single processing step in a single CNN. Both of these network types were improved by a wide variety of developments, leading to great performances on several image processing benchmarks. One major drawback of the CNN-based approaches is the typically high computational effort. Therefore, they usually rely on a GPU or specialized chips for fast computation.

With the publication of modern perception datasets, like KITTI, the tasks of image processing were extended to 3-dimensional environment perception. For an object-based multi-sensor multi-object tracking system on trucks, the task of 3D object detection is of particular interest. Due to lower costs, manufacturers mainly focus on monocular camera systems. Therefore, this thesis gives an overview of methods for monocular 3D detection algorithms and the application of such algorithms on a truck. In chapter 3.3.3, the development and application of a new object detection approach for trucks and close distances is described. Due to the outstanding performance of CNNs compared to traditional approaches and their maturity, only CNN approaches are used in this thesis.

3.3.1 Monocular CNN-based 3D detection algorithms

The task of extending object detection in image coordinates to 3D objects is of central importance for ADAS camera systems. Often, 2D CNNs are used and depth estimation is performed either by applying assumptions or by extending the CNN. Kim and Hwang (2021) give an overview of pure deep learning monocular 3D detection algorithms. Here, the different methods for extracting 3D OBBs from monocular images are divided into three groups, which are listed below and visualized in Figure 3.16:

- Multi-stage approaches:
  These approaches typically are built upon a first stage of 2D detection or feature extraction, so well established and existing approaches can be used. In the second step these proposals are then extended to 3D-detection by exploiting geometric relationships (Li et al. 2019; Mousavian et al. 2017), geometric shape matching (Chabot et al. 2017) or similar techniques.

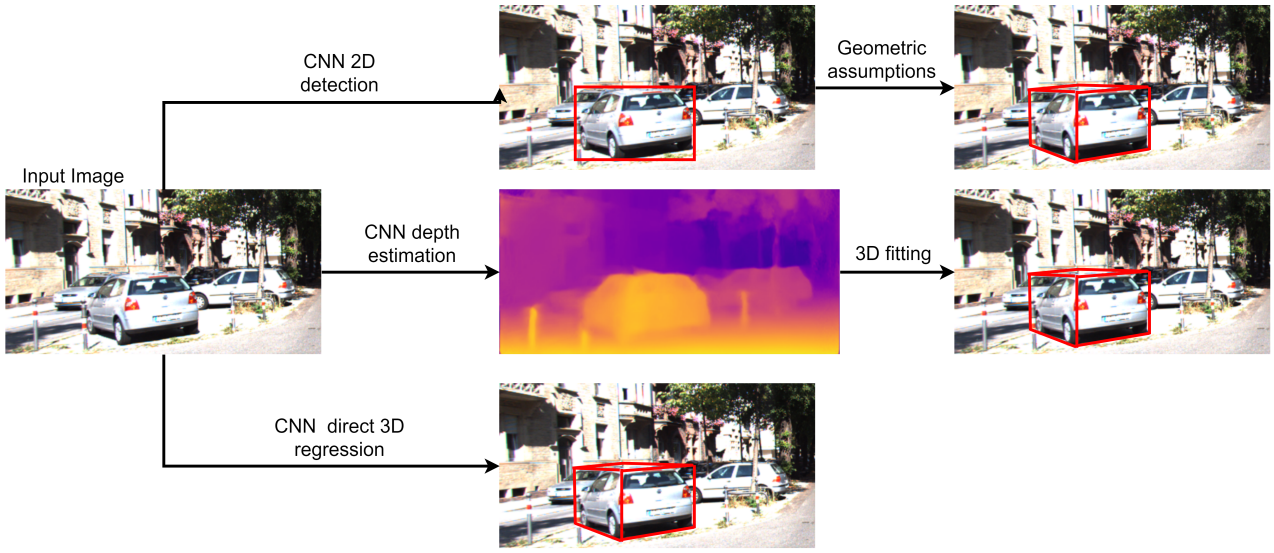


Figure 3.16: Visualization of different groups of monocular 3D camera detection. On top, multi-stage approaches, in the middle depth estimation approaches and on bottom direct 3D regression approaches are shown. The actual implementation may vary from approach to approach.

- Depth estimation approaches:
  Based on the camera image, the depth is estimated instance- (Qin et al. 2021) or pixel-wise (He and Soatto 2019) to extract 3D information for the OBB generation using a CNN. These CNNs do not directly measure the physical depth in scenes, but their ability to learn contextual features allows them to make an estimation based on visual cues. By using pixel-wise estimation, a pseudo Lidar point cloud can be generated, that provides data for point cloud based approaches (Wang et al. 2019).
- Direct 3D regression:
  This group of algorithms usually regresses the 3D OBB directly during the detection. As an example, YOLO3D (Liu et al. 2021) extends the idea of the one stage detector by directly regressing 3D OBBs instead of 2D bounding boxes. Similar to the depth estimation approaches, the distance is not directly measured, but estimated by the CNN based on the visual information available.

3.3.2 Distance estimation using ground projection

One of the main challenges is the conversion from 2D image coordinates to 3D world coordinates, which can be resolved by using geometric assumptions. A multi-stage approach with projection onto the ground plane is a simple method of estimating the distance of an object, that is detected by a well established 2D detector. It is assumed, that the camera's extrinsic calibration is known and the detected objects are always positioned on a flat ground surface. This method is shown by Rezaei et al. (2015) with a car and uses a geometric approach to project the bottom line of a 2D bounding box to the ground plane while assuming an almost planar ground surface. Rezaei et al. (2015) report, that the majority of all measurements are within a margin of ± 60 cm of distance to the ground truth for a range of 6 to 50 m. The estimation procedure is shown in Figure 3.17. The distance d to a detected object can be calculated using geometry. Based on equation (2.2), the vertical pixel coordinate v of a 3D point $\begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^T$ in the camera coordinate system as described in Figure 2.11 can be calculated by

$$sv = f_y y_c + v_0 z_c. \tag{3.10}$$

When applying the constraint $s = z_c$ of scaling factor s, v can be described by

$$v = f_y \frac{y_c}{z_c} + v_0. \tag{3.11}$$

The object projection angle θ with $\tan(\theta) = \frac{y_c}{z_c}$ can be inserted, which allows to calculate the object projection angle based on the pixel coordinate v:

$$\theta = \arctan\left(\frac{v - v_0}{f_y}\right). \tag{3.12}$$

The distance d can be calculated by $d = h_c \tan(\theta_c + \theta)$. Using the camera height h_c and the camera orientation θ_c with the lowest object point d_p and the image height in pixels h_i in $v = h_i - d_p$ results in

$$d = h_c \left[\tan\left(\theta_c + \arctan\left(\frac{h_i - d_p - v_0}{f_y}\right)\right)\right]. \tag{3.13}$$

If the camera intrinsic calibration is not available, applying the two assumptions

$$v_0 = \frac{h_i}{2} \tag{3.14}$$

and

$$f_y = \frac{h_i}{2\tan(\frac{\alpha}{2})} \tag{3.15}$$

with α being the vertical FoV, the distance can be calculated as follows:

$$d = h_c \left[\tan\left(\theta_c + \arctan\left(\frac{\frac{h_i}{2} - d_p}{\frac{h_i}{2\tan(\frac{\alpha}{2})}}\right)\right)\right]. \tag{3.16}$$

Since all other parameters are constants, the distance d relies on the vertical position d_p of the projection of the closest point on the road surface of a detected object. This point is usually defined by the lower edge of a detected 2D bounding box.

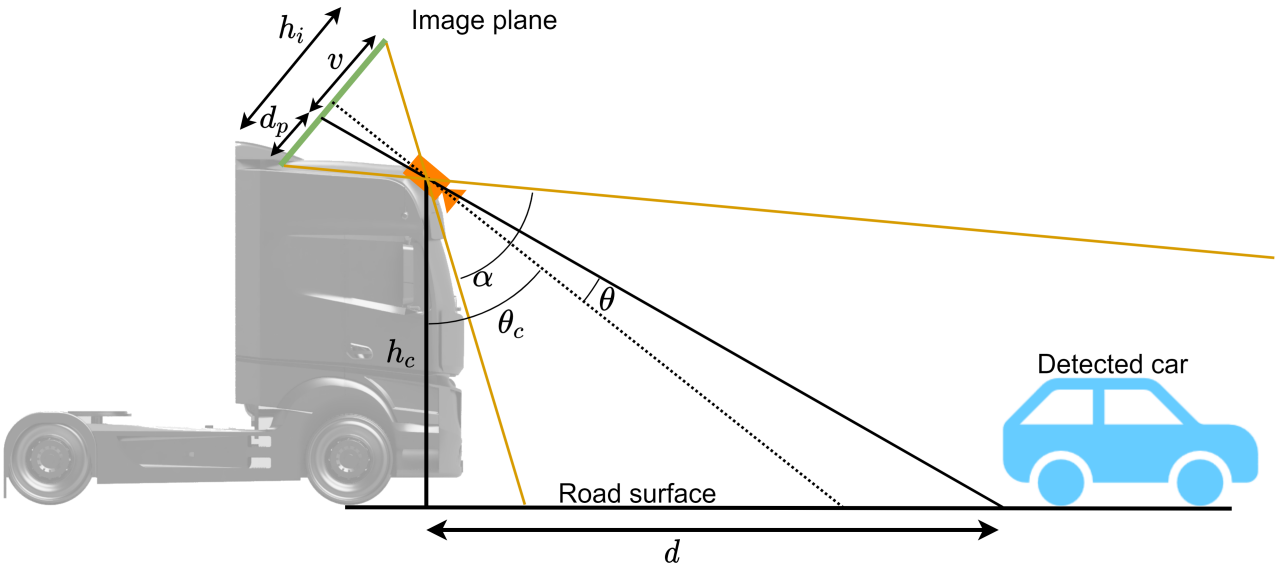


Figure 3.17: Distance estimation using the projection to the ground surface with a camera mounted to the truck cabin. Based on figure from Rezaei et al. (2015), but applied to truck.

3.3.3 Multi-stage object detection for close ranges at trucks

Monocular camera-based object detection for trucks is getting increased attention in recent years due to the increasing number of cameras built into the truck. Mirror replacement systems for rear surveillance (class II, IV after UN ECE R46/04 (UNECE 2014)) are already used by many manufacturers today due to aerodynamic reasons, but yet without perception features. In addition, more and more manufacturers move towards replacing the mirrors used for surveillance of class V and VI with cameras as well, as shown in Figure 3.18. Upcoming regulations require functions like the Moving Off Information System (MOIS), which monitors the area in front of the truck. This shows the potential of utilizing those mirror replacement camera streams for object detection as well, since the FoV overlaps with the class VI mirror area. This applies in particular, since the forward facing ADAS cameras have a blind spot directly in front of the truck, where MOIS is used. Therefore, a detection pipeline using a downward looking camera covering the area directly in front of the truck is developed. Note, that the other available mirror camera systems offer improvements for the overall perception as well, since they can boost the performance of the blind spot information system and the monitoring of rearward objects in general.

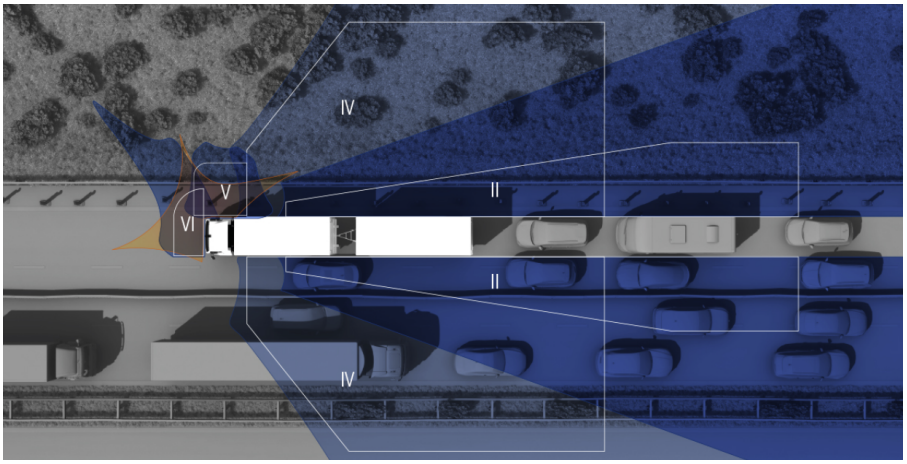


Figure 3.18: FoV areas of mirror classes for trucks following UN ECE R46/04 (UNECE 2014) projected on ground level (MEKRA Lang GmbH & Co. KG 2023).

One of the main differences of camera applications at trucks to the applications at cars is the mounting position. They are usually mounted to the cabin at greater heights, regardless of whether they face forward or backward. Such mounting positions can provide a better overview in crowded situations, are less effected by occlusion and can detect road markings in higher distance. In addition, they reach steeper angles, which has the potential to reduce errors in distance estimations. However, the cabin is also in a different, moving coordinate system compared to most other sensors, since it is suspended separately. An additional consideration is the increased pitch and roll motion compared to cars, which is amplified by the separately suspended cabin.

Based on the method shown in chapter 3.3.2 for distance estimation, an analysis is carried out to show its potential for a downward looking camera mounting position at a truck's cabin, as indicated in Figure 3.19. The cabin of trucks is subject to higher roll and pitch angles compared to passenger cars, although the high position of the camera partially compensates for this: The change in position in camera coordinates d_p while pitching is lower at greater camera heights h_c. Measurements of an Xsens MTi-100 IMU show maximum pitch values of $[-1.4°, 3.6°]$ for the cabin of a tractor while normal driving on flat roads in an urban environment without high dynamic maneuvers like emergency braking. The distance errors for these maximum pitch values are simulated based on the described projection method and shown in Figure 3.20. The simulation is carried out for distances less than 10 m, assuming a camera at height of $h_c = 3.5$ m, as shown in Figure 3.19. For close range scenarios of up to 10 m distance, the errors for maximum pitch angles are below 1.5 m with a majority of distance errors in real data expected to be lower. Due to the low errors for close ranges, the application of the

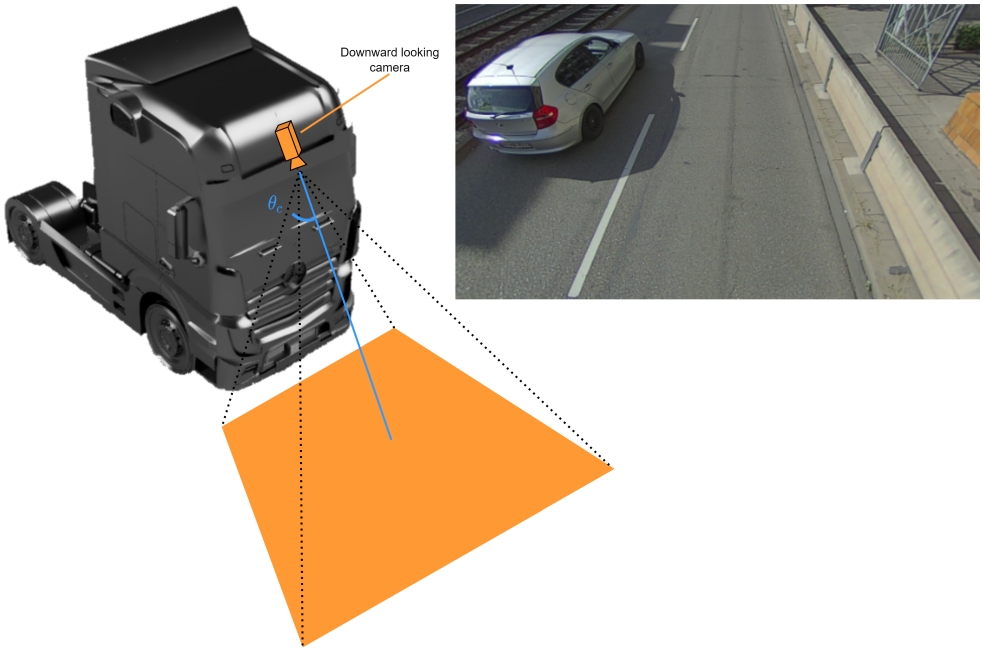


Figure 3.19: Front mirror replacement camera position and rectified example image.

described distance estimation approach to a multi-stage object detector is desirable. Especially for applications, where only limited training data is provided, the utilization of an established 2D detector in combination with a distance estimation method can produce good performances. In general, it is possible to correct those errors, if precise measurements of the cabin pitch angle are available in real-time. However, this is not always given in the current generation of trucks. Note, that such a projection method is not working for long distances, where the detection is close to the horizon.

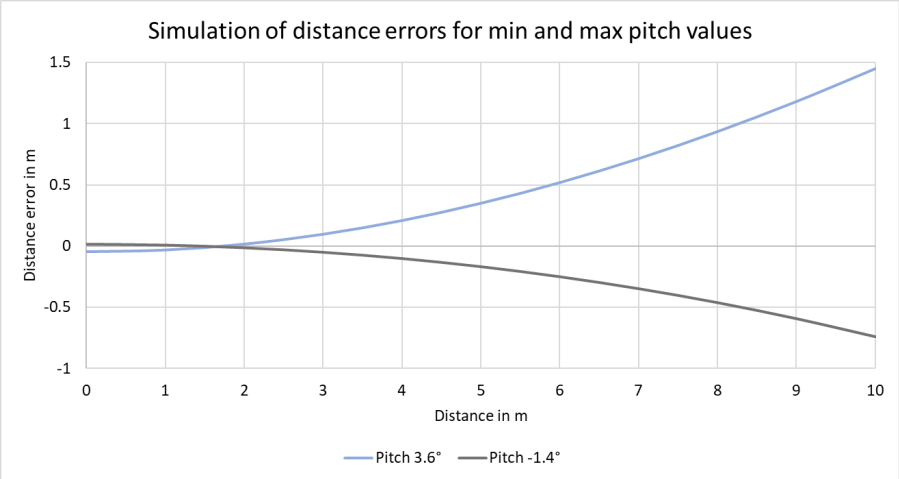


Figure 3.20: Simulation of distance estimation error using minimum and maximum pitch angle at height of $h_c = 3.5$ m.

Zhang (2021) investigates the application of state-of-the-art detection algorithms for close ranges truck applications using a camera position. Zhang (2021) uses YoloV5s (Jocher et al. 2022) for 2D object detection and a distance estimation based on the inverse perspective mapping. Based on that work, a monocular 3D object detector for 3D OBBs is developed. Similar to Zhang (2021), the first step is the detection of 2D bounding boxes in the image of the camera using YoloV5s (Jocher et al. 2022). However, the distance estimation differs

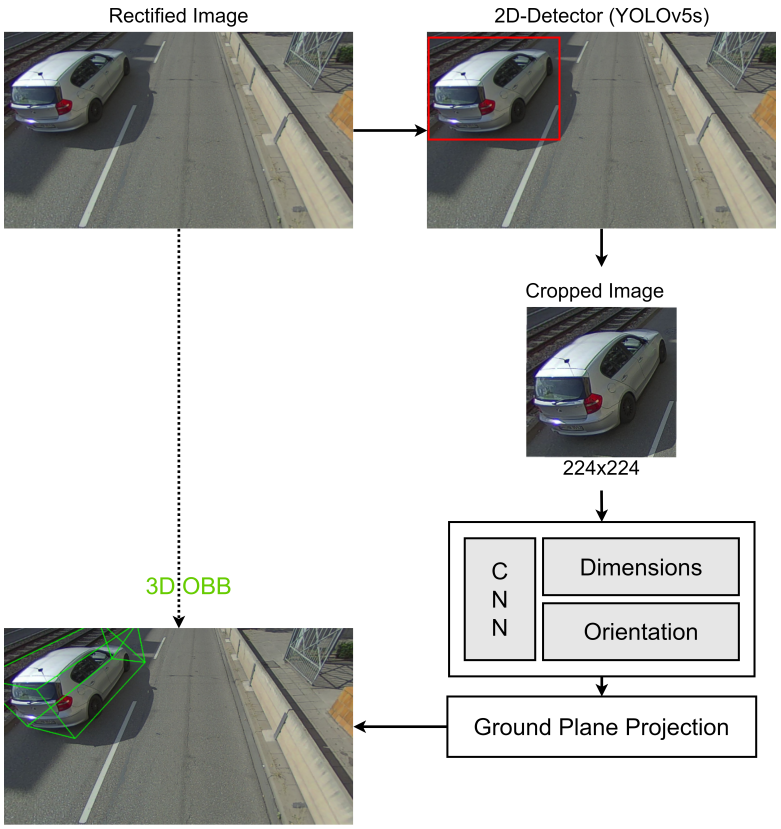


Figure 3.21: Scheme of 3D OBB generation approach: 2D bounding box detection, cropping and rescaling, CNN-based orientation and dimension estimation, geometric road surface projection.

and the dimensions and orientation of the OBB are estimated in addition, as shown in the overview in Figure 3.21. Based on the detections from the first step, small crops of the camera image of size 224x224 pixels are generated, that only contain the detected object. Next, the approach from Mousavian et al. (2017) is applied to generate a 3D OBB: A CNN is applied to the cropped image of the object to estimate the dimensions (length, width, height) and the orientation of the OBB. The position of the OBB in world coordinates is then calculated using the previously described distance estimation approach.

For the estimation of the OBB dimensions and orientation, the approach from Mousavian et al. (2017) is used. The approach uses a CNN to estimate the dimensions and orientation of objects that are previously detected. The input to the CNN are cropped areas from the detections in an image, which are first scaled to 224x224 pixels. A pretrained backbone extracts features from this image. Then, multiple branches working on these features regress the OBB dimensions and the orientation. Here, some adaptions are made to the approach of Mousavian et al. (2017) for real-time capability and modularity:

- In contrast to the work of Mousavian et al. (2017), MobileNetV2 (Sandler et al. 2018) is used as backbone, which has a faster runtime.
- Mousavian et al. (2017) also show an approach of calculating the OBB position based on the estimated dimensions and the camera intrinsic. However, in this thesis the geometric projection from equation (3.13) is used as described previously. This adaptation leads to increased performance and robustness.

In order to get the position of the OBB, both the X- and Y-Position on the road surface need to be calculated. Using the approach from equation (3.13), the distance to the closest point of a detected object can be calculated. Figure 3.22 shows the 2D bounding box of a detection in image coordinates and its projection to the road surface in BEV (right) and from the side (bottom). Here, the blue line is the closest point of a detection. Using the

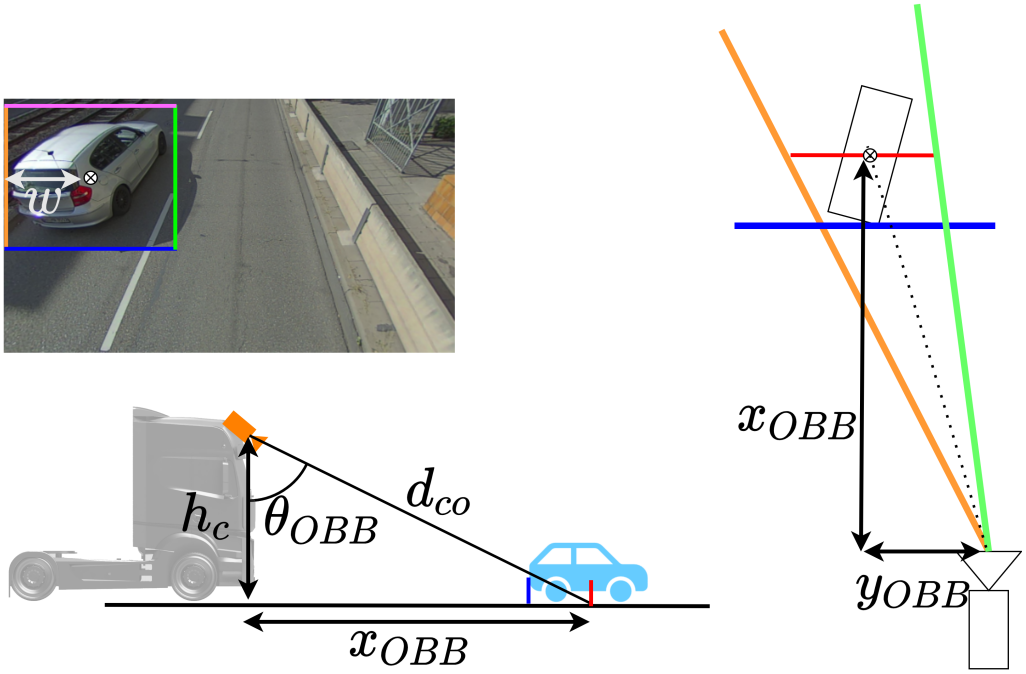


Figure 3.22: The center of the bounding box is fitted to the center of the left (orange) and right (green) 2D bounding box boarders. The closest point is fitted to the distance line (blue), which in combination defines the X- and Y-Position of the 3D OBB in vehicle coordinates

estimated dimensions and orientation from the CNN, the closest edge point of the OBB needs to match with this blue line. This allows to calculate the x-position x_{OBB} of the OBB center. It is assumed, that the OBB is centered between the left and right bounding box edge marked in orange and green. Note, that the OBB's edges do not necessarily need to have match these 2d bounding box edges, since the orientation is estimated in a separate step. The middle point w is used to calculate the OBB's Y-Position y_{OBB}.

Equation (2.2) states

$$su = f_x x_c + u_0 z_c, \tag{3.17}$$

resulting in

$$y_{OBB} = -x_c = -\frac{z_c w - z_c u_0}{f_x} \tag{3.18}$$

for the calculation of y_{OBB} when applying the constraint $s = z_c$ and $u = w$. With known distance x_{OBB}, the distance z_c can be calculated by

$$z_c = \cos(\theta_{OBB}) d_{co} \tag{3.19}$$

with the distance d_{co} between camera OBB center on ground level

$$d_{co} = \sqrt{h_c^2 + x_{OBB}^2} \tag{3.20}$$

and the OBB angle θ_{OBB}

$$\theta_{OBB} = \arctan\left(\frac{x_{OBB}}{h_c}\right). \tag{3.21}$$

After this calculation step, the position on the road surface, as well as the dimensions and the orientation of the OBB are known and the full detection pipeline is calculated using a 2D detection in combination with a CNN for dimension and orientation estimation and a geometric projection to the road surface for the calculation of the position. The approach has low errors for the distance estimation and achieves accurate AP results, which is shown in chapter 6.4.

3.4 Radar-based object detection

The radar sensors used in this work all have integrated object detection functionalities. In each time-step, the sensor detects surrounding objects and outputs a list of objects including several object properties. Usually, these objects are reported as 3D point locations with additional information about the relative speed, a classification, and a confidence level. Some sensors additionally report information about the spatial expansion, like an L-shape with length and width. Over-segmentation is possible, so multiple detections may be created for one target. An example for radar detections is shown in Figure 3.23. Here, the red, turquoise, yellow and blue markings are radar detections from different radars generating L-shapes, point objects or box objects. Internally, the detected objects may be processed and tracked over multiple time-steps. The detection and tracking process is not shown in detail here, but more detailed information can be found in the work of Gamba (2020).

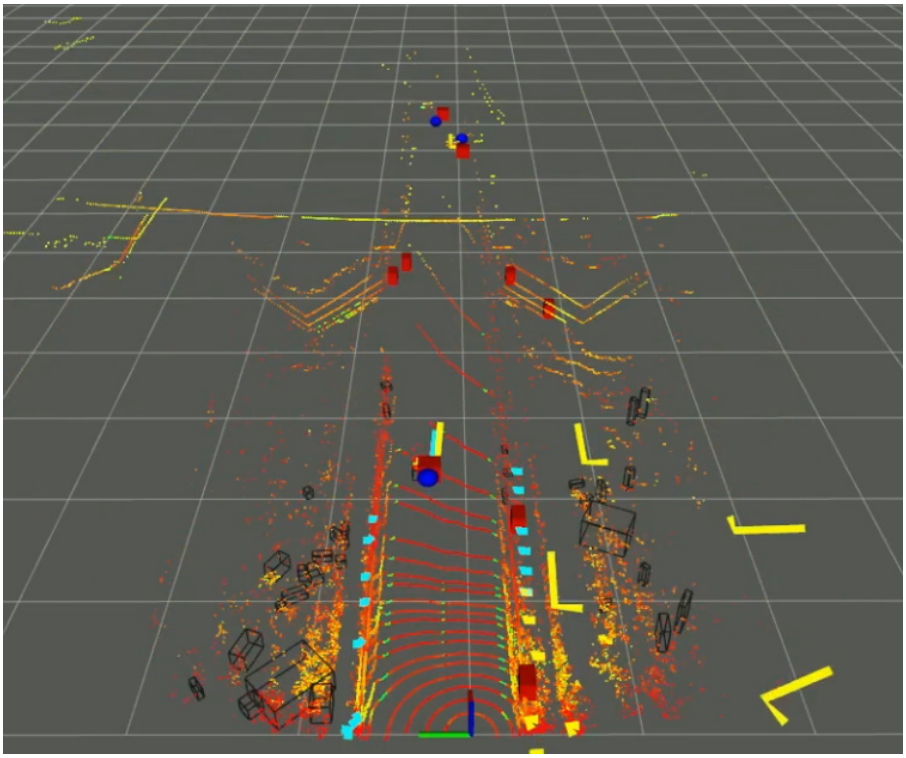


Figure 3.23: Example of radar detections from various radar sensors and a Lidar point cloud in a highway scenario. Radar detections shown in yellow, turquoise, red and blue are available as point, box or L-shape objects.

4 Basics of multi-object tracking and multi-sensor fusion

Multi-Object Tracking (MOT) is the problem of the (joint) estimation of the states of multiple objects throughout their existence. Therefore, the task of a MOT system is to determine the number of objects present in each time-step and maintain their unique identity (Luo et al. 2021). In addition, the states, like position and velocity, and thus the trajectory of each object are estimated. A tracked object is usually called "track", while an observation of an object by a sensor is usually called "measurement". A multi-object tracker estimates the track's states based on the measurements provided by sensors. In practical applications, this is often done by using a probabilistic filter to estimate the states, which is supported by some sort of track management. This track management manages the inputs and outputs of the system, the timings and possibly has additional tasks to improve the tracking performance or efficiency.

The tracks created by MOT systems is usually used as input for ADAS and automated driving functions. According to (Dietmayer 2016), perception stacks for automated driving need to deal with three uncertainty domains, which are state uncertainty (e.g. physical properties like position), existence uncertainty and class uncertainty. The state and existence uncertainty can be addressed by using a multi-object tracker framework, while the class uncertainty can be addressed by extending such a framework with additional classification estimation.

MOT is used in several domains, like tracking all sorts of objects in videos (see examples from Luo et al. (2021)) or tracking of radar echos in air traffic. In automotive applications, however, MOT is done using onboard perception sensors like Lidar, radar or cameras, with a strong focus on other traffic participants like cars, trucks, pedestrians, and cyclists. Additional input sources, like Global Navigation Satellite System (GNSS) or map data, can potentially improve the performance. In automotive applications, the MOT system faces some challenges that come along with the used sensors and hardware:

- Imperfect sensors that produce random spurious measurements, called clutter, and are often associated with the presence of real objects. It can, for example, arise from sensor imperfections or reflections.
- Imperfect sensors that miss detections of real objects.
- Different sensing strengths and weaknesses of different sensor modalities with different noise properties need to be combined.
- Changing environment conditions (illumination, weather, operational domain) that have influence on the current detection capabilities.
- Interchangeable sensor setups. It is desired to use approaches that work with multiple sensor types and sensor setups across different vehicle models.
- Real-time capability.
- Occlusion and truncation of objects.
- Interactions between objects, like pedestrians getting in or out of cars.

This chapter shows the basics of multi-object tracking, which are necessary for understanding the following chapters. However this chapter does not propose new approaches or developments beyond the literature. The described basics cover in particular the following topics:

- The definition and basics of MOT.
- The Kalman filter basics and the application of the KF for MOT frameworks.

- The random finite set basics, the GM-PHD filter description and the application of the GM-PHD filter for MOT frameworks.
- The basics for tracking the classification of a target over time.
- The basics of multi-sensor fusion in the context of MOT.

4.1 Tracking approaches

The challenges of MOT can be solved with a large variety of different approaches with different strengths and weaknesses. This includes probabilistic approaches, deterministic optimizations and neural networks.

Luo et al. (2021) classify MOT approaches based on three categories of properties: Initialization method, processing mode and type of output. The initialization method is divided into "detection-based tracking", where the tracking process links detection hypotheses from each time-step to trajectories and "detection-free tracking", where the localization and tracking is conducted simultaneously. The processing mode is divided into "online tracking", where the data is processed sequentially in each time-step and "offline tracking", where the approach jointly processes a whole time series. The type of output is divided into "stochastic tracking", that may vary between two runtimes and "deterministic tracking", that always creates the same results when running multiple times. An example of "stochastic tracking" is the usage of particle filters for inference (Luo et al. 2021), which will lead to slightly different results, "deterministic tracking" does not change between multiple runs.

The most common types of trackers used in automotive applications are based on probabilistic filters, that are suitable for online processing and can be used in detection-based frameworks, like the Kalman filter. Therefore, based on the categorization of Luo et al. (2021), they are "detection based", "online" and "deterministic" tracking approaches. However, in the last few years also a rising amount of research has been conducted on tracking algorithms based on neural networks. These networks often use recurrent network techniques, which are capable of evolving through time. Milan et al. (2016) and Holz (2023) for example, use recurrent neural networks in an architecture inspired by Bayesian filters to perform the steps of prediction, update, and data association within the network.

Since detection-based deterministic online trackers are well established in automotive applications and the deterministic properties offer great arguments in terms of functional safety, the thesis focuses on this type of tracker.

In this thesis, MOT is an estimation problem of a set of states $X_k = \{\boldsymbol{x}_{k,1}, \boldsymbol{x}_{k,2}, ..., \boldsymbol{x}_{k,n_k}\}$ at time-step k, where $\boldsymbol{x}_{k,i}$ is the state of the i-th object. In detection-based tracking frameworks, there is a corresponding measurement set $Z_k = \{\boldsymbol{z}_{k,1}, \boldsymbol{z}_{k,2}, ..., \boldsymbol{z}_{k,m_k}\}$ for each time-step k, where $\boldsymbol{z}_{k,j}$ is the j-th measurement. MOT approaches try to maximize the a posteriori function of the conditional distribution of the states given the measurements, in order to find the optimal sequence of states for each tracked object (Luo et al. 2021):

$$\hat{X}_{1:k} = \underset{X_{1:k}}{\arg\max}\, P\left(X_{1:k}|Z_{1:k}\right). \tag{4.1}$$

For better understanding, however, single target tracking using a Bayesian filter is explained here first.

4.2 Bayesian filters

Due to several sources of errors, states are usually tracked using probability distributions that can reflect the amount of uncertainty. The Bayesian theory (Bayes 1763) is the fundamental basis in many filtering applications, since it describes the calculation of a posterior Probability Density Function (PDF) of a state $\boldsymbol{x}_k$ using

the prior and the likelihood of measurements given a series of states $\boldsymbol{x}_{0:k} = \boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_k$ and a series of observations $\boldsymbol{z}_{1:k} = \boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_k$:

$$\overbrace{p(\boldsymbol{x}_{0:k}|\boldsymbol{z}_{1:k})}^{Posterior} = \frac{\overbrace{p(\boldsymbol{z}_{1:k}|\boldsymbol{x}_{0:k})}^{Likelihood}\overbrace{p(\boldsymbol{x}_{0:k})}^{Prior}}{\underbrace{p(\boldsymbol{z}_{1:k})}_{Normalization}}. \tag{4.2}$$

When applying the Markov assumption, that each state $\boldsymbol{x}_k$ only depends on the previous step $\boldsymbol{x}_{k-1}$ and the assumption that each measurement $\boldsymbol{z}_k$ only depends on the current state $\boldsymbol{x}_k$, a recursive solution can be found (Challa et al. 2011; Schreier 2015; Bar-Shalom et al. 2001):

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) = \underbrace{\frac{\overbrace{p(\boldsymbol{z}_k|\boldsymbol{x}_k)}^{Likelihood}}{p(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})}}_{Normalization} \underbrace{\int_{\boldsymbol{x}_{k-1}} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1})\, d\boldsymbol{x}_{k-1}}_{Prior(Chapman-Kolmogorov)}. \tag{4.3}$$

This equation is solved by Bayesian filters by dividing it into two steps. The Chapman-Kolmogorov integral is the prediction step with the transition density $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$. The update step corrects the prior PDF using the likelihood to calculate the new posterior PDF. To ensure a correct PDF distribution, it is normalized by

$$p(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1}) = \int p(\boldsymbol{z}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})\, d\boldsymbol{x}_k. \tag{4.4}$$

4.3 Kalman filter

The Kalman Filter (KF) (Kalman 1960) can be used to track one object defined by multiple states over time and estimate the states and its uncertainties. It is an optimal Bayesian filter for estimating states of linear, Gaussian distributed systems. Therefore, it needs a linear state-space model describing the system dynamics and a linear observation model describing the measurements. The equations and notation used are based on the work of Scheider (2021), with some changes made for consistency with the other chapters. The general description of a time-discrete state space system

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{w}_k \tag{4.5}$$

is reduced to the linear time-invariant system for the application of the KF:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{C}\boldsymbol{w}_k \tag{4.6}$$

Here, $f(\boldsymbol{x}_k, \boldsymbol{u}_k)$ is the state transition function of the state vector $\boldsymbol{x}_k$ and the control vector $\boldsymbol{u}_k$ at time-step k. In the linear system, it is substituted by the dynamic matrix $\boldsymbol{A}$, the control matrix $\boldsymbol{B}$ and noise matrix $\boldsymbol{C}$. The state experiences mutually independent white Gaussian noise $\boldsymbol{w}_k$.

Using covariance propagation, the covariance $\boldsymbol{\Sigma}_{xx,k+1}$ of the state $\boldsymbol{x}_k$ can be calculated by

$$\boldsymbol{\Sigma}_{xx,k+1} = \boldsymbol{A}\boldsymbol{\Sigma}_{xx,k}\boldsymbol{A}^T + \boldsymbol{B}\boldsymbol{\Sigma}_{uu,k}\boldsymbol{B}^T + \boldsymbol{C}\boldsymbol{\Sigma}_{ww,k}\boldsymbol{C}^T. \tag{4.7}$$

Here, $\boldsymbol{\Sigma}_{uu,k}$ is the covariance matrix of the control and $\boldsymbol{\Sigma}_{ww,k}$ is the process noise covariance matrix.

The state estimation of the KF is done in two consecutive steps, which are called prediction and update. Figure 4.1 shows a scheme of the KF's calculation steps through multiple time-steps. A detailed description and derivation of the equations can be found in the works of Kalman (1960) and Challa et al. (2011); Bar-Shalom et al. (2001).

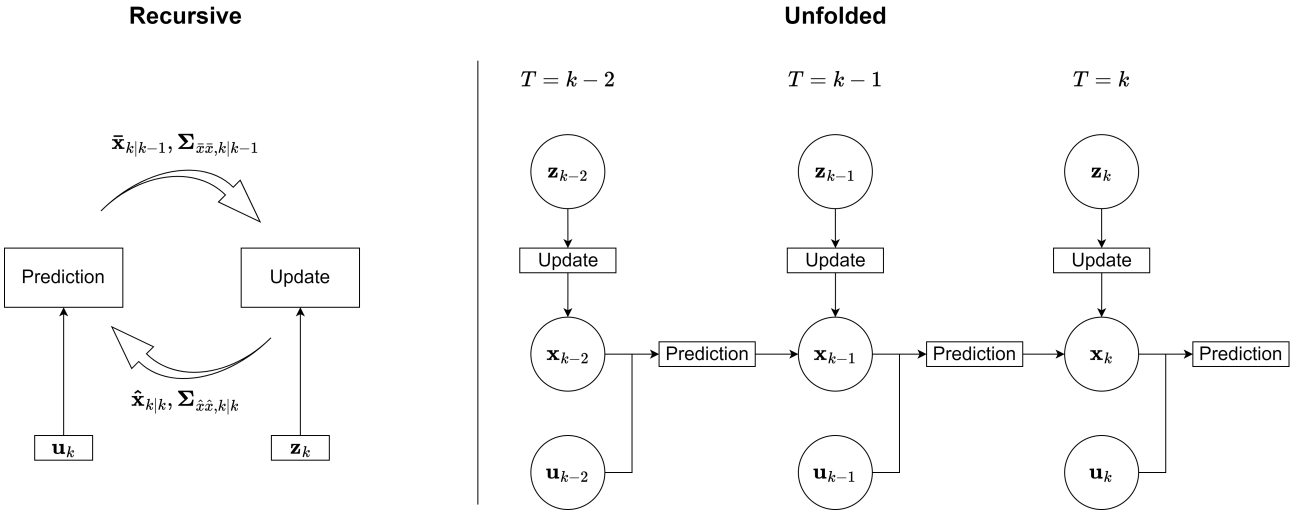


Figure 4.1: Scheme of Kalman filter calculation steps. Shown recursive on the left side and unfolded on the right side.

Prediction: During the prediction step, the predicted state $\bar{\boldsymbol{x}}_{k|k-1}$ and covariance $\mathbf{\Sigma}_{\bar{x}\bar{x},k|k-1}$ for the current time-step k are calculated based on the posterior state vector $\hat{\boldsymbol{x}}_{k-1|k-1}$ and covariance $\mathbf{\Sigma}_{\hat{x}\hat{x},k-1|k-1}$ of the previous time-step $k-1$. With the expected value $E(\boldsymbol{w}_k) = 0$ and $\boldsymbol{C} = \boldsymbol{I}$, the following equations are derived for the prediction:

$$\begin{aligned} \bar{\boldsymbol{x}}_{k|k-1} &= \boldsymbol{A}_{k-1}\hat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{B}\boldsymbol{u}_{k-1}, \\ \mathbf{\Sigma}_{\bar{x}\bar{x},k|k-1} &= \boldsymbol{A}_{k-1}\mathbf{\Sigma}_{\hat{x}\hat{x},k-1|k-1}\boldsymbol{A}_{k-1}^T + \boldsymbol{B}_{k-1}\mathbf{\Sigma}_{uu,k-1}\boldsymbol{B}_{k-1}^T + \mathbf{\Sigma}_{ww,k-1}. \end{aligned} \tag{4.8}$$

Update: The update step then corrects the predicted state and covariance estimation using the measurement model

$$\boldsymbol{z}_k = \boldsymbol{H}\hat{\boldsymbol{x}}_k + \boldsymbol{v}_k. \tag{4.9}$$

with the measurement $\boldsymbol{z}_k$, the measurement matrix $\boldsymbol{H}$ and the measurement noise $\boldsymbol{v}_k$. With $E(\boldsymbol{v}_k) = 0$, the innovation $\boldsymbol{d}_k$ between measurement and predicted measurement and the corresponding innovation covariance $\mathbf{\Sigma}_{dd,k}$ can be calculated using the measurement covariance matrix $\mathbf{\Sigma}_{zz,k}$. This is required to calculate the Kalman gain $\boldsymbol{K}_k$.

$$\begin{aligned} \boldsymbol{d}_k &= \boldsymbol{z}_k - \boldsymbol{H}\bar{\boldsymbol{x}}_{k|k-1}, \\ \mathbf{\Sigma}_{dd,k} &= \boldsymbol{H}\mathbf{\Sigma}_{\bar{x}\bar{x},k|k-1}\boldsymbol{H}^T + \mathbf{\Sigma}_{zz,k}, \\ \boldsymbol{K}_k &= \mathbf{\Sigma}_{\bar{x}\bar{x},k|k-1}\boldsymbol{H}_k^T\mathbf{\Sigma}_{dd,k}^{-1}. \end{aligned} \tag{4.10}$$

Next, the posterior state vector and covariance are calculated using the prior, the innovation, and the Kalman gain:

$$\begin{aligned} \hat{\boldsymbol{x}}_{k|k} &= \bar{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\boldsymbol{d}_k, \\ \mathbf{\Sigma}_{\hat{x}\hat{x},k|k} &= \mathbf{\Sigma}_{\bar{x}\bar{x},k|k-1} - \boldsymbol{K}_k\mathbf{\Sigma}_{dd,k|k-1}\boldsymbol{K}^T. \end{aligned} \tag{4.11}$$

In practical implementations, Josephs form of the covariance correction should be used because it is numerically more stable, since it will not lead to negative eigenvalues and ensures the covariance matrix to be always positive semidefinite (Bar-Shalom et al. 2001):

$$\mathbf{\Sigma}_{\hat{x}\hat{x},k|k} = (I - \boldsymbol{K}_k\boldsymbol{H})\mathbf{\Sigma}_{\bar{x}\bar{x},k|k-1}(I - \boldsymbol{K}_k\boldsymbol{H})^T + \boldsymbol{K}_k\mathbf{\Sigma}_{zz,k}\boldsymbol{K}_k^T. \tag{4.12}$$

The Kalman gain defines the amount of correction during the update step. The filter properties of the KF are mainly defined by the two covariance matrices $\boldsymbol{\Sigma}_{ww,k}$ and $\boldsymbol{\Sigma}_{zz,k}$. When selecting a high process noise $\boldsymbol{\Sigma}_{ww,k}$, the Kalman gain will be high, resulting in a high correction by the measurements. Using a high measurement noise $\boldsymbol{\Sigma}_{zz,k}$ results in a low Kalman gain and thus results in a small correction by the measurements.

Note, that a large variety of multi-object tracking literature uses different symbols for the covariances compared to the ones used in this introduction. To align with these notations, in the following parts, $\boldsymbol{Q}_k$ will be used for the process noise covariance matrix $\boldsymbol{\Sigma}_{ww,k}$, $\boldsymbol{R}_k$ will be used for the measurement covariance matrix $\boldsymbol{\Sigma}_{zz,k}$ and $\boldsymbol{S}_k$ will be used for the innovation covariance matrix $\boldsymbol{\Sigma}_{dd,k}$. The state covariance matrix $\boldsymbol{\Sigma}_{xx,k|k}$ will be represented as $\boldsymbol{P}_{k|k}$ from now on.

4.3.1 Kalman filter extensions

There are several extensions of the Kalman filter that can be applied to non-linear systems. Two popular versions are presented here.

The Extended Kalman Filter (EKF) (Anderson and Moore 1979; Challa et al. 2011; Bar-Shalom et al. 2001) is applicable to systems with nonlinear state transition and measurement functions $f(\boldsymbol{x}_k, \boldsymbol{u}_k)$ and $h(\boldsymbol{x}_k)$ by approximating the dynamic matrix $\boldsymbol{A}_k$ and $\boldsymbol{H}_k$ at each time-step using the Taylor-series. These are called the Jacobians. The prior state and the innovation can be calculated using the nonlinear functions, while the covariances are calculated using the Jacobians. A detailed description of the calculation steps is given by Challa et al. (2011).

The Unscented Kalman Filter (UKF) (Julier and Uhlmann 1997; Julier 2002) is another non-linear version of the KF, but it does not need to calculate the Jacobians. Instead, it uses the unscented transformation that utilizes multiple sigma points to estimate the covariances. This often leads to higher accuracies compared to the KF and EKF, while still having a low computational complexity.

4.3.2 Kalman filter in MOT

The KF is a state estimator that is frequently used in object tracking (Bader 2019; Kampker et al. 2018; Wu et al. 2022; Mobus and Kolbe 2004; Chiu et al. 2020; Himmelsbach et al. 2009; Schueler et al. 2012). In MOT, it is used to estimate the states, like position, velocity, orientation, or dimensions of tracks based on measurements from perception sensors. The states of each target are estimated using a separate KF. Therefore, the application of KF to MOT is essentially an extension of single target tracking and requires a track management and data association for proper operation. Figure 4.2 shows the necessary calculation steps and components that are usually done for MOT with KF. In each time-step, the elements of a measurement set $Z_k = \{\boldsymbol{z}_{k,1}, \ldots, \boldsymbol{z}_{k,m}\}$ are associated to existing tracks, where $\boldsymbol{z}_{k,m}$ is the measured state of the m-th detection. The existing tracks can be forwarded as a set of tracks $X_k = \{\boldsymbol{x}_{k,1}, \ldots, \boldsymbol{x}_{k,n}\}$ to the ADAS applications.

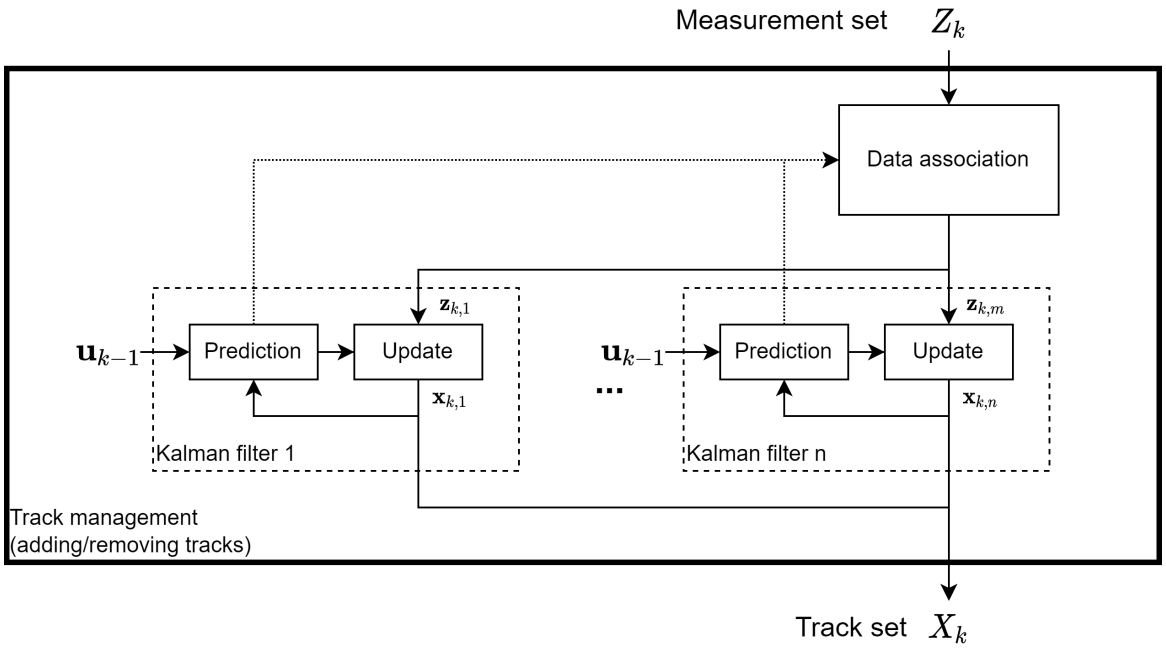


Figure 4.2: Scheme of a KF in an MOT framework including track management and data association.

4.3.2.1 Data association

In many MOT applications, each of the objects is tracked using a separate KF, so each track needs to be updated with a separate measurement. Therefore, the Data Association (DA) of measurements to the existing tracks is a key issue. Although some approaches use additional information, like appearance models (Wu et al. 2022), in this thesis only the measurement states $\boldsymbol{z}_{k,i}$ are considered for DA.

The DA process starts with the gating based on the Chi-square test, which is a pre-selection of possible measurements for each track that lie within a hyper-ellipsoid based on the innovation covariance (Challa et al. 2011). This gate region V_k is defined by the innovation $\boldsymbol{d}_k$, the innovation covariance $\boldsymbol{S}_k$ and a distance threshold γ_{DA} (Bar-Shalom and Li 1995; Challa et al. 2011):

$$V_k(\gamma_{DA}) = \{\boldsymbol{z}_k | D_M(\boldsymbol{z}_k)^2 \leq \gamma_{DA}\}. \tag{4.13}$$

The Mahalanobis distance D_M is defined as

$$D_M(\boldsymbol{z}_k) = \sqrt{\boldsymbol{d}_k^T \boldsymbol{S}_k^{-1} \boldsymbol{d}_k}. \tag{4.14}$$

The squared Mahalanobis distance follows a Chi-square distribution, so the gate can be derived based on the inverse Chi-square cumulative distribution using a chosen gate probability $P_{DA} = P\left(\boldsymbol{z} \in V_k(\gamma_{DA})\right)$ (Schreier 2015). The gating has similarities to an innovation test for detecting erroneous measurements, like described by Yu et al. (2021) for the detection of abrupt fault in GNSS systems.

After the gating, several approaches can be utilized to solve the data association problem. The Nearest Neighbor Data Association (NNDA) and its global extension, the Global Nearest Neighbor Data Association (GNNDA) (Konstantinova et al. 2003) are among the most popular ones. They carry out a hard data association, whereby always a single measurement is assigned to a single track. The Probabilistic Data Association Filter (PDAF) (Shalom et al. 2009) and the Joint Probabilistic Data Association Filter (JPDAF) (Shalom et al. 2009) on the other side use soft association that calculates the probability of associations between measurements and

tracks. Therefore, the update step is done using a weighted combination of multiple measurements, which can be beneficial in cluttered situations. A Multi Hypothesis Tracker (MHT) (Blackman 2004) uses hard associations, but keeps track of multiple association hypotheses over time. This can lead to the optimal solution, but the computational costs are significant. The mentioned approaches are summarized in Table 4.1.

In recent years, the increasing success of NNs led to an increased number of publications of NNs for data association tasks. Some of these approaches use Long Short-Term Memory (LSTM) modules for operation over multiple time-steps. These techniques are not discussed in detail here, but (Rakai et al. 2022) gives an overview of some currently available methods.

Table 4.1: Comparison of DA filters.

	NNDA	GNNDA	PDAF	JPDAF	MHT
Association	1-1 (hard)	n-m (hard)	1-m (soft)	n-m (soft)	n-n (hard)
Calculation complexity	++	+	+	-	–
Accuracy under clutter	-	-	+	++	++

In this thesis, the GNNDA approach is used, since it is simple and fast in computation and offers a hard association. Such a hard association offers good results under the assumptions that the detection sensors and algorithms detect one object per track and do not create a lot of clutter measurements. Compared to the NNDA approach, a global optimum is found, which makes it the most common choice for MOT systems with Kalman filters. Since the GNNDA is used in this thesis, the other approaches are not described in detail.

The GNNDA tries to minimize the sum of the costs of all associations, where the cost of associating track i with measurement j is $c_{ij} = D_{M,i}(\boldsymbol{z}_{k,j})$. All costs c_{ij} of the association of n tracks to m measurements are summarized in the cost matrix $\boldsymbol{C}^{nxm}$. Now, the linear association problem can be solved using the Hungarian algorithm (Kuhn 1955) or similar approaches. In order to save computational effort, it is possible to create association clusters, where the gates overlap. This divides the cost matrix $\boldsymbol{C}$ into multiple sub-problems, resulting in increased computation speed. The whole association problem with two clusters is shown in Figure 4.3. It is obvious, that a proper dynamic model and proper noise covariance matrices are necessary to obtain correct gates.

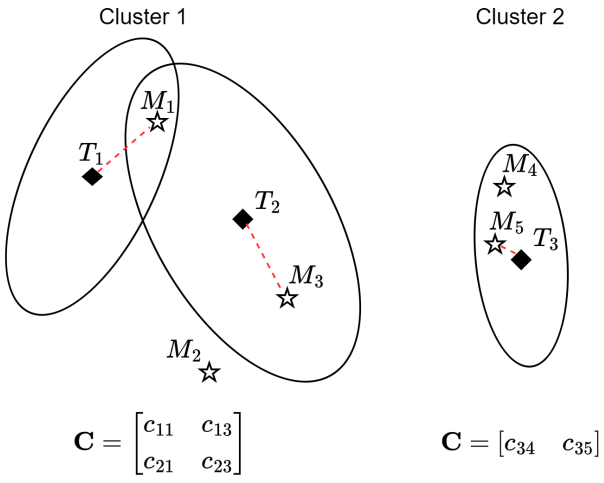


Figure 4.3: Example of data association and gating with two clusters, 3 tracks T and 5 measurements M. In the example, two clusters are generated. The first cluster contains T_1, T_2, M_1 and M_3, while the second cluster contains T_3, M_4 and M_5. M_2 is not located in either cluster and therefore cannot be assigned to any track. The result of the association is indicated by red dotted lines.

4.3.2.2 Track management

All logical operations and rules, that are necessary for the operation of a multi-object tracker are summarized here as track management. The main task of the track management is to keep the list of all tracked objects updated, add new tracks from measurements and remove ones that are not any more existent.

Heuristic approaches that use a state machine for each track can be used. Typical states are "initializing", "tracking" and "drifting" (Kampker et al. 2018) or similar, where the transition rules are based on indicators like the number of consecutive updates. New tracks typically are added with the state "initializing", if a measurement cannot be associated to an existing track. The main challenge is to find appropriate transition rules that represent the reality of appearing and disappearing objects.

An alternative to state machines is the calculation of the existence probability. If this probability drops below a threshold, the tracks are deleted. This can be done using integrated PDAF (Musicki et al. 1994) or JPDAF (Musicki and Evans 2004) filters, that incorporate the existence probability. For hard association filters, the existence probability can also be modeled as a Bayesian formulation, where the existence probability is a two-state Markov process, as shown by Aeberhard (2017). Within this framework, the existence probability $p(\exists \boldsymbol{x}_k|Z_k)$ and the non-existence probability $p(\nexists \boldsymbol{x}_k|Z_k)$ of a track $\boldsymbol{x}_k$ with $p(\exists \boldsymbol{x}_k|Z_k) = 1 - p(\nexists \boldsymbol{x}_k|Z_{k-1})$ are estimated over time. This results in the prediction step

$$\begin{aligned} p(\exists \boldsymbol{x}_k|Z_{k-1}) &= p_p(\hat{\boldsymbol{x}}_{k|k-1})p(\exists \boldsymbol{x}_{k-1}|Z_{k-1}) + p_b(\hat{\boldsymbol{x}}_{k|k-1})(\nexists \boldsymbol{x}_{k-1}|Z_{k-1}), \\ p(\nexists \boldsymbol{x}_k|Z_{k-1}) &= [1 - p_p(\hat{\boldsymbol{x}}_{k|k-1})]p(\exists \boldsymbol{x}_{k-1}|Z_{k-1}) + [1 - p_b(\hat{\boldsymbol{x}}_{k|k-1})](\nexists \boldsymbol{x}_{k-1}|Z_{k-1}), \end{aligned} \tag{4.15}$$

with the persistence probability of a track p_p and the birth probability p_b that is also used for initialization. In the update step, the existence probability is corrected depending on a successful data association. If a measurement is associated to the track, the existence update is as follows:

$$\begin{aligned} p(\exists \boldsymbol{x}_k|Z_k) &= \eta p(\boldsymbol{z}_k|\exists \boldsymbol{x}_k)p(\exists \boldsymbol{x}_k|Z_{k-1}), \\ p(\nexists \boldsymbol{x}_k|Z_k) &= \eta p(\boldsymbol{z}_k|\nexists \boldsymbol{x}_k)p(\nexists \boldsymbol{x}_k|Z_{k-1}), \\ \eta &= [p(\boldsymbol{z}_k|\exists \boldsymbol{x}_k)p(\exists \boldsymbol{x}_k|Z_{k-1}) + p(\boldsymbol{z}_k|\nexists \boldsymbol{x}_k)p(\nexists \boldsymbol{x}_k|Z_{k-1})]^{-1}. \end{aligned} \tag{4.16}$$

Here, η is the normalization factor. If no measurement is associated to a track, the update changes to

$$\begin{aligned} p(\exists \boldsymbol{x}_k|Z_k) &= \eta p(\overline{\boldsymbol{z}}_k|\exists \boldsymbol{x}_k)p(\exists \boldsymbol{x}_k|Z_{k-1}), \\ p(\nexists \boldsymbol{x}_k|Z_k) &= \eta p(\overline{\boldsymbol{z}}_k|\nexists \boldsymbol{x}_k)p(\nexists \boldsymbol{x}_k|Z_{k-1}), \\ \eta &= [p(\overline{\boldsymbol{z}}_k|\exists \boldsymbol{x}_k)p(\exists \boldsymbol{x}_k|Z_{k-1}) + p(\overline{\boldsymbol{z}}_k|\nexists \boldsymbol{x}_k)p(\nexists \boldsymbol{x}_k|Z_{k-1})]^{-1}. \end{aligned} \tag{4.17}$$

$p(\boldsymbol{z}_k|\exists \boldsymbol{x}_k)$ is the probability that the measurement is associated given the object exists, which can be interpreted as the detection probability $p_{D,k}$. $p(\boldsymbol{z}_k|\nexists \boldsymbol{x}_k)$ therefore is the probability of the measurement being clutter $p_{c,k}$ resulting in

$$\begin{aligned} p(\boldsymbol{z}_k|\exists \boldsymbol{x}_k) &= p_{D,k}, \\ p(\boldsymbol{z}_k|\nexists \boldsymbol{x}_k) &= p_{c,k}, \\ p(\overline{\boldsymbol{z}}_k|\exists \boldsymbol{x}_k) &= 1 - p_{D,k}, \\ p(\overline{\boldsymbol{z}}_k|\nexists \boldsymbol{x}_k) &= 1 - p_{c,k}. \end{aligned} \tag{4.18}$$

This can be further extended by using a generalized Bayes formulation (Aeberhard 2017) that incorporates L time-steps to create more robust estimations with sporadic missing associations:

$$\begin{aligned} p(\boldsymbol{z}_k|Z_{k-L:k-1}, \exists \boldsymbol{x}_k) &= \frac{1}{L+1} \sum_{i=k-L}^{k} p_{D,i}, \\ p(\boldsymbol{z}_k|Z_{k-L:k-1}, \nexists \boldsymbol{x}_k) &= \frac{1}{L+1} \sum_{i=k-L}^{k} p_{c,i}, \\ p(\overline{\boldsymbol{z}}_k|Z_{k-L:k-1}, \exists \boldsymbol{x}_k) &= \frac{1}{L+1} \sum_{i=k-L}^{k} 1 - p_{D,i}, \\ p(\overline{\boldsymbol{z}}_k|Z_{k-L:k-1}, \nexists \boldsymbol{x}_k) &= \frac{1}{L+1} \sum_{i=k-L}^{k} 1 - p_{c,i}. \end{aligned} \tag{4.19}$$

The parameters p_p, p_b, $p_{D,k}$ and $p_{c,k}$ can be modeled depending on multiple variables, like the position of the measurement and for each sensor independently. A detailed description of the existence estimation process is shown by Aeberhard (2017).

4.4 Random finite set filters

As discussed in the previous chapters, the most common solution for MOT is extending a single target tracker, like the KF, to multiple targets using data association and a track management. Despite its popularity, this is a non-optimal solution. According to Ristic (2013), there are several situations, that are not covered intrinsically:

- Tracking of multiple objects: Each track needs a separate filter instance. This introduces the challenge of data association.
- Switching targets: In MOT applications, multiple targets may appear and disappear through time.
- Clutter and missed detections: Imperfect measurements, like false or missed detections, are not considered in the standard KF formulation.

These challenges are not covered by the KF directly, but need to be handled separately using data association techniques and track management algorithms based on heuristic rules or estimations. However, this introduces heuristics and approximations that are not ideal in a probabilistic way. The DA can introduce major errors if measurements are associated incorrectly. The track management may introduce additional errors if the heuristic rules for adding or removing tracks are set up poorly. In addition, while these approaches "[...] provide an idea of the number of objects present, they do not have explicit modeling to estimate the object number" (Challa et al. 2011).

Mahler (Mahler 2004) introduced the Finite-Set Statistics (FISST) which resulted in Random Finite Set (RFS) filters, that intrinsically cover these situations. From this formulation, several filters emerged that are well suited for MOT applications, which are believed to be able to outperform KF-based and deep learning approaches (Pang and Radha 2021). The main reason for that believe is, that the modeling of the unknown number of states, their appearing and disappearing and the data association is not covered by the KF itself. RFS filters are more flexible and robust in handling these complexities of MOT scenarios, such as varying object numbers, data association challenges, clutter and missed detections.

Even though there is a lot of evidence for the benefits of RFS filters, a gap exists in the literature regarding the comparison to KF approaches on real-world automotive data. Most existing comparisons use simulated data

and do not include a direct comparison to the KF. The experiments conducted in this thesis address this gap in the literature.

4.4.1 Random finite sets

Mahler (2007b) introduced the RFS, which is a set of states where both the number of states and the states themselves are random and time varying. With n_k targets being present at time-step k and with their states being $\boldsymbol{x}_{k,1}, \ldots, \boldsymbol{x}_{k,n_k} \in \mathcal{X}$ in the state-space $\mathcal{X}$, an RFS is defined as follows:

$$X_k = \{\boldsymbol{x}_{k,1}, \ldots, \boldsymbol{x}_{k,n_k}\} \in \mathcal{F}(\mathcal{X}). \tag{4.20}$$

$\mathcal{F}(\mathcal{X})$ is the collection of all finite subsets of $\mathcal{X}$ (Vo and Ma 2006). Using this representation, the number of tracked objects can be modeled using a point process model, while the targets states are modeled using a motion model (Challa et al. 2011). In this notation, the RFS is unordered, which means that $X_k = \{\boldsymbol{x}_{k,1}, \boldsymbol{x}_{k,2}\} = \{\boldsymbol{x}_{k,2}, \boldsymbol{x}_{k,1}\}$. The cardinality is the discrete distribution $\rho(n) = \mathrm{P}\{|X| = n\}$ with $n \in \mathbb{N}$ and is calculated as follows:

$$\rho(n) = \mathrm{P}\{|X| = n\} = \frac{1}{n!} \int f(\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\})\, d\boldsymbol{x}_1, \ldots, d\boldsymbol{x}_n. \tag{4.21}$$

The PDF of an RFS is now completely described by the cardinality $\rho(n)$ and a family of joint probability densities $f_n(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ (Ristic 2013):

$$f(X) = f(\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}) = n! \cdot \rho(n) \cdot f_n(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n). \tag{4.22}$$

Here, the cardinality distribution models the number of objects, while for each cardinality, the according probability density models the distribution of the set elements. This multi state PDF integrates to 1 using the set integral formulation from (Mahler 2007b):

$$1 = \int f(X)\, \delta X = f(\emptyset) + \sum_{i=0}^{\infty} \frac{1}{i!} \int_{\mathbb{X}^i} f(\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i\})\, d\boldsymbol{x}_1, \ldots, d\boldsymbol{x}_i. \tag{4.23}$$

The first order statistical moment is called Probability Hypothesis Density (PHD) or intensity function. The PHD $v(\boldsymbol{x})$ is the random set variable's equivalent to the expected value of random variables and defined by

$$v(\boldsymbol{x}) = \mathbb{E}\{\delta_X(\boldsymbol{x})\} = \int \delta_X(\boldsymbol{x}) f(X)\, \delta X. \tag{4.24}$$

This is based on the set Dirac delta function $\delta_X(\boldsymbol{x}) = \sum_{w \in X} \delta_w(\boldsymbol{x})$ with the Dirac delta function $\delta_w(\boldsymbol{x})$ concentrated at w. The PHD $v(\boldsymbol{x}_0)$ can be interpreted as the density of expected number of targets at $\boldsymbol{x}_0$. Therefore, the integral over a region S is the expected number of targets in S.

IID RFS For Independent Identically Distributed (IID) cluster RFS, the probability distribution $\rho(n)$ models the cardinality, while the standard PDF $p(\boldsymbol{x})$ is the distribution of one object. For IID RFS, the multi-object PDF $f(\boldsymbol{x})$ and PHD $v(\boldsymbol{x})$ of the RFS X with $|X| = n$ are calculated by:

$$f(X) = n! \rho(n) \prod_{\boldsymbol{x} \in X} p(\boldsymbol{x}), \tag{4.25}$$

$$v(\boldsymbol{x}) = p(\boldsymbol{x}) \sum_{n=1}^{\infty} n \rho(n). \tag{4.26}$$

Poisson RFS An IID RFS X is called Poisson RFS if the cardinality distribution is Poisson distributed with the rate λ:

$$\rho(n) = \frac{e^{-\lambda}\lambda^n}{n!}. \tag{4.27}$$

This leads to the following equations:

$$f(X) = e^{-\lambda} \prod_{x \in X} \lambda p(\boldsymbol{x}), \tag{4.28}$$

$$v(\boldsymbol{x}) = \lambda p(\boldsymbol{x}). \tag{4.29}$$

Poisson RFS are an important type, since those are completely characterized by their intensities (Vo and Ma 2006) and thus are the basis for the PHD filter as shown in chapter 4.4.3.

4.4.2 Multi target Bayes filter

Similar to the single target Bayesian filtering described in chapters 4.2 and 4.3, Bayesian filtering can be applied to multi target frameworks. This implies, that the filtering process estimates both the cardinality and the states of an RFS with their uncertainties. Given a state RFS X_k and a corresponding measurement RFS Z_k, the prediction and update step of an exact multi target filter can be calculated using the Markov assumption:

$$f_{k|k-1}(X_k|Z_{1:k-1}) = \int \Pi_{k|k-1}(X_k|X_{k-1}) f_{k-1|k-1}(X_{k-1}|Z_{1:k-1})\, \delta X_{k-1}, \tag{4.30}$$

$$f_{k|k}(X_k|Z_{1:k}) = \frac{\varphi_k(Z_k|X_k) f_{k|k-1}(X_k|Z_{1:k-1})}{\int \varphi_k(Z_k|X) f_{k|k-1}(X|Z_{1:k-1})\, \delta X}. \tag{4.31}$$

This formulation, however, does not have a general analytic closed form solution (Ristic 2013) and is computational very demanding, since the multi-target transitional density $\Pi_{k|k-1}(X_k|X_k - 1)$ and the multi-target likelihood $\varphi_k(Z_k|X_k)$ are complex and the integrals in (4.30) and (4.31) are set integrals. Therefore, the exact multi target filter is generally known to be computationally infeasible for most applications. Several approximations emerged, that can massively decrease the computational demands using different assumptions, which are summarized here:

- The Probability Hypothesis Density (PHD) filter (Mahler 2003) propagates the PHD instead of the multi-object probability density, which reduces the computational complexity.
- The Cardinalized Probability Hypothesis Density (CPHD) filter (Mahler 2007a) propagates the cardinality distribution in addition to the first order statistical moment to increase the accuracy and stabilize the cardinality estimation.
- The Cardinality Balanced Multi Bernoulli Filter (CBMeMBer) (Vo et al. 2009) propagates a multi-Bernoulli RFS, which is a sum of Bernoulli RFSs, that can either be empty or have a single object.

In addition to the popular filter approaches listed here, the recent years show the developments of additional RFS-based filters. In particular, labeled RFS filters are mentioned, which directly include labeled tracks to keep track of full trajectories over time (Beard et al. 2020; Lu et al. 2017; Reuter 2014). However, due to the simplicity, low computational demands and good performances in the experiments of chapter 6.6, the focus of this thesis is on the PHD filter. In general, most of these filters listed above can be implemented using Sequential Monte Carlo (SMC) methods or using a Gaussian distribution approximation with similar assumptions like the KF. The calculation of the Gaussian version is usually faster, but it is possible to implement real-time capable SMC filters, like Reuter (2014) showed using a GPU. However, this thesis focuses on the Gaussian implementations that have low runtimes.

4.4.3 PHD filter

The PHD filter (Mahler 2003) approximates the exact multi target filter by only propagating the first order statistical moment (PHD) of an RFS. Under the assumption of a Poisson RFS, there is a closed form analytic solution for the prediction and update step, since both the cardinality distribution and the spatial distribution are defined by the PHD. The main advantages of the PHD filter is the fast computation, since no set integrals are used for the estimation process. The main drawback of the PHD filter is the cardinality estimation, which is less accurate than other RFS-based filters and can be unstable (Reuter 2014, p. 42). To overcome this weakness, either the cardinalized PHD filter (Mahler 2007a) or a second order moment approximation, as shown by Schlangen et al. (2018) can be used. The "spooky effect" (Franken et al. 2009; Vo and Vo 2012) is also an issue, that can appear for both the PHD and CPHD filter. This effect can lead to a weight shift in case of missed detections for objects far apart from each other.

In order to examine the theoretical limitations of the PHD filter in comparison to the CPHD filter, both filters are tested on real data in this thesis. Short term weight drops, as they can appear from the spooky effect are adressed by the track confirmation strategy proposed in chapter 5.9. Similar to other Bayesian filters, the PHD filter is divided into prediction and update, which are described in the next sections.

The PHD prediction of the PHD $v_{k|k-1}(\boldsymbol{x})$ consists of three parts: The birth of new targets, the spawning of new targets from existing ones and the prediction of existing targets:

$$\begin{aligned} v_{k|k-1}(\boldsymbol{x}) =& v^{b}_{k|k-1}(\boldsymbol{x}) \\ &+ \int p^{s}_{k|k-1}(\boldsymbol{x}|\boldsymbol{x'}) v_{k-1|k-1}(\boldsymbol{x'})\, d\boldsymbol{x'} \\ &+ \int p_S(\boldsymbol{x'}) \pi_{k|k-1}(\boldsymbol{x}|\boldsymbol{x'}) v_{k-1|k-1}(\boldsymbol{x'})\, d\boldsymbol{x'}. \end{aligned} \tag{4.32}$$

The birth process is modeled by the likelihood $v^{b}_{k|k-1}(\boldsymbol{x})$ that new targets appear. The spawn process is modeled by the likelihood $p^{s}_{k|k-1}(\boldsymbol{x}|\boldsymbol{x'})$, that a target with state $\boldsymbol{x}$ will spawn from the target with state $\boldsymbol{x'}$. The prediction process is modeled by the transitional single-target density $\pi_{k|k-1}(\boldsymbol{x}|\boldsymbol{x'})$ and the survival probability $p_S(\boldsymbol{x'})$, which is the probability of a target with state $\boldsymbol{x'}$ to survive from time-step $k-1$ to k.

The PHD update of the PHD $v_{k|k}(\boldsymbol{x})$ consists of two parts: The estimation for non-detected targets and the estimation for detected targets:

$$\begin{aligned} v_{k|k}(\boldsymbol{x}) =& (1 - p_D(\boldsymbol{x})) v_{k|k-1}(\boldsymbol{x}) \\ &+ \sum_{\boldsymbol{z} \in Z_k} \frac{p_D(\boldsymbol{x}) g_k(\boldsymbol{z}|\boldsymbol{x})}{\kappa_k(\boldsymbol{z}) + \int p_D(\boldsymbol{x'}) g_k(\boldsymbol{z}|\boldsymbol{x'}) v_{k|k-1}(\boldsymbol{x'})\, d\boldsymbol{x'}} v_{k|k-1}(\boldsymbol{x}). \end{aligned} \tag{4.33}$$

The estimation of non-detected targets is modeled by $(1-p_D(\boldsymbol{x}))$, where the detection probability $p_D(\boldsymbol{x})$ is the probability that a target with state $\boldsymbol{x}$ is detected. For the calculation of the detected targets, the sensor likelihood function $g_k(\boldsymbol{z}|\boldsymbol{x})$ is used. $\kappa_k(\boldsymbol{z})$ models the appearance of clutter, which is defined to be a Poisson RFS with PHD $\kappa_k(\boldsymbol{z}) = \lambda c(\boldsymbol{z})$, where $c(\boldsymbol{z})$ is the spatial distribution of clutter and the Poisson rate λ.

The PHD filter equations shown above can be implemented as a particle filter using the SMC method (see Ristic (2013) for details). Alternatively, Vo et al. proposed the Gaussian Mixture Probability Hypothesis Density (GM-PHD) filter (Vo and Ma 2006), which results in a very fast implementation. The filter equations derived for the GM-PHD use the following assumptions provided by Vo and Ma (2006):

- A.1: Each target evolves and generates observations independently of one another.
- A.2: Clutter is Poisson distributed and independent of target-originated measurements.
- A.3: The predicted multiple-target RFS governed by $f_{k|k-1}$[1] is Poisson distributed.
- A.4: Each target follows a linear Gaussian dynamical model and the sensor has a linear Gaussian measurement model, i.e.

$$\pi_{k|k-1}(\boldsymbol{x}|\boldsymbol{x'}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{A}_{k-1}\boldsymbol{x'}, \boldsymbol{Q}_{k-1})^1,$$
$$g_k(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{H}_k\boldsymbol{x}, \boldsymbol{R}_k)^1.$$

- A.5: The survival and detection probabilities are state independent, i.e. $p_{S,k}(\boldsymbol{x}) = p_{S,k}$ and $p_{D,k}(\boldsymbol{x}) = p_{D,k}$.
- A.6: The intensities of the birth and spawn RFSs are Gaussian mixtures.

The assumption of Poisson distributions of A.2 and A.3 is necessary to be able to characterize the complete RFS using the PHD. Note, that state-depending detection and survival probabilities can be used, if they are modeled by mixtures using the equations shown by Vo and Ma (2006).

4.4.3.1 The GM-PHD filter equations

The GM-PHD equations can also be extended to non-linear formulations by using the equivalent non-linear KF, like the EKF or UKF. Based on the assumptions above, a recursive formulation with prediction and update can be derived, as shown by Vo and Ma (2006). In this formulation, the PHD $v_{k-1}(\boldsymbol{x})$ at time step k is always represented as a sum of J_{k-1} weighted Gaussians with weight $w_{k-1}^{(i)}$, mean $\boldsymbol{m}_{k-1}^{(i)}$ and covariance $\boldsymbol{P}_{k-1}^{(i)}$:

$$v_{k-1}(\boldsymbol{x}) = \sum_{i=1}^{J_{k-1}} w_{k-1}^{(i)} \mathcal{N}(\boldsymbol{x}; \boldsymbol{m}_{k-1}^{(i)}, \boldsymbol{P}_{k-1}^{(i)}). \tag{4.34}$$

Note, that the GM-PHD filter uses Gaussian components with mean values that combine to describe the intensity function, in contrast to the individual states used for separate tracks of the KF approach.

The GM-PHD prediction $v_{k|k-1}(\boldsymbol{x})$ is calculated by the following equation:

$$v_{k|k-1}(\boldsymbol{x}) = v_{S,k|k-1}(\boldsymbol{x}) + v_{\beta,k|k-1}(\boldsymbol{x}) + \gamma_k. \tag{4.35}$$

Here, $v_{\beta,k|k-1}(\boldsymbol{x})$ models the spawn process and $\gamma_k(\boldsymbol{x})$ the birth process that are represented as Gaussian mixtures as stated by A.6. The prediction is modeled by $v_{S,k|k-1}(\boldsymbol{x})$ with

$$v_{S,k|k-1}(\boldsymbol{x}) = p_{S,k} \sum_{j=1}^{J_{k-1}} w_{k-1}^{(j)} \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{m}_{S,k|k-1}^{(j)}, \boldsymbol{P}_{S,k|k-1}^{(j)}\right), \tag{4.36}$$

$$\boldsymbol{m}_{S,k|k-1}^{(j)} = \boldsymbol{A}_{k-1}\boldsymbol{m}_{k-1}^{(j)}, \tag{4.37}$$

$$\boldsymbol{P}_{S,k|k-1}^{(j)} = \boldsymbol{Q}_{k-1} + \boldsymbol{A}_{k-1}\boldsymbol{P}_{k-1}^{(j)}\boldsymbol{A}_{k-1}^T \tag{4.38}$$

using the survival probability $p_{S,k}$. The means and covariances are predicted using the state transition matrix $\boldsymbol{A}_{k-1}$ and process noise covariance matrix $\boldsymbol{Q}_{k-1}$.

[1] Formulas adapted to fit nomenclature of thesis

The GM-PHD update step then calculates the posterior intensity $v_k(\boldsymbol{x})$ at time k:

$$v_k(\boldsymbol{x}) = (1 - p_{D,k})\, v_{k|k-1}(\boldsymbol{x}) + \sum_{\boldsymbol{z} \in Z_k} v_{D,k}(\boldsymbol{x}; \boldsymbol{z}) \tag{4.39}$$

with

$$v_{D,k}(\boldsymbol{x}; \boldsymbol{z}) = \sum_{j=1}^{J_{k|k-1}} w_k^{(j)}(\boldsymbol{z}) \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{m}_{k|k}^{(j)}(\boldsymbol{z}), \boldsymbol{P}_{k|k}^{(j)}\right), \tag{4.40}$$

$$w_k^{(j)}(\boldsymbol{z}) = \frac{p_{D,k} w_{k|k-1}^{(j)} q_k^{(j)}(\boldsymbol{z})}{\kappa_k(\boldsymbol{z}) + p_{D,k} \sum_{l=1}^{J_{k|k-1}} w_{k|k-1}^{(l)} q_k^{(l)}(\boldsymbol{z})}, \tag{4.41}$$

$$q_k^{(j)}(\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{z}; \boldsymbol{H}_k \boldsymbol{m}_{k|k-1}^{(j)}, \boldsymbol{R}_k + \boldsymbol{H}_k \boldsymbol{P}_{k|k-1}^{(j)} \boldsymbol{H}_k^T\right), \tag{4.42}$$

$$\boldsymbol{m}_{k|k}^{(j)}(\boldsymbol{z}) = \boldsymbol{m}_{k|k-1}^{(j)} + \boldsymbol{K}_k^{(j)}\left(\boldsymbol{z} - \boldsymbol{H}_k \boldsymbol{m}_{k|k-1}^{(j)}\right), \tag{4.43}$$

$$\boldsymbol{P}_{k|k}^{(j)} = \left[\boldsymbol{I} - \boldsymbol{K}_k^{(j)} \boldsymbol{H}_k\right] \boldsymbol{P}_{k|k-1}^{(j)}, \tag{4.44}$$

$$\boldsymbol{K}_k^{(j)} = \boldsymbol{P}_{k|k-1}^{(j)} \boldsymbol{H}_k^T (\boldsymbol{H}_k \boldsymbol{P}_{k|k-1}^{(j)} \boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1}. \tag{4.45}$$

Here, $p_{D,k}$ is the detection probability and $\kappa_k(\boldsymbol{z})$ the clutter density, while the update of the Gaussian components is done using the observation matrix $\boldsymbol{H}_k$ and the observation noise covariance matrix $\boldsymbol{R}_k$.

To give some intuition about the GM-PHD filter, Figure 4.4 shows a scene with several vehicles and a possible visualization of it's PHD. With a working filter, peaks will arise in the PHD to track the measurements over time. No data association is necessary, since the PHD represents both the states and the cardinality.

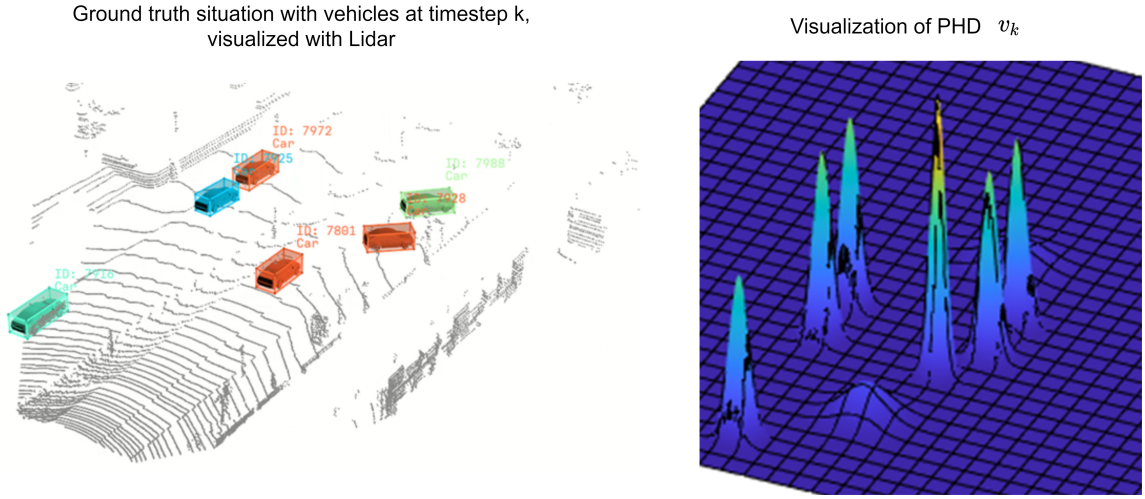


Figure 4.4: Intuition of GM-PHD filter shown by visualization of intensity on right side based on situation on left side. The visualization is an example to gain intuition and does not focus on accuracy.

4.4.4 PHD filter in MOT

The GM-PHD filter has already been used for automotive MOT applications, as shown by Granström (2012); Lindenmaier et al. (2022); Shi et al. (2022). Most of these practical implementations use additional software components or adaptations of the GM-PHD filter in order to produce an efficient and reliable system. Some of these approaches are described here.

4.4.4.1 Adaptive birth intensity

The original birth process is independent of the received measurements and assumed to be known. In case of known hotspots of target births, like at airports for air surveillance, this may be reasonable. In MOT applications, where objects can appear in the whole measurement space, many potential birth targets spread over the whole monitored space need to be modeled to avoid "blind spots". To overcome this inefficiency, Ristic (2013) shows an adaptive birth intensity model, based on the current measurements likelihood $g_k(\boldsymbol{z}|\boldsymbol{x})$. This leads to an increased birth intensity in areas of current measurements. A similar approach is derived for the Gaussian mixture implementation by Houssineau and Laneuville (2010), which results in more efficient calculations.

4.4.4.2 Labeled tracking

A large weakness of the PHD filter for MOT applications is, that it is not able to preserve a track's unique ID throughout time. A unique identity of tracked objects is referred to as label in the context of RFS filters. The GM-PHD filter does not directly calculate labeled tracks, but instead the unlabeled intensity function, as shown by Figure 4.4. Therefore, the intensity is tracked throughout time, but there is no direct tracking of unique labels, since this information is not included in the Gaussian mixture description formulated in equation (4.34). However, for ADAS applications, this is important information for verification of tracks. There are several techniques that overcome this issue.

Liu et al. (2015) utilizes the aliasing of particle clouds for association between time-steps in order to keep track continuity for a particle-PHD filter. Panta et al. (2005) introduce hidden indices to the particles of a particle-PHD filter, which are used to store the identity over time and therefore can be used for the creation of labels. A similar approach is shown by Clark et al. (2006) for the GM-PHD implementation, where tags are applied to each Gaussian component that are propagated over time, which is a simple and effective solution. Panta et al. (2006) show a tracking scheme with a tree structure based on tags for the GM-PHD, which enables more advanced possibilities in track pruning. This approach also showed better tracking performance than a MHT approach (Panta et al. 2006). Another option is to implement a labeled PHD filter based on the labeled random finite set theory, as shown by Lu et al. (2017). In this thesis, an approach with tags similar to the work of Clark et al. (2006) and Panta et al. (2006) is proposed in chapter 5.7.

A classic track management approach from chapter 4.3.2.2 assigns an ID to each track with a KF as long as the track is alive. In contrast, labeled tracking for RFS filters integrates the label assignment to the joint existence and state estimation of all targets.

4.4.4.3 Typical implementation

The derived equations from Vo and Ma (2006), which are shown in equation (4.36) to (4.45), are usually implemented in 5 computation steps. They are described by Clark et al. (2006) and are listed here. After initialization, the filter repeats the steps 1 to 5:

- Step 0: Initialization at $k = 0$.
- Step 1: Prediction of the prior Gaussian mixture.
- Step 2: Update by a measurement Gaussian mixture.
- Step 3: Pruning of Gaussian components with low weights.
- Step 4: Merging of Gaussian components that are similar.
- Step 5: Estimation of target states by interpreting Gaussian components with high weights as tracks.

Step 3 and 4 are necessary in real-world implementations, since the number of Gaussian components would rapidly increase without the reduction in these steps. The fifth step is to output the filtered targets. To obtain a track label in addition to the tracked state, some sort of management system needs to be applied, as shown in the previous paragraph. The exact implementation details of the steps used in this thesis are shown in chapter 5.7.

4.4.4.4 State-dependent detection probability

Although the standard implementation assumes a constant detection probability in assumption A.5, Vo and Ma (2006) show the implementation of a state dependent detection probability $p_{De,k}(\boldsymbol{x})$, which is modeled as a mixture with an offset weight $w_{d,k}^{(0)}$ and $J_{d,k}$ components with mean $\boldsymbol{m}_{d,k}^{(j)}$ and covariance $\boldsymbol{P}_{d,k}^{(j)}$:

$$p_{De,k}(\boldsymbol{x}) = w_{d,k}^{(0)} + \sum_{j=1}^{J_{d,k}} w_{d,k}^{(j)} \mathcal{N}(\boldsymbol{x}, \boldsymbol{m}_{d,k}^{(j)}, \boldsymbol{P}_{d,k}^{(j)}). \tag{4.46}$$

Based on this, the update step of the GM-PHD filter is calculated as shown by (Vo and Ma 2006):

$$v_k(\boldsymbol{x}) = v_{k|k-1}(\boldsymbol{x}) - v_{d,k}(\boldsymbol{x}) + \sum_{\boldsymbol{z} \in Z_k} v_{d,k}(\boldsymbol{x}; \boldsymbol{z}), \tag{4.47}$$

with

$$\begin{aligned}
v_{d,k}(\boldsymbol{x}) &= \sum_{i=1}^{J_{k|k-1}} \sum_{j=0}^{J_{d,k}} w_{k|k-1}^{(i,j)} \mathcal{N}(\boldsymbol{x}; \boldsymbol{m}_{k|k-1}^{(i,j)}, \boldsymbol{P}_{k|k-1}^{(i,j)}), \\
w_{k|k-1}^{(i,j)} &= w_{d,k}^{(j)} w_{k|k-1}^{(i)} q_{k|k-1}^{(i,j)}, \\
q_{k|k-1}^{(i,0)} &= 1, \quad \boldsymbol{m}_{k|k-1}^{(i,0)} = \boldsymbol{m}_{k|k-1}^{(i)}, \quad \boldsymbol{P}_{k|k-1}^{(i,0)} = \boldsymbol{P}_{k|k-1}^{(i)}, \\
q_{k|k-1}^{(i,j)} &= \mathcal{N}(\boldsymbol{m}_{d,k}^{(j)}; \boldsymbol{m}_{k|k-1}^{(i)}, \boldsymbol{P}_{d,k}^{(j)} + \boldsymbol{P}_{k|k-1}^{(i)}), \\
\boldsymbol{m}_{k|k-1}^{(i,j)} &= \boldsymbol{m}_{k|k-1}^{(i)} + \boldsymbol{K}_{k|k-1}^{(i,j)} (\boldsymbol{m}_{d,k}^{(j)} - \boldsymbol{m}_{k|k-1}^{(i)}), \\
\boldsymbol{P}_{k|k-1}^{(i,j)} &= (\boldsymbol{I} - \boldsymbol{K}_{k|k-1}^{(i,j)}) \boldsymbol{P}_{k|k-1}^{(i)}, \\
\boldsymbol{K}_{k|k-1}^{(i,j)} &= \boldsymbol{P}_{k|k-1}^{(i)} (\boldsymbol{P}_{k|k-1}^{(i)} + \boldsymbol{P}_{d,k}^{(j)})^{-1}
\end{aligned} \tag{4.48}$$

and

$$\begin{aligned}
v_{d,k}(\boldsymbol{x}; \boldsymbol{z}) &= \sum_{i=1}^{J_{k|k-1}} \sum_{j=0}^{J_{d,k}} w_k^{(i,j)}(\boldsymbol{z}) \mathcal{N}(\boldsymbol{x}; \boldsymbol{m}_{k|k}^{(i,j)}(\boldsymbol{z}), \boldsymbol{P}_{k|k}^{(i,j)}), \\
w_k^{(i,j)}(\boldsymbol{z}) &= \frac{w_{k|k-1}^{(i,j)} q_k^{(i,j)}(\boldsymbol{z})}{\kappa_k(\boldsymbol{z}) + \sum_{r=1}^{J_{k|k-1}} \sum_{s=0}^{J_{d,k}} w_{k|k-1}^{(r,s)} q_k^{(r,s)}(\boldsymbol{z})}, \\
q_k^{(i,j)}(\boldsymbol{z}) &= \mathcal{N}(\boldsymbol{z}; \boldsymbol{H}_k \boldsymbol{m}_{k|k-1}^{(i,j)}, \boldsymbol{R}_k + \boldsymbol{H}_k \boldsymbol{P}_{k|k-1}^{(i,j)} \boldsymbol{H}_k^T), \\
\boldsymbol{m}_{k|k}^{(i,j)}(\boldsymbol{z}) &= \boldsymbol{m}_{k|k-1}^{(i,j)} + \boldsymbol{K}_k^{(i,j)} (\boldsymbol{z} - \boldsymbol{H}_k \boldsymbol{m}_{k|k-1}^{(i,j)}), \\
\boldsymbol{P}_{k|k}^{(i,j)} &= (\boldsymbol{I} - \boldsymbol{K}_k^{(i,j)} \boldsymbol{H}_k) \boldsymbol{P}_{k|k-1}^{(i,j)}, \\
\boldsymbol{K}_k^{(i,j)} &= \boldsymbol{P}_{k|k-1}^{(i,j)} \boldsymbol{H}_k^T (\boldsymbol{H}_k \boldsymbol{P}_{k|k-1}^{(i,j)} \boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1}.
\end{aligned} \tag{4.49}$$

The main difference to the equations 4.39-4.45 is that here a sum is formed over elements of both the detection probability $p_{De,k}$ and the predicted intensity function to calculate $v_{d,k}(\boldsymbol{x})$ and $v_{d,k}(\boldsymbol{x}; \boldsymbol{z})$. With this implementation, negative weights of Gaussian components may occur during the update step. Therefore, the prune and merge step needs mechanisms to ensure that the intensity function remains positive. This means, that the sum of all weights is not allowed to be negative.

4.5 Classification tracking

In automotive applications, MOT is applied to multiple classes of objects, like cars, pedestrians or cyclists. A correct classification is necessary for several ADAS functions to work well. There are two fundamental approaches dealing with different classes: Either there is one MOT application for each class that deals exclusively with this class, or one MOT application deals with objects from multiple classes. Some automotive sensors, like Lidar or radar, misclassify detected objects on a regular basis. Therefore, it is more feasible to use a MOT application tracking objects of all classes and allow the change of classification in case of prior misclassification.

4.5.1 Bayesian framework for classification tracking

By estimating the class over multiple time-steps and sensors, the classification result gets more stable. To create an universal framework, that works well with different numbers of classes, a Bayesian formulation of the classification estimation similar to the existence estimation (Aeberhard 2017; Bader 2019) can be used. Given a set of classes $C = \{C_1, C_2, \dots, C_k\}$, the probability for each class C_i can be calculated using the classification vector $\boldsymbol{c}_k = [p(C_{1,k}|Z_k), p(C_{2,k}|Z_k), \dots, p(C_{n,k}|Z_k)]$ consisting of the class probabilities $p(C_{i,k}|Z_k)$ given measurements Z_k up to time-step k. In the prediction step of this classification vector, the classification transition probabilities p_{ij} define the transition between classes i and j for one time-step:

$$p(C_{i,k}|Z_{k-1}) = \sum_{C_j \in C} p_{ij} p(C_{j,k-1}|Z_{k-1}). \tag{4.50}$$

The transition probabilities can be represented as a matrix $\boldsymbol{P}_t^{n \times n}$. Each element p_{ij} is the transition probability from class i to class j and the sum of each row in $\boldsymbol{P}_t$ equals one:

$$\begin{aligned} \boldsymbol{P}_t &= \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nn} \end{bmatrix}, \\ r_i &= \sum_{j=1}^{n} p_{ij} = 1. \end{aligned} \tag{4.51}$$

The update step is then correcting the classification using the measured classification probability $p(\boldsymbol{z}_k|C_{i,k})$ provided by a sensor with the normalization factor η:

$$p(C_{i,k}|Z_k) = \eta p(\boldsymbol{z}_k|C_{i,k}) p(C_{i,k}|Z_{k-1}), \tag{4.52}$$

$$\eta = \left[\sum_{C_i in C} p(\boldsymbol{z}_k|C_{i,k}) p(C_{i,k}|Z_{k-1}) \right]^{-1} \tag{4.53}$$

Note, that it is possible to use a generalized Bayes formulation similar to the existence probability in chapter 4.3.2 to create a more robust estimation using an evaluation window of L time-steps.

The approach shown here is well suited for estimating the classification probabilities of single sensor systems, but does not include the confidence level of the current update. This becomes necessary, when updates from multiple sensors occur, since sensors like cameras typically have higher confidences on the classification compared to radar. In chapter 5.6, additional methods that also focus on the fusion of multiple classifications from different sensor sources with different confidences are developed and tested.

4.5.2 Dempster-Shafer framework for classification tracking

The Dempster-Shafer Theory (DST), introduced by Dempster (1967) and Shafer (1976), is a mathematical framework for evidence calculation. The framework can be used in a broad field of applications in the area of environment perception, like occupancy grids (Yi et al. 2000; Nuss et al. 2018), existence estimation (Aeberhard et al. 2011) or the classification fusion (Aeberhard 2017). A detailed introduction and comparison to the Bayesian theory is provided by Koks and Challa (2003); Challa and Koks (2004).

The DST uses an interval of "support" (Spt) and "plausibility" (Pls) with Spt $\leq$ Pls instead of the single value of probability as used by the Bayesian framework. Here, the support can be interpreted as the lower limit for the evidence of a hypothesis, while the plausibility is the upper limit with the difference being the uncertainty (Aeberhard 2017; Challa and Koks 2004). Therefore, the probability roughly lies within the uncertainty bound by support and plausibility (Challa and Koks 2004). One major advantage over the Bayes theory is, that it allows the explicit modeling of an unknown hypothesis.

To achieve this, a power set 2^C, that includes all possible subsets of hypotheses, is created from the set of classes C where all components have an assigned mass m. Therefore, the power-set for this example includes not only the individual classes but also their combinations. With a set of $C = \{C_1, C_2, C_3\}$ such combinations could be "either C_1 or C_2" with $C_{12} = C_1, C_2$ and "any class" with $C_{123} = C = C_1, C_2, C_3$, where the hypothesis "any class" would represent unknown. The number of hypotheses in the power set is therefore $2^{|C|} = 8$, including the empty set $\emptyset$.

The support Spt and plausibility Pls of hypothesis A can now be calculated using the masses m assigned to each of the hypotheses of the power set. For this calculation, the sum of all masses has to be 1:

$$\sum_{A \subseteq P^C} m(A) = 1, \tag{4.54}$$

$$\text{Spt}(A) = \sum_{B \subseteq A} m(B), \tag{4.55}$$

$$\text{Pls}(A) = \sum_{B \cap A \neq \emptyset} m(B). \tag{4.56}$$

$A \cap B$ is the intersection of the hypothesis A and B, so the plausibility of A is defined by the masses of all hypotheses intersecting with A. The support of A, on the other hand, is defined by the masses of all hypotheses being subsets of A. The DST can be seen as a generalization of the Bayes theory, since it reduces to the Bayes theory if masses are only assigned to the original hypothesis given by C.

When an old and a new source of evidence should be combined, Dempster's rule of combination can be applied to calculate the new mass $m(C)$ based on the old masses m_o and m_n with

$$m(C) = \frac{\sum_{A \cap B = P^C} m_n(A) m_o(B)}{1 - \sum_{A \cap B = \emptyset} m_n(A) m_o(B)}. \tag{4.57}$$

In MOT applications, DST can be used to propagate the weights of classification hypotheses over time, as shown by Aeberhard (2017). For the classification propagation, the track's class is represented by the masses $m_{k|k-1,T_i}$, which are transported through the track's lifetime, while other masses $m_{k,z_j}^{(s_p)}$ are based on the measurements of sensor (s_p). At each time step, the masses of the track are now updated by the masses of the measurement using equation (4.57):

$$m_{k|k,T_i}(C) = \frac{\sum_{A \cap B = P^C} m_{k|k-1,T_i}(A) m_{k,z_j}^{(s_p)}(B)}{1 - \sum_{A \cap B = \emptyset} m_{k|k-1,T_i}(A) m_{k,z_j}^{(s_p)}(B)}. \tag{4.58}$$

To work with this form, the classification provided by a sensor needs to be converted to masses first. Aeberhard (2017) does this by modeling each hypothesis based on a combination of a weighting factor and the measurement probability.

If probability values are required as output, the masses can be converted to pseudo probabilities using the Pignistic transformation from Smets (1990). The pseudo probability $\tilde{p}(C_a)$ of class C_a is calculated by

$$\tilde{p}(C_a) = \sum_{B \subseteq C \neq \emptyset} \frac{|C_a \cap B|}{|B|} m_{k|k,T_i}(B). \tag{4.59}$$

4.6 Sensor fusion for MOT

Modern ADAS systems require a reliable environment perception, which is usually not given by a single sensor. The performance of camera-based perception, for example, is significantly influenced by environment conditions like fog, rain or darkness. Therefore, many modern solutions feature multiple sensors from different sensing domains to complement each other and create redundancy for all conditions and situations. However, to combine the advantages of all sensors, a robust sensor fusion must process the data of all sensors and create a combined output. The sensor fusion is responsible for the correct combination of the outputs of all sensors, considering the accuracies of all sensors. This processing step is therefore a critical part in a multi-sensor perception pipeline.

There is a large variety of possible approaches, that reaches from function-specific fusion based on heuristic rules to probabilistic models and neural networks. This chapter gives an overview of possible approaches and the requirements for spatial and temporal data alignment.

4.6.1 State of the art and overview

In the literature, sensor fusion concepts are often characterized by the "level" of fusion, which is broadly divided into high-level or decentralized and low-level or centralized (Aeberhard 2017; Challa and Koks 2004; Bar-Shalom and Li 1995; Kämpchen 2007; Darms and Winner 2005). This "level" corresponds to the stage in the processing pipeline, where the data of multiple sensors are fused. A low-level fusion approach is therefore applied on or close to raw data, while high-level fusions are applied on data that is already processed. Low-level approaches are implemented more centralized, while high-level approaches may be implemented decentralized.

4.6.1.1 Levels of fusion

In this overview, the fusion approaches are divided into low-level, feature-level and high-level. For each concept, the advantages and drawbacks are described. A schematic of the fusion concepts is shown in Figure 4.5.

Low-Level: Low-level fusion approaches are typically applied to sensor data before any tracking filter is applied. Therefore, both the fusion of raw data, and the fusion of detections before tracking are low-level fusion approaches. These concepts focus on the maximum possible accuracy by fusing the data from multiple sources before any information loss in processing steps. Examples are the detection approaches shown in chapter 3, which fuse the raw data from several sensors to detect objects, that can be tracked in the next processing step. This raw data fusion is getting rising attention due to the application of neural networks, that can achieve high performances on raw data. Low-level fusion has also been applied to applications with focus

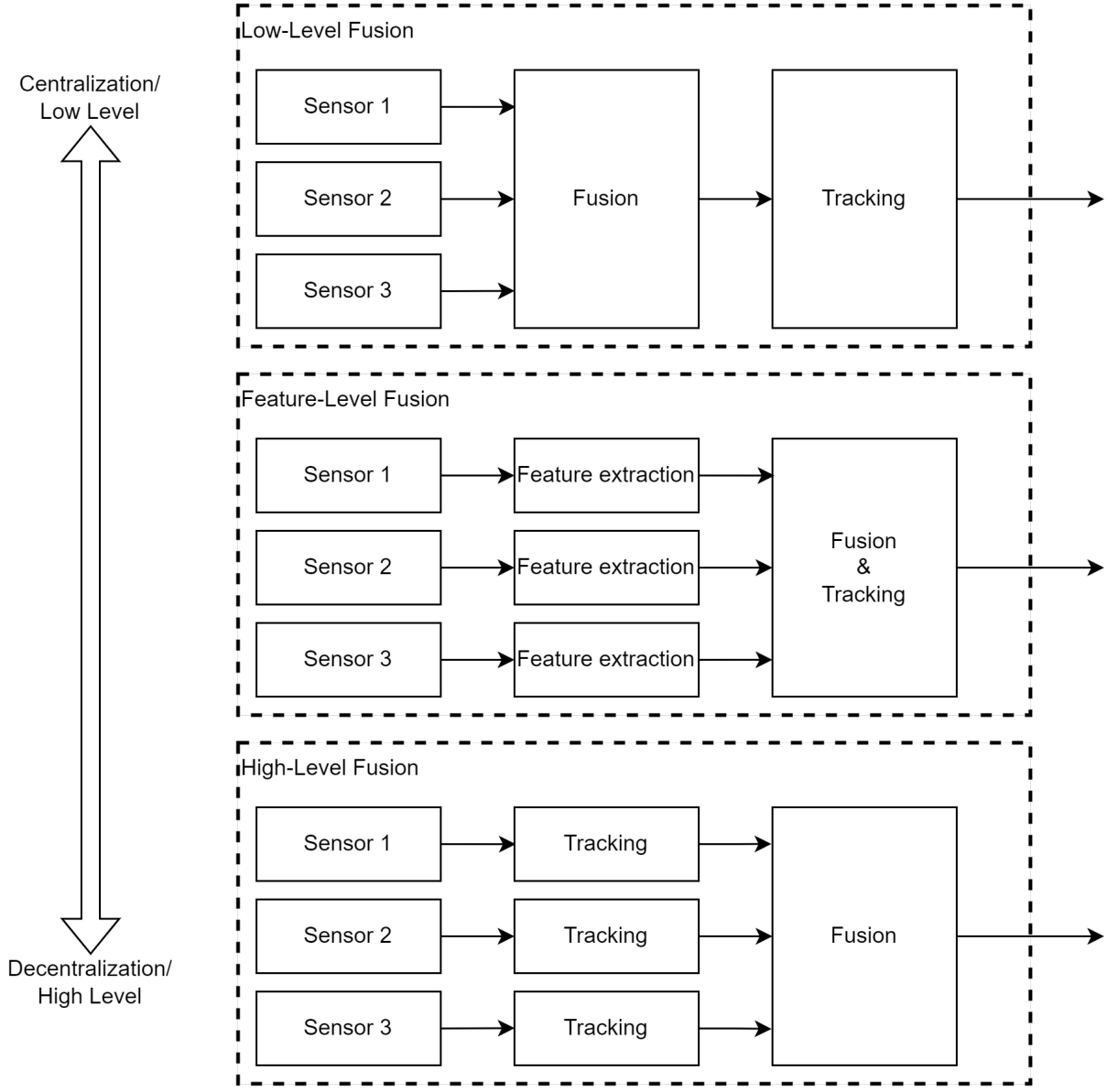


Figure 4.5: Scheme of high-level, feature-level and low-level fusion concepts.

on ADAS functions, like pre-crash detection (Pietzsch et al. 2008) or ACC (Takizawa et al. 2004). Usually, low-level approaches are optimized for a specific set of sensors and may not work with different ones, which gives it less flexibility. Low-level fusion concepts fit well to most applications that rely on occupancy grids, since this environment representation does not use high-level features. A brief summary of the advantages and disadvantages is given in Table 4.2.

Table 4.2: Advantages and disadvantages of low-level fusion approaches.

Advantages	Disadvantages
• High accuracy potential. • No information loss before fusion.	• Low flexibility in sensor setup due since input data of different sensors may vary. • High computational demands due to high amount of data. • Precise spatial and temporal alignment necessary, typically a common trigger source is used. • High communication bandwidth necessary due to high amount of data.

Feature-Level: Sensor fusion on feature-level describes a system, that uses a central tracking algorithm which receives measurements in a common data format (e.g. OBBs) from various sensors. Therefore, all sensors must align to a similar type of object representation to fuse them within the tracking process (Kämpchen 2007). Within that process, the information content of the raw data is reduced to the calculated features. Instead of a point cloud, the Lidar sensor, for example, will just report a list of objects. In addition, a sensor is responsible for all sensor-specific parts and calculations of the MOT process and needs to provide all sensor-specific information to the fusion. This includes, for example, the measurement covariances, the measurement noise or detection probabilities, which are sensor-specific. Feature-level fusion offers a good flexibility to the setup, since the sensors can be easily replaced if they support the common object representation and some key sensor data is known. At the same time, "[...] feature-level fusion aims at an optimal tracking and classification performance as the data is fused on a low abstraction level" (Kämpchen 2007). Feature-level fusion approaches have been used in autonomous driving projects, including successful participants at the Darpa Urban Challenge (Darms et al. 2009). They are also proven to be effective in other automotive applications, like ACC (Mahlisch et al. 2006). Overall, this concept can be used for most applications, that use an object-based perception framework, where the sensors deliver non-tracked detections.

In feature-level fusion systems, Bayesian filters, like the KF, are often used as a central tracking approach. These filters typically assume to have uncorrelated sensor measurements. If this is not given (e.g. each sensor uses its own tracking prior to the central tracking), disruptive effects, like the common process noise (Bar-Shalom and Campo 1986) can appear. In these situations, high-level track-to-track fusion approaches can lead to better performances, as shown by (Matzka and Altendorfer 2008). In practical applications, however, this assumption might be violated to be able to use both tracked and non-tracked detections in a single central fusion.

While low-level fusion typically needs to trigger all sensors for measurements at the exact same time, this is often not necessary in feature-level and high-level approaches. The tracked states can be predicted to any time stamp of the measurements to be updated. A brief summary of the advantages and disadvantages is given in Table 4.3.

Table 4.3: Advantages and disadvantages of feature-level fusion approaches.

Advantages	Disadvantages
• Flexibility for sensor setups with common object representation. • Low/Moderate computational demands. • No common trigger source for simultaneous measurement necessary, since filter update can be performed at any time. • Good tradeoff between performance, computational demands and flexibiltiy.	• Moderate information loss due to feature calculation. • Correlated detections can lead to performance degradation.

High-Level: The goal of high-level fusion systems is a maximum flexibility and modularity by encapsulating the sensors completely from the fusion. Therefore, each sensor independently detects and tracks objects before sending these filtered data to the fusion system. The fusion system itself just combines the estimates from all sensors to a global estimation. Therefore, it needs to know the least amount of information about the used sensors and allows distributed processing. However, this can cause some challenges in terms of accuracy, since it implies a high information loss. High-level fusion approaches have been used in automotive applications, as shown by Labayrade et al. (2005); Mobus and Kolbe (2004); Aeberhard (2017); Floudas et al. (2007). The High-level fusion concept fits best to applications, where several sensors are encapsulated and provide tracked high-level object data. For high-level track-to-track fusion, algorithms like covariance intersection, covariance union and the use of cross covariance can lead to good performances (Matzka and Altendorfer 2008). This requires that complex detection and tracking algorithms are integrated to the sensors. A brief summary of the advantages and disadvantages is given in Table 4.4.

Table 4.4: Advantages and disadvantages of high-level fusion approaches.

Advantages	Disadvantages
• Highest flexibility regarding sensor setup. • Low computational demands (on fusion side). • No common trigger source for simultaneous measurement necessary, since fusion can be performed at any time. • Low communication bandwidth, since only high-level tracking information are transmitted.	• Highest information loss: Working with filtered information from extracted features. • Potential lower performance due to information loss.

Comparison and conclusion Several works compare the different fusion concepts against each other based on different criteria. Becker (1999) comes to the conclusion, that a low-level measurement fusion is the best suited for their autonomous vehicle application. The analysis of Herpel et al. (2008) shows that a low-level fusion achieves better performances than high-level approaches. Darms and Winner (2005) and Kämpchen

(2007) compare centralized and decentralized concepts, with the result that a feature-level approach delivers a good compromise. The comparison from Aeberhard (2017) favors a high-level approach due to the good practicality of high-level approaches in terms of mass production, functional safety and the supplier chain.

Overall, each fusion concept offers some advantages and drawbacks. When the maximum possible accuracy is required within a fixed sensor setup and a centralized processing architecture, a low-level concept may be the best suited. If the main goal is flexibility and decentralization, a high-level concept may be best suited.

Based on the boundary conditions that are formulated in chapter 1.3, the low-level concept is not suitable for this thesis, since a modular sensor setup is required. The sensors used in for this work usually detect objects in 3D coordinates, which can easily be transformed into a common coordinate system and object representation. In order to maximize the accuracy within the constraints given by the sensors, a feature-level fusion is developed, which is proposed in chapter 5.

4.6.1.2 Fusion frameworks

Regardless of the fusion concept, there are different frameworks to fuse the data from the sensors. This includes probability-based frameworks, neural networks and others.

Typical mathematical frameworks are the Dempster-Shafer theory and the Bayesian theory, which are compared with respect to sensor fusion by Challa and Koks (2004). The Bayesian theory is frequently used in sensor fusion, since Bayesian filters, like the KF or the PHD filter, can be updated by multiple sensors. This provides a probabilistic framework, that can provide optimal filter data. The Dempster-Shafer theory deals with "belief" instead of probability, and includes the plausibility of the sources to the calculation. In a binary problem with the states "zero" and "one", the framework presents the alternative of "unknown" (Challa and Koks 2004), which therefore represents the current knowledge.

In recent years, neural networks have been increasingly used for multi object tracking and sensor fusion, as shown by Park et al. (2021). NNs are in particular used for low-level fusion, where the networks can benefit from an increased amount of raw data from different sensor domains. Nevertheless, they can still be used as assisting roles in Bayesian fusion frameworks, where they are for example used for data association (Liu et al. 2019), or be completely responsible for the fusion.

4.6.2 Temporal and spatial alignment

In multi-sensor architectures, the data from all sensors needs to be aligned for a proper fusion. This requires both spatial alignment and temporal synchronization, since the fusion algorithm needs knowledge about where and when to combine information from different sensors.

4.6.2.1 Spatial alignment

The precise position and orientation of each sensor must be known, to fuse the data together. This allows the algorithms to convert the measurements into a common vehicle coordinate system, where the fusion is performed. The transformation process is shown in chapter 2.5. If the sensor's positions are precisely known and the sensors deliver exact object detections, the detections from the same object will be perfectly spatially aligned after the transformation to the vehicle coordinate system. The position and orientation of the sensors can be obtained by a calibration process.

In truck applications, the spatial alignment additionally includes the motion of the cabin, which is independent of the chassis. Sensors like cameras are often mounted to the cabin, which introduces errors. A correction of these errors requires a dynamic measurement of the cabin's pose relative to the chassis. In order to overcome

the need for additional sensors, current generations of cabin mounted cameras correct these errors by estimating the pose based on visual cues in the camera image, like measuring the position and orientation of the visible horizon. Another possibility is to use approaches where only small errors occur within an acceptable range, like the one proposed in chapter 3.3.3.

4.6.2.2 Temporal alignment

In addition to a precise spatial alignment, the sensors also have to be aligned temporally, which means that they have a precise common time frame. If this is not the case, sensor measurements from different times may be fused.

In many applications, the perception sensors are asynchronous, which means that the detections do not arrive at the same time. If a timestamp within a common time frame is available, they can still be fused by predicting the state vectors to the next fusion time. Figure 4.6 shows a sensor to global approach, where the global object states are continuously updated by the sensors measurements without delay. Here, the global state $\boldsymbol{x}_{k-1}$ at time t_{k_i-1} is predicted to time t_{k_j}, where it is updated by $\boldsymbol{z}_{kj}$.

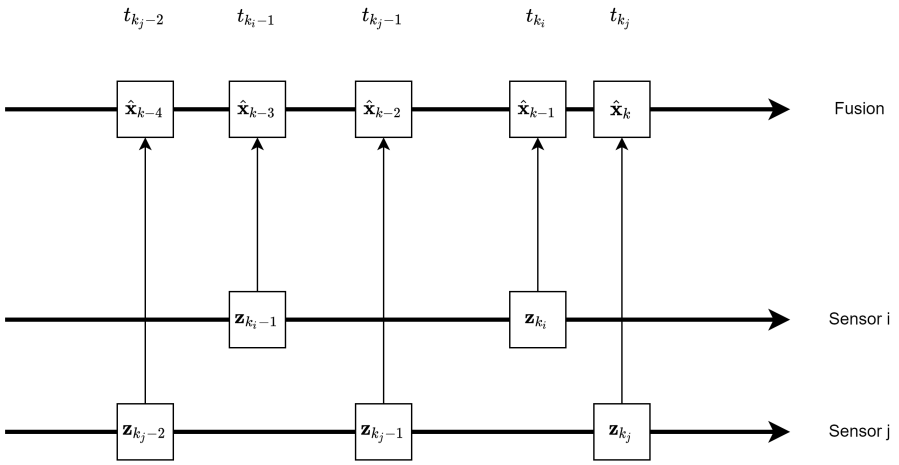


Figure 4.6: Scheme of direct update sensor fusion without delay. Multiple sensors update the states at different time stamps. This idealized system does not include delays between measurement and fusion.

In real-time fusion systems, there are usually delays within the detection process. It takes time for the sensors to process the data and transmit it to a fusion system. These delays are usually not predictable and vary over time, as they frequently arise from processing latency resulting from object detection or tracking algorithms that vary with varying amounts of surrounding objects or environmental scenes. Another reason for communication delays is that they are caused by data transmission over networks with limited bandwidth or buffering.

The delays might lead to an unordered arrival of the measurements at the fusion module, as shown in Figure 4.7. Here, sensor i has a higher delay between the measurement and the availability to the fusion module than sensor j. Therefore, the measurement $\boldsymbol{z}_{k_j}$ from sensor j arrives at the fusion module before the measurement $\boldsymbol{z}_{k_i}$, even though with $t_{k_i} < t_{k_j}$, the measurement from sensor i is observed earlier. Since the fusion module has no knowledge about the duration of varying delays, the update with $\boldsymbol{z}_{k_j}$ is performed as soon as it arrives. This makes $\boldsymbol{z}_{k_i}$ an "out-of-sequence" measurement, since it should have been used prior to $\boldsymbol{z}_{k_j}$. There are several methods to deal with these "out-of-sequence" measurements. Out-of-sequence update equations called retrodiction are derived for the KF, as shown by Bar-Shalom and Li (1995); Bar-Shalom (2002). However, Kämpchen (2007) and Stüker (2003) state, it is difficult to implement exact out-of-sequence approaches. Another possibility is the buffering of arriving measurements, like it is done by Stüker (2003). This approach

always fuses the measurement, if based on the expected measurement times, no other measurement will be sent towards the fusion system. However, this can lead to delays in the fusion output. To obtain simultaneous time stamps, the hardware architecture must be designed to trigger the measurement of all sensors simultaneously and use communication channels with minimum delays, which typically results in higher effort and costs.

This thesis proposes an approach, that focuses on a fixed fusion cycle time to overcome potential delays, but still buffers old measurements in case of "out-of-sequence" corrections in chapter 5.2, which is working for both the KF and GM-PHD filter.

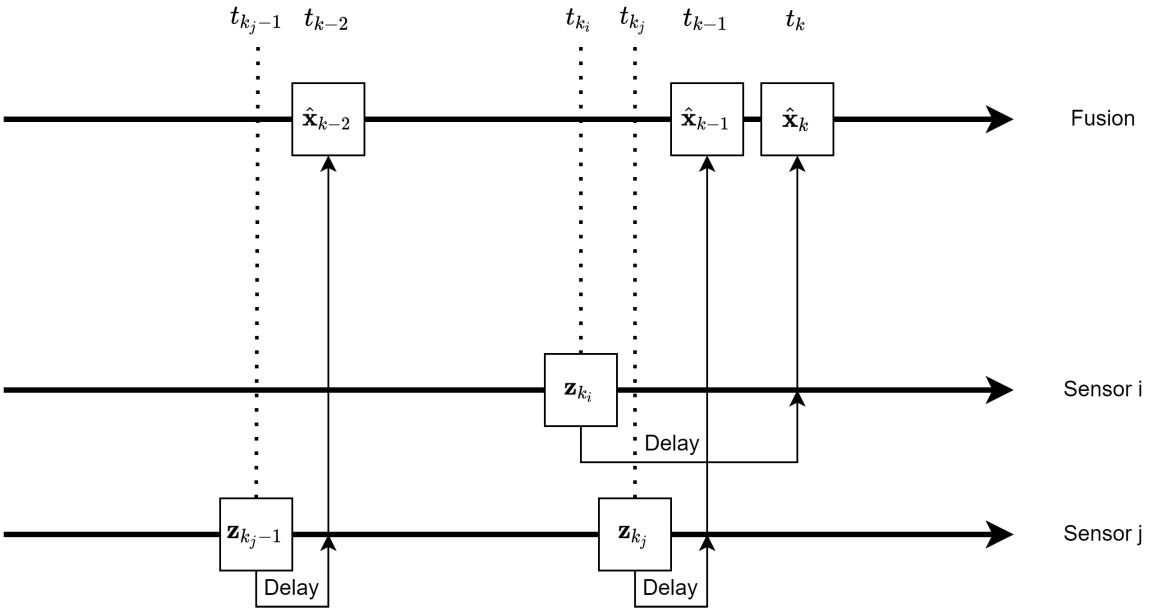


Figure 4.7: Scheme of sensor fusion with different sensor delays. Two sensors i and j with different delays result in the out-of-sequence measurement z_{k_i}.

5 Proposed multi-sensor multi-object tracking pipeline

This chapter proposes a modern and variable sensor fusion framework, that is capable of working with various types of sensors and sensor data types. The main advantage of this type of architecture is the modular construction that easily allows changing the sensor setup and can work with different state estimators, like the Kalman filter and the GM-PHD filter. In contrast to other works, this allows a direct comparison of the KF and GM-PHD filter for real-world systems without limitations in terms of sensor setup and offers a unique possibility to work with both types of filters in a single framework.

While the GM-PHD has been used for some automotive applications, there are usually limitations that prevent the developments from replacing a current state-of-the-art KF-based systems. Therefore, multiple improvements are developed for the practical implementation of the GM-PHD filter for feature-level fusion systems, since previous works either focus on theoretical investigations, single-sensor systems or tailored the application to a specific sensor setup.

The framework is described in the following sections, while the contributions and achievements of this system summarized here:

- A completely modular sensor fusion system capable of dealing with various automotive sensor setups is described.
- A fusion management system for temporal alignment to manage the update order and deal with delayed messages is presented.
- A spatial matching algorithm based on turning functions for mapping all types of geometric object inputs to OBBs to be able to work with sensors of different output types is presented. This brings a huge advantage when working with different types of representations at the same time.
- The development of sensor-based parameter models that describe the tracking parameters, like the detection probability and clutter density for each sensor with respect to multiple influences, like distance, FoV and position in space supported by map data is proposed. In particular, the modeling of the detection probability allows an elegant way of using the GM-PHD for multi-sensor fusion with different FoVs tailored to automotive sensors.
- A track confirmation strategy for the GM-PHD filter for stable and reliable track outputs is proposed, which improves the standard filter description since it increases robustness and can deal with occlusion to a certain extent.
- Multiple approaches for classification fusion, taking into account the different classification confidences of different sensors, are developed.
- The developed framework proposes the integration of the classification fusion to the GM-PHD filter, which is not covered by previous works, but necessary for ADAS applications.

5.1 Fusion framework overview

The proposed fusion framework is based on a Feature-Level fusion layout with a roughly similar structure to that proposed by Kämpchen (2007), where each perception sensor detects objects, which are transmitted to the fusion module. A central tracking algorithm then tracks all the surrounding objects on a global basis and is

supported by a fusion management system. Therefore, the general structure of the framework is able to work with various types of filters, like the KF or a GM-PHD filter.

Figure 5.1 shows an overview of the fusion framework. As shown, N sensors individually detect and classify the objects within their FoV. Here, a sensor is defined as a combination of hardware and software, that is able to provide a list of detections for each time step. Chapter 2 and 3 provide an overview of these models. These detections are then sent to the sensor temporal management system, which is responsible for the buffering and temporal alignment of all measurements to ensure a correct fusion and explained in chapter 5.2. After the temporal management, one of N sensor specific sensor-based parameter models is applied, which enriches the measurement data with sensor specific parameters, like the measurement noise covariance, the classification confidence or the FoV. The sensor-based parameter models are described in detail in chapter 5.5 and are required by the MOT filter in the update step. This is in particular important for the GM-PHD filter implementation when the sensors use different FoVs. The MOT filter & track management part is responsible for the actual tracking and fusion. To be versatile in geometric object representation, a unique measurement mapping procedure is applied to match any geometric measurement to OBBs, which is described in chapter 5.4.1. Different Bayesian filters may be used within the framework for state and existence estimation. In chapter 6.6, the filter performance of a KF and three RFS-based filters are compared using the KITTI dataset and point objects to give an indication about the potential performance. Since the KF has the lowest runtime and is often used in state-of-the art solutions and the GM-PHD filter has the best results, those two filters are implemented within the proposed framework as described in sections 5.7 and 5.8. However, other filter implementations are also possible. For classification estimation and fusion, multiple approaches are developed and tested, as described in chapter 5.6. When state, existence and classification are estimated for each generated track, a track confirmation strategy as described in chapter 5.9 is developed to gain robustness for unstable tracks and against ID changes. The confirmed and stable tracks can then be provided to an ADAS application.

Prerequisites for operation are the sensors' calibration for spatial alignment and the sensors' synchronization for temporal alignment. All used sensors within this thesis are assumed to be perfectly calibrated, and the measurements sent to the fusion module are already converted to vehicle coordinates. Additionally, each measurement is assumed to have a synchronized global timestamp attached for temporal alignment in case of delays, as described by chapter 5.2.

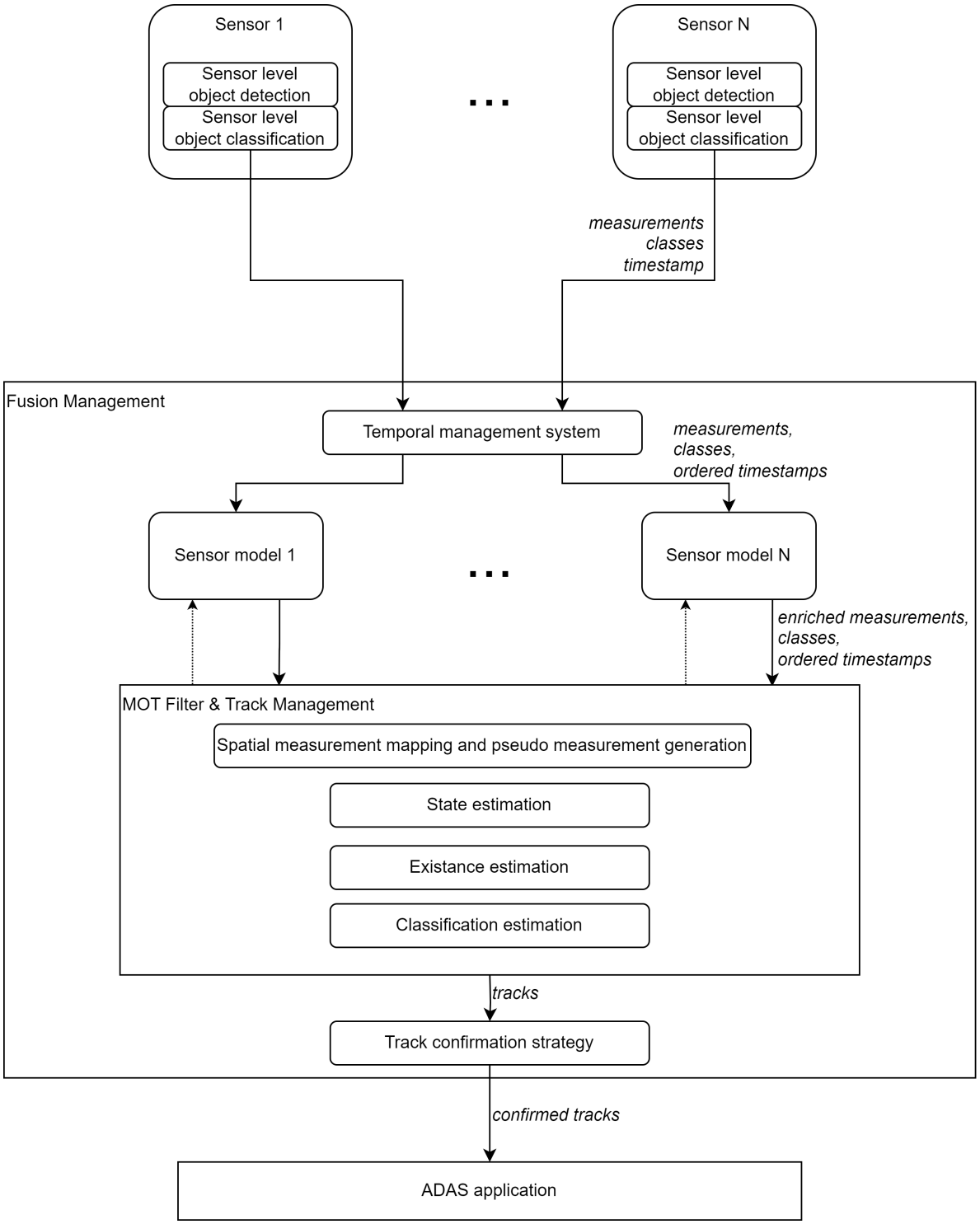


Figure 5.1: Overview of the proposed fusion framework. The top section shows the individual detection and classification of each sensor, while the blocks below show the tasks of fusion, which include estimation of states, existence and classification with inputs from multiple sensors over time.

5.2 Temporal management system

The main task of the fusion management system is the temporal management of all incoming detections. The fusion framework should provide information about the fused tracks with a fixed output data rate, which is independent of the incoming measurement data. This guarantees compliance with the cycle times of ADAS applications regardless of missing or delayed sensor measurement data, which is possible in the used system.

All incoming measurement data is therefore first stored in a measurement buffer, where it is inserted in the correct temporal position in the buffer. The buffer is therefore always sorted according to the time stamp of the measurements. Within the buffer, the oldest measurement not yet used for an update of the MOT system is marked as oldest non-used. If the incoming measurement is older than the last marked measurement due to latency, the incoming measurement is marked instead as oldest non-used. At each time-step in which the MOT system outputs the tracks, the prediction and update cycle is computed for each measurement received one after the other, starting with the one marked as oldest non-used. All states are stored in a state buffer after the update in order to be able to jump back at any time if delayed measurements arrive. The stored states and measurements are dropped, if the time difference to the current time exceeds a threshold.

It is assumed that the storage of the measurements itself does not need significant time and that the total additional runtime of the temporal management system can therefore be described by the number of steps to be recalculated after a jump back. The runtime of one step of the approach is evaluated in chapter 6.7.4.

An example of this buffering principle is shown in Figure 5.2, where the content of the measurement buffer and the state buffer is shown for multiple time stamps. The orange-marked measurement in the measurement buffer represents the oldest non-used measurement. The necessary calculation steps for the respective time steps are indicated by arrows. The fusion system outputs the tracks at fixed intervals at the times t_{k-2}, t_{k-1} and t_k. Therefore, in part 1 happening at $t = t_{k-2}$, the oldest non-used measurement is $\boldsymbol{z}_1$, so state $\boldsymbol{x}_1$ is updated by $\boldsymbol{z}_1$ and then predicted to t_{k-2} to output the tracks $\boldsymbol{x}_{t_{k-2}}$. In part 2, happening at $t = t_{k-1}$, $\boldsymbol{z}_2$ is the oldest non-used measurement, so two prediction and update steps need to be performed to predict the tracks at t_{k-1}. In part 3, with $t_{k-1} < t < t_k$, the out of sequence measurement $\boldsymbol{z}_4$ appears, which is older than the newest prediction. Therefore, $\boldsymbol{z}_4$ is now marked as oldest non-used measurement. In the following estimation step in part 4, the calculation starts at the latest buffered state before $\boldsymbol{z}_4$, which is $\boldsymbol{x}_3$ to be able to predict the current tracks with including all available information. In addition, old measurements and states are dropped, if they are older than the maximum storage duration allows. If strongly delayed measurements older than this duration appear, they are also dropped.

This proposed buffering scheme is very flexible, can deal with delays and always creates estimates based on the best current knowledge without delay except the processing time of the filter itself. In addition, it is a flexible approach that can be used in combination with various MOT filters, since it purely manages a buffer system and triggers the prediction and update steps. The main drawback is the large required memory for storing all the states and measurements. Due to the possible jumps to previous states, the processing time will also increase in case of out-of-sequence measurements, since some measurements may be processed multiple times.

The buffering scheme is chosen, because this allows to stay independent of the used filter compared to solutions, like the out-of-sequence retrodiction (Bar-Shalom 2002) and is easier to implement in practice. Such retrodiction solutions may be not derived or possible for some types of MOT filters. In addition, it offers outputs with a constant cycle time without delays, that can appear for buffering solutions like the one proposed by Stüker (2003). Overall, processing is achieved without any loss of accuracy in the long term, while fixed cycle times are still guaranteed.

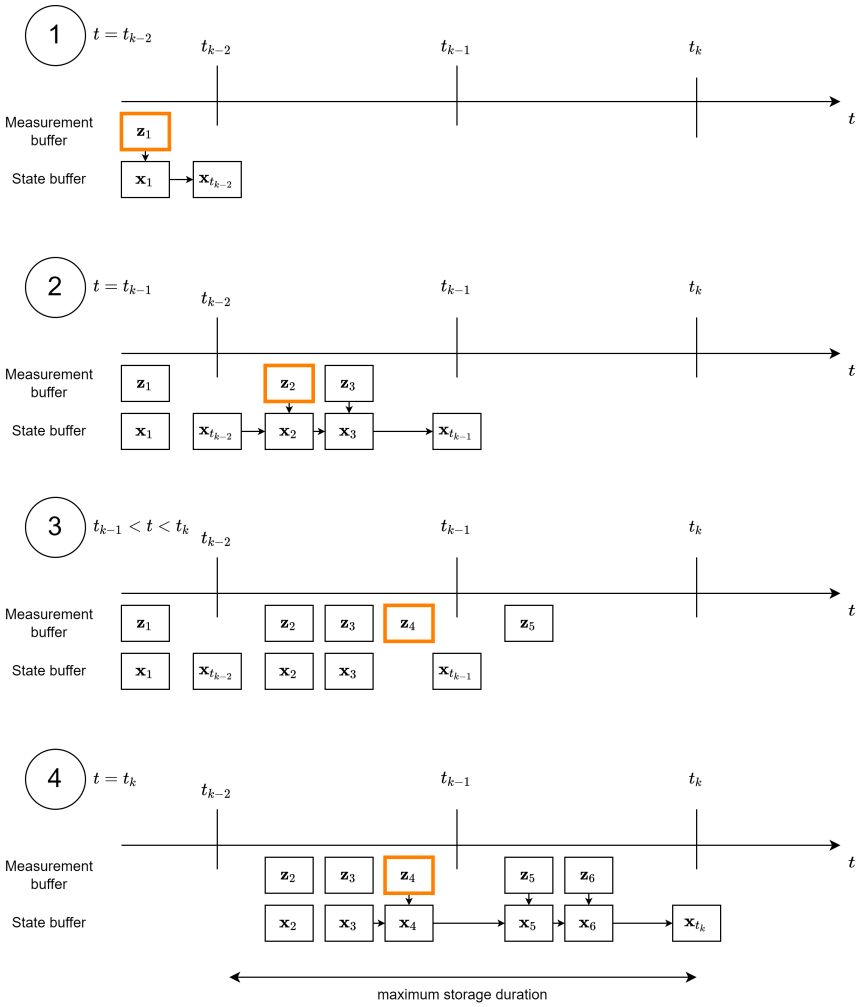


Figure 5.2: Scheme of the temporal management, that is responsible for the correct buffering.

5.3 Track definition and motion model

An ADAS function usually requires information in the form of object tracks, i.e. objects tracked over time, whose properties are estimated. These properties typically include at least the object's current state, the ID, the classification and some sort of confidence value. In this thesis, the i-th tracked object $T_k^{(i)}$ of time step k is therefore represented by a tuple $T_k^{(i)} = (\tau_k^{(i)}, \boldsymbol{x}_k^{(i)}, \boldsymbol{c}_k^{(i)}, p_k^{(i)}(\exists))$ including the track's unique ID $\tau_k^{(i)}$, the state $\boldsymbol{x}_k^{(i)}$, the classification $\boldsymbol{c}_k^{(i)}$ and the probability of existence $p_k^{(i)}(\exists)$, while the set of all n tracked objects at time step k is $T_k = \{T_k^{(1)}, T_k^{(2)}, \ldots, T_k^{(n)}\}$.

The movement and therefore the states of all tracks are estimated using a kinematic state-space model. A two-dimensional model is sufficient to describe the motion of ground objects, like traffic participants. The state vector used to describe a track contains the two-dimensional position p_x and p_y, speed v_x and v_y and acceleration a_x and a_y, as well as the OBB orientation ϕ and OBB dimensions, consisting of length l, width w and height h. These states are visualized in Figure 5.3 while the state vector can be described as follows:

$$\boldsymbol{x}_k = \begin{bmatrix} p_x & p_y & v_x & v_y & a_x & a_y & l & w & h & \phi \end{bmatrix}^T. \tag{5.1}$$

To propagate such a state through time, various kinematic models can be used. Schubert et al. (2008) show, that advanced motion models that also simulate turn maneuvers can improve the tracking performance for road

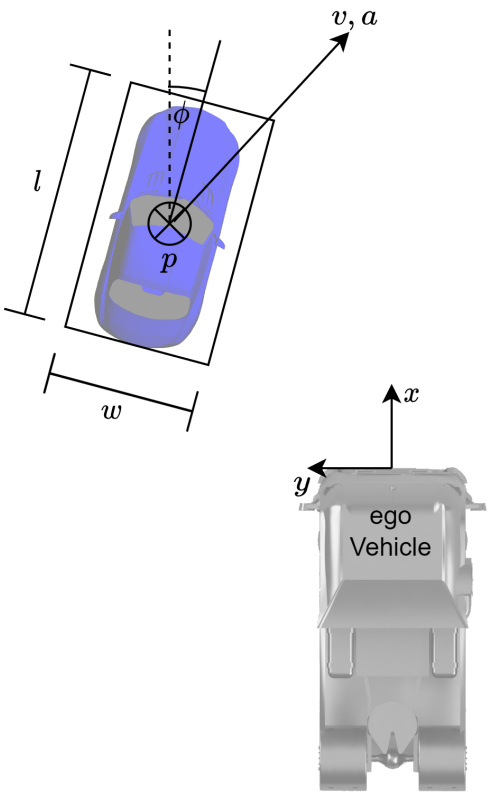


Figure 5.3: Visualization of ego vehicle and one tracked object, all states except h are visualized. Note, that ϕ and Ψ do not necessarily need to align.

users, like vehicles. The best results were achieved by the Constant Turn Rate Acceleration (CTRA) model, which assumes a constant acceleration and a constant turn rate. This was successfully applied to automotive tracking and trajectory prediction applications (Bader 2019; Houenou et al. 2013; Xie et al. 2018).

However, this thesis uses the Constant Acceleration (CA) model, which assumes constant velocities or accelerations without further kinematic relationships, to be as versatile as possible. The CA can model motions of all road users and is not optimized for a specific class. Another advantage is the initialization, where no assumptions have to be made about an initial direction of movement. In addition, it is easily applicable to both the KF and the GM-PHD filter for comparison. Following the time discrete state-space system description from equation (4.5), it can be described by

$$f_{CA}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \begin{bmatrix} p_x + \Delta t v_x + \frac{1}{2}\Delta t^2 a_x + \Delta p_{x,u} \\ p_y + \Delta t v_y + \frac{1}{2}\Delta t^2 a_y + \Delta p_{y,u} \\ v_x + \Delta t a_x \\ v_y + \Delta t a_y \\ a_x \\ a_y \\ l \\ w \\ h \\ \phi + \Delta \phi_u \end{bmatrix}, \tag{5.2}$$

with the ego vehicle's movement induced deltas $\Delta p_{x,u}$, $\Delta p_{y,u}$ and $\Delta \phi_u$. The time $\Delta t = t_{k+1} - t_k$ is the duration between the two time-steps. The deltas are used to compensate the ego vehicle movement and estimate absolute velocities and accelerations. The known ego vehicle speed v_{ego} and yaw-rate ω_{ego} are used:

$$\Delta p_{x,u} = -v_{ego}\Delta t + \sin(\omega_{ego}\Delta t)\omega_{ego}\Delta t p_x + \cos(\omega_{ego}\Delta t)\omega_{ego}\Delta t p_y, \tag{5.3}$$

$$\Delta p_{x,u} = -\cos(\omega_{ego}\Delta t)\omega_{ego}\Delta t p_x + \sin\omega_{ego}\Delta t\omega_{ego}\Delta t p_y, \tag{5.4}$$

$$\Delta \phi_u = -\omega_{ego}\Delta t. \tag{5.5}$$

Although it is shown that the usage of an Interacting Multiple Model (IMM) with multiple motion models can improve the overall tracking performance (Genovese 2001), this approach is not used for the fusion framework, in order to keep the complexity low and to be more flexible in the MOT filter choice.

5.4 Spatial measurement mapping using turning functions

The motion models used in the filters are suitable for tracking points in space. Together with the dimensions and orientation of the state vector from equation (5.1), three-dimensional OBB objects can be tracked. In this case, a direct update by a measurement vector is only possible if it also contains the position x and y, as well as the dimensions l, w and h and orientation ϕ, like $\boldsymbol{z}_k = \begin{bmatrix} p_x & p_y & l & w & h & \phi \end{bmatrix}^T$.

In real-world applications, it is not given, that the OBB dimensions are detected correctly, which induces errors while tracking the center point. A common approach to solve this is using a reference point at the OBB outside (e.g. an edge) instead of the center for the matching and point-tracking to increase robustness (Schueler et al. 2012; Kampker et al. 2018), since many environmental perception sensors naturally recognize parts of the outer surface of objects. However, this approach is usually limited to OBBs.

In real-world applications, several sensors might not provide the measurements in this uniform data format. While many deep-learning-based Lidar and camera detectors extract 3-dimensional OBBs from the raw data, other sensors and detection algorithms create objects with different geometric representations. Radar sensors often create point or L-shape objects, geometric Lidar algorithms may create OBBs with incorrect sizes and centers, projecting camera-based algorithms may create point objects with estimated widths. An overview of these possibilities is given on the left side in Figure 5.4, where multiple possible measurements for a car are shown. In addition, other geometric representations, like ellipses (Granström et al. 2011), polylines (Kraemer et al. 2018) or local occupancy grids (Quehl et al. 2019; Schütz et al. 2014) are possible as geometric representation. Since the existing literature typically uses only one of these representations, a new algorithm is proposed here to fill the gap for multiple representations in a single system. The proposed algorithm is able to match measurements from all the mentioned representations.

If the measurement data is not given as a 3D OBB, it needs to be spatially aligned to the track. The proposed algorithm solves that by converting any geometric representation to a pseudo-measurement $\tilde{\boldsymbol{z}}_k \left(\boldsymbol{x}_{k|k-1}, \boldsymbol{z}_k\right)$ with the data format of a 3D OBB that can be used for the filter update. This is shown on the right side of Figure 5.4, where the predicted state vector $\boldsymbol{x}_{k|k-1}$ is aligned with a point-shape and a wrong sized OBB for a proper update. This alignment creates the pseudo measurement $\tilde{\boldsymbol{z}}_k$ with the data format of a 3D OBB. The calculation process is described in the following sections.

5.4.1 Spatial measurement mapping and pseudo measurement generation

It is assumed that all sensors best detect parts of the outer contours of the ego vehicle facing side, as indicated by the brown line in Figure 5.4 and represent that detection with a geometric description, like points, L-shapes, or OBBs. The proposed algorithm, that generates a pseudo measurement from any of these shapes, works in

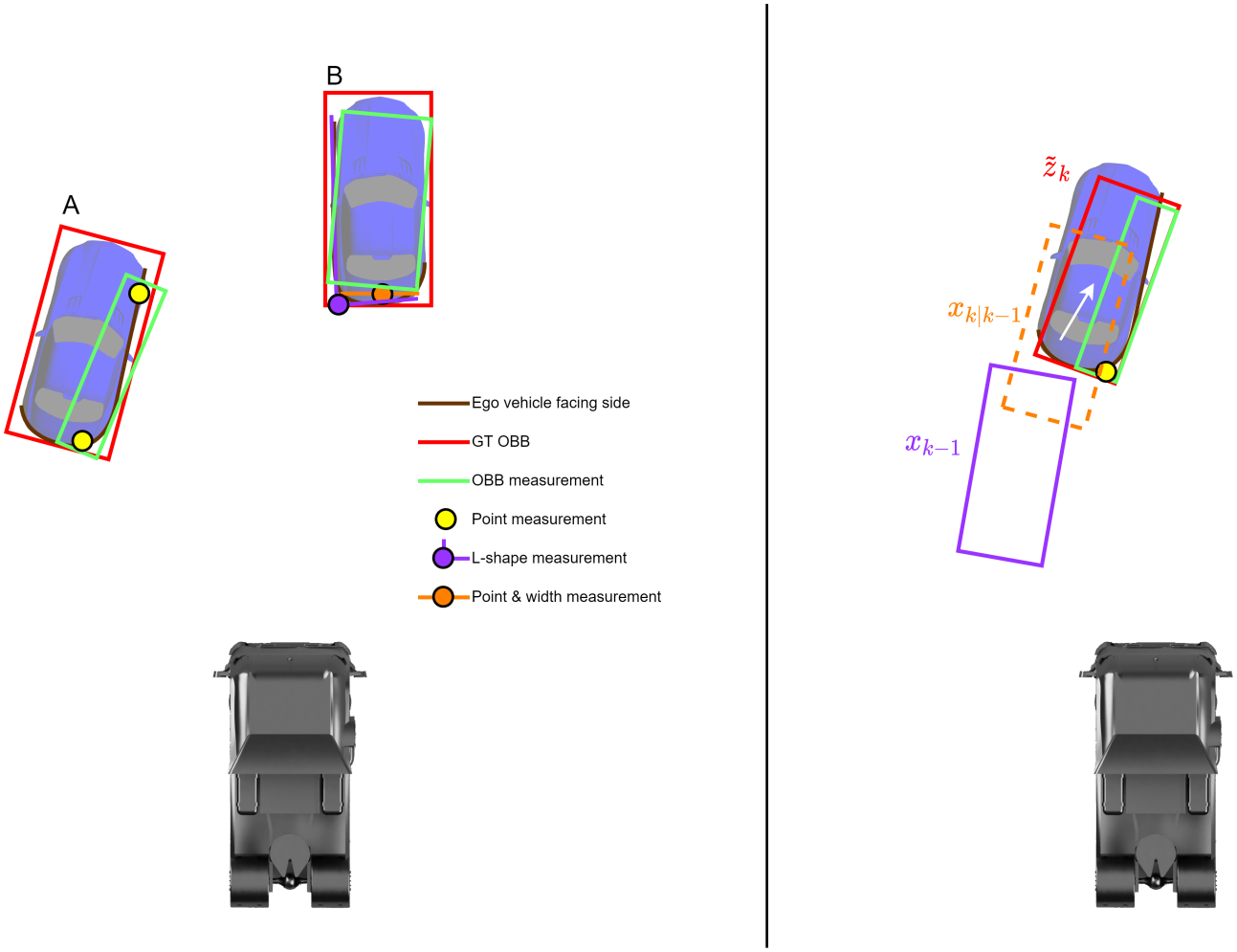


Figure 5.4: Examples of faulty measurements based on real-world targets (left). Example for pseudo measurement creation using point- and wrong sized OBB (right).

2D BEV, while the OBB height is handled separately. Note, that the basic structure of this algorithm should also be applicable to pseudo measurements of other geometric representations than OBBs as well.

The algorithm is illustrated in Figure 5.5 in multiple steps. If the measurement $\boldsymbol{z}_k$ already has the form of a 3D OBB, and it can be assumed that this OBB does represent the full scale object, $\tilde{\boldsymbol{z}}_k = \boldsymbol{z}_k$ applies. Otherwise, the proposed algorithm tries to transform the tracked OBB in a meaningful way to match the shape of the measurement and thus creating the pseudo measurement. The well-known Iterative Closest Point (ICP) algorithm and many of its variants also provide the possibility to register such shapes. However, it requires a good initial estimate and is sensitive to outliers. Furthermore, it is usually used for dense point clouds, where the correct relationships between points are found iteratively. In this application, the contours are sometimes sparse, there may be outliers and non-optimal mappings, and the initial estimate may differ greatly from the result. The process presented here can overcome these disadvantages while achieving a low runtime. However, in pure Lidar-based applications, where the contour of objects is represented using accumulated point clouds, the ICP can be used, as shown by Moosmann and Fraichard (2010).

The overview in Figure 5.5 shows the pseudo measurement generation for a measurement $\boldsymbol{z}_k$ with geometric representation of a polyline. In the first step shown in Figure 5.5, the Visible Convex Hull (VCH) of the contour is calculated based on the given edge points of the polyline. The VCH is a subset of the convex hull V, which is the smallest convex polygon enclosing a given set of points and visualized at the top of Figure 5.5. V can be calculated using the Graham Scan algorithm (Graham 1972), as soon as more than one point is available. If the geometry is not given as a set of points, a point set needs to be sampled from its boundary. This is, for example, the case for parametric representations like ellipses. However, such a sampling is possible for almost every two-dimensional geometric representation. The convex hull V is a closed polyline with m points $\boldsymbol{p}_i$, defined by

$$V = \bigcup_{i=1}^{m-1} \overrightarrow{\boldsymbol{p}_i \boldsymbol{p}_{i+1}} \bigcup \overrightarrow{\boldsymbol{p}_m \boldsymbol{p}_0}. \tag{5.6}$$

The VCH is the part of the convex hull that is on the visible side towards the sensor center $\boldsymbol{p}_c$, as shown in Figure 5.5. Therefore, this subset can be formed using an angle condition between each convex hull segment $\overrightarrow{\boldsymbol{V}_i}$ and the line between the sensor center and segment point $\overrightarrow{\boldsymbol{p}_i \boldsymbol{p}_c}$:

$$VCH = \bigcup_{i=1}^{m} \overrightarrow{\boldsymbol{V}_i}, \quad \forall |\angle \overrightarrow{\boldsymbol{V}_i} \overrightarrow{\boldsymbol{p}_i \boldsymbol{p}_c}| < \pi. \tag{5.7}$$

After the VCH is created for the measurement, the angle is matched with the VCH of the track's prediction $\boldsymbol{x}_{k|k-1}$. Inspired by Veltkamp (2001) and Cohen and Guibas (1997), a "turning function" $\Theta_{VCH}(s)$ is created for both track and measurement, which is defined as the angle θ of the VCH over the arc distance s of the VCH (see step 2 in Figure 5.5). For processing, this is discretized by a defined length Δs. Now the best match of the turning function Θ_X of length S_X and Θ_Z of length S_Z as a function of the offset β is searched, where β is an offset over the arc distance s, as shown in step 3 of Figure 5.5. This matching allows to find the corresponding points of both VCHs, as shown by the green arrows in step 3 of Figure 5.5. The best match is found by numerically minimizing the matching error function $e(\beta)$ with respect to β. This function consists of the absolute integral between both functions in the overlapping area and an additional error term for non-overlapping areas:

$$e(\beta) = \frac{1}{S_C} \int_{\beta>0}^{(\beta+S_X)<S_Z} |\Theta_Z(s) - \Theta_X(s+\beta)| \, ds + e_{out} \tag{5.8}$$

$$S_C = \max\{(S_X + \beta), S_Z\} - \min\{\beta, 0\} \tag{5.9}$$

$$e_{out} = \int_{\beta}^{0} e_0 \, ds + \int_{S_Z}^{\beta} e_0 \, ds \tag{5.10}$$

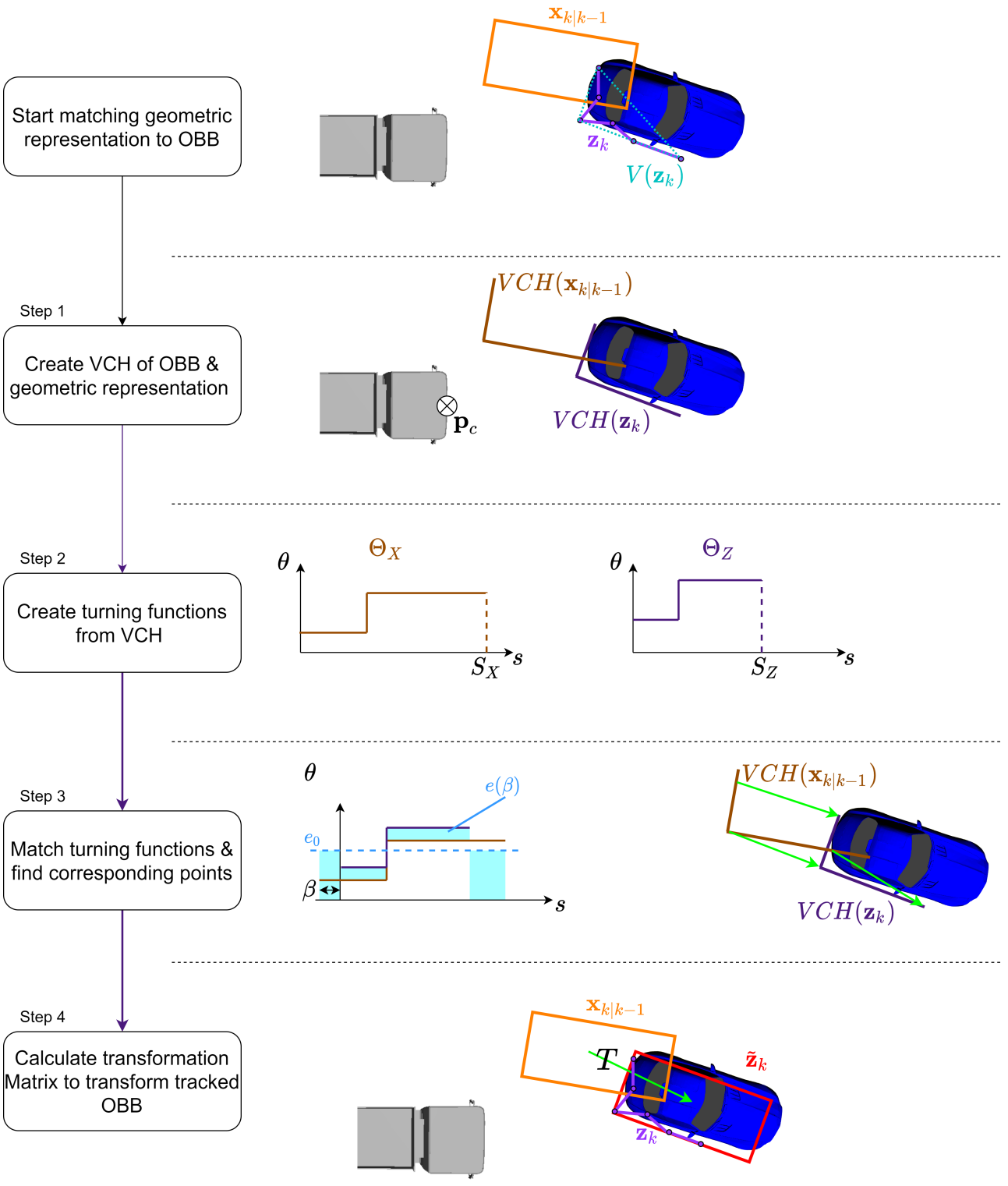


Figure 5.5: Schematic illustration of all steps to generate OBB pseudo measurements from geometric formats. Exemplary done with polyline geometry.

The error term e_{out} calculates an error value for regions that do not overlap to ensure that the track and the measurement have a proper overlap. Therefore, the non-overlapping arc is integrated over the error value $e_0 = \frac{\pi}{2}$. The minimum is found by scanning over $e(\beta)$ in steps of Δs. As a result, pairs of points on the VCH of the track and the measurement can be created.

In some situations, several identical minima of β values can be found. This can happen, for example, if a measurement consisting of a short line is matched, where the turning function therefore only consists of a plateau. In these cases, the minimum with the lowest average distance of the generated point pairs is chosen for further processing to ensure correct behavior.

The chosen pairs of points are represented by the point sets $\boldsymbol{P}_T$ and $\boldsymbol{P}_M$. Based on the Singular Value Decomposition (SVD), the transformation matrix T of the track can be calculated, where $\boldsymbol{c}_P$ is the centroid of the point set P:

$$\boldsymbol{H} = (\boldsymbol{P}_T - \boldsymbol{c}_{P_T})(\boldsymbol{P}_M - \boldsymbol{c}_{P_M})^T \tag{5.11}$$

$$[\boldsymbol{U}, \boldsymbol{S}, \boldsymbol{V}] = SVD(\boldsymbol{H}) \tag{5.12}$$

$$\boldsymbol{R} = \boldsymbol{V}\boldsymbol{U}^T \tag{5.13}$$

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{c}_{P_M} - \boldsymbol{c}_{P_T} \\ 0 & 1 \end{bmatrix} \tag{5.14}$$

If the measurement format is a single point, the point is matched with the closest point on the tracks VCH. In this case, it is assumed that there is no rotation and only the translation between the two points is used. Figure 5.6 shows some matching examples created by this algorithm divided into different types of geometric representations. The pink OBBs are the track predictions, while the blue shapes are VCHs of measurements of arbitrary geometric shapes. As shown, for each of the measurements, a reasonable pseudo measurement in form of an OBB is found and visualized in red. Note, that the orthogonal L-shapes can also be created from OBBs by using the two vehicle facing sides.

Some sensors are able to correctly detect the size of objects, while others only detect parts. If it can be assumed that a measurement represents the correct size of an object, the size of the pseudo measurement is corrected at last. In this thesis, the size is corrected depending on which sensor generates the measurement (see chapter 6.1). If the size is corrected, the length, width and center are changed so that the resulting OBB contains all points of the VCH of the measurement. This is shown by the red dotted OBBs in Figure 5.6, where the dimension correction is shown for polylines and orthogonal L-shapes. It is intuitively logical that a dimension correction does not make sense for certain geometric representations, such as points or simple lines. Overall, this approach offers a very flexible framework for matching measurements of arbitrary data formats to a tracked OBB.

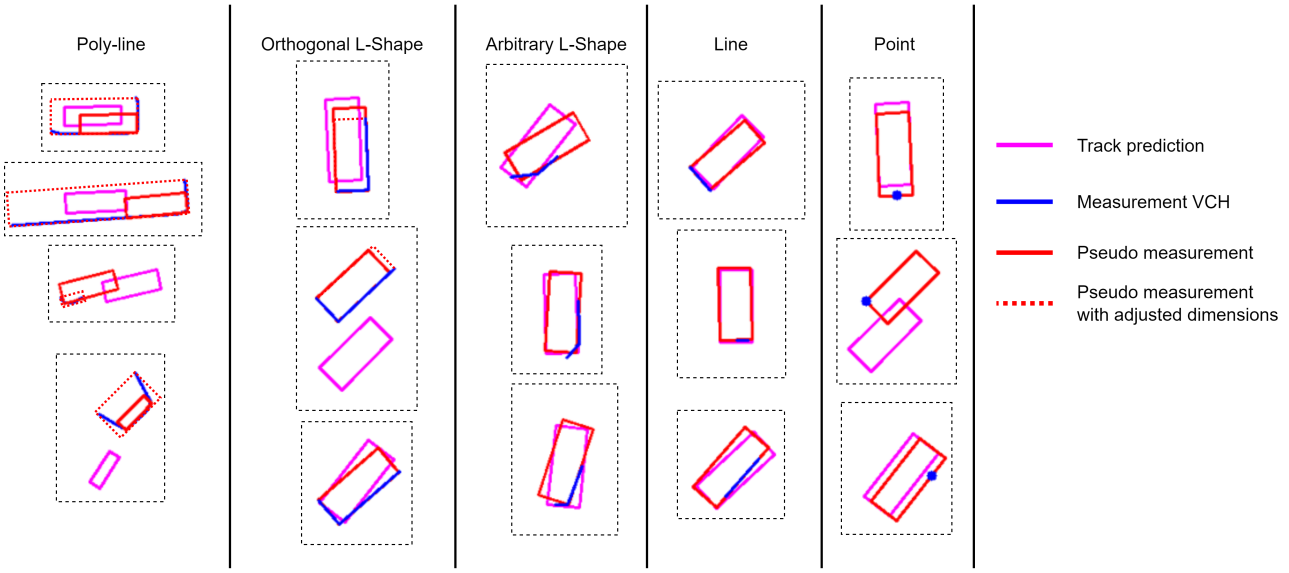


Figure 5.6: Mappings of proposed pseudo measurement generation approach using different input data formats.

5.5 Sensor-based tracking parameter models

The sensor-based tracking parameter models are a central element in the proposed sensor fusions framework because they contain the sensor-specific information necessary to correctly fuse the information. The goal of the proposed parameter models is to describe important MOT filter parameters for various sensors using mathematical models. On the one hand, these models can improve tracking and, on the other, it is necessary to enable fusion using the GM-PHD filter in the presented framework in the first place, as described in chapter 5.5.1.

Properties like the measurement noise or the classification accuracy, as well as physical limitations like the FoV or the maximum detection range, differ from sensor to sensor. For example, cameras typically achieve higher classification accuracies but often have lower FoVs and detection ranges compared to spinning Lidar sensors. Therefore, the relevant tracking parameters need to be modeled separately for each sensor to enable precise fusion. In particular, three parameters vary between different sensors s_p:

- Detection probability $p_{D,k}^{s_p}(\boldsymbol{x})$ of track $\boldsymbol{x}$.
- Clutter density $\kappa_k^{s_p}(\boldsymbol{z})$ of measurement $\boldsymbol{z}$.
- Measurement covariance matrix $\boldsymbol{R}_k^{s_p}$.

Both the detection probability and clutter density have a strong influence on the appearing and disappearing of tracks using the GM-PHD filter. In addition, the detection probability has a considerable influence in the estimation of the existence probability using the methods proposed in chapter 4.3.2.2. Similarly, the measurement covariance matrix is a central influence on the state and covariance estimation.

As illustrated by Figure 5.7, all of these parameters are influenced by multiple external sources. The sensor hardware creates best-case boundary conditions for these parameters. A sensor can, for example, not work outside its FoV and has a maximum detection distance. The parameters of most sensors, however, dynamically change depending on environmental influences, like the weather or daylight. Cameras, for example, work worse at night and Lidar sensors can be disturbed by heavy rain.

The operational domain can also influence the sensor performance and thus its parameters. Radar sensors experience more reflections and therefore have a higher potential for clutter detection in dense or closed environments like cities and tunnels. Most sensors provide higher accuracies in "clean" environments, like highways, since there are fewer objects besides cars that can be misinterpreted.

The measurement covariance matrix $\boldsymbol{R}_k^{s_p}$ is assumed to be constant and experimentally set for each sensor separately in this thesis. Due to the high influence that the clutter density has on the GM-PHD filter and the detection probability on both GM-PHD and existence probability estimation, mathematical models are developed for these two parameters, which include the general sensor properties and influences from the operational domain. In the literature, models for the detection probability have only rarely been investigated and not at all for the clutter density, especially for automotive applications. This thesis therefore develops mathematical models for both parameter in the following chapters and investigates the potential and the influence, especially for multi-sensor applications.

The environmental influences are not included in this thesis due to a lack of data and the complexity of the models across different sensor domains. However, Vargas et al. (2021) show an overview of weather influences on various sensor domains, while Pinchon et al. (2019) show a comparison for passive visual sensors.

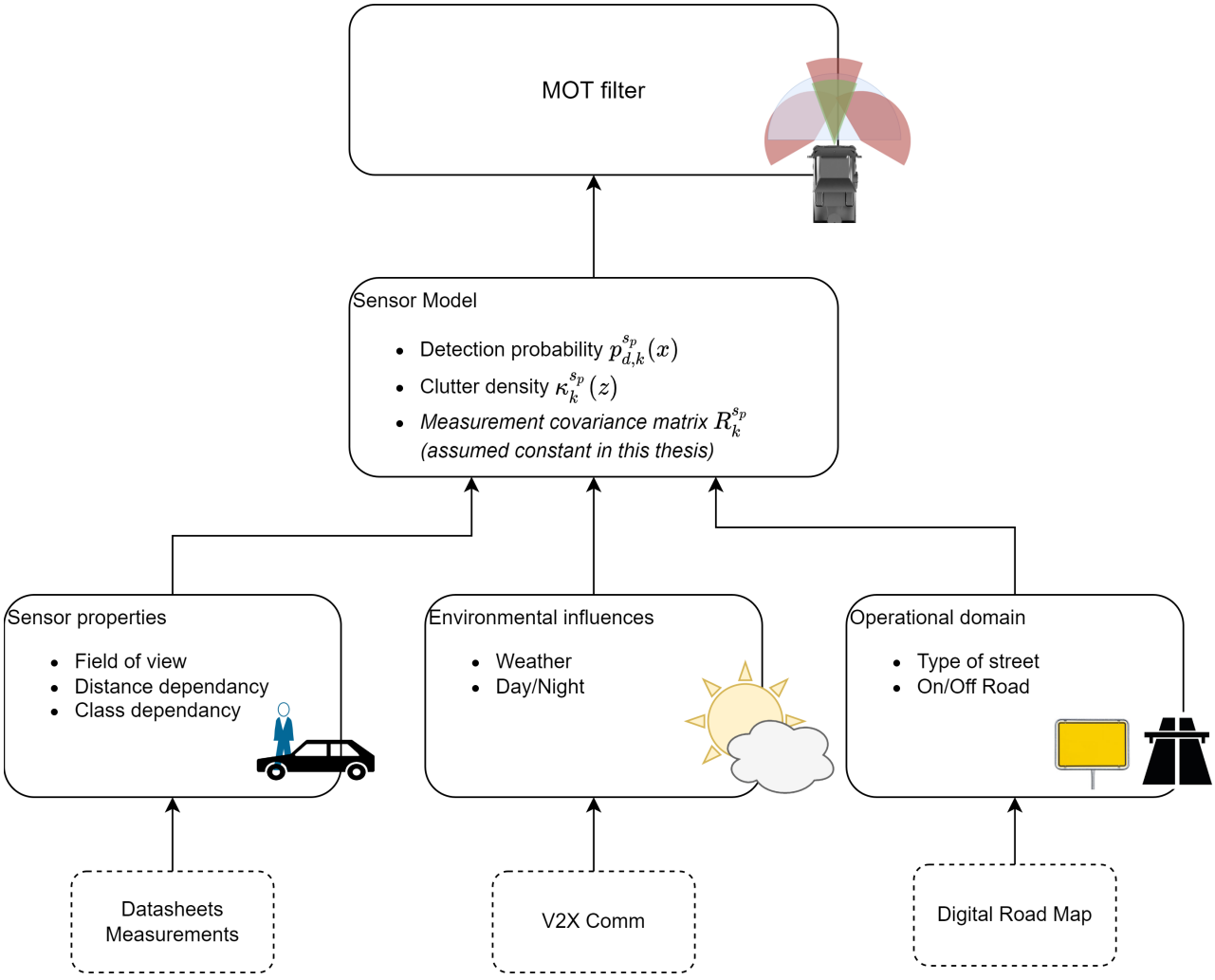


Figure 5.7: Examples of influences on the measurement model and data sources for information on influences.

5.5.1 Detection probability

The detection probability model is an important part for a proper sensor fusion in the proposed framework, since it strongly influences the appearance and disappearance, as well as the estimation of the existence probability of each track. When using a KF, the existence probability of each track can be estimated using the approach from chapter 4.3.2.2. Using this approach, if a track has a high detection probability, but is not detected by the updating sensor, the estimated existence probability decreases. In turn, if a track has a very low detection probability and is not detected, the existence probability does not change a lot. The weights of the GM-PHD filter are affected in a similar way.

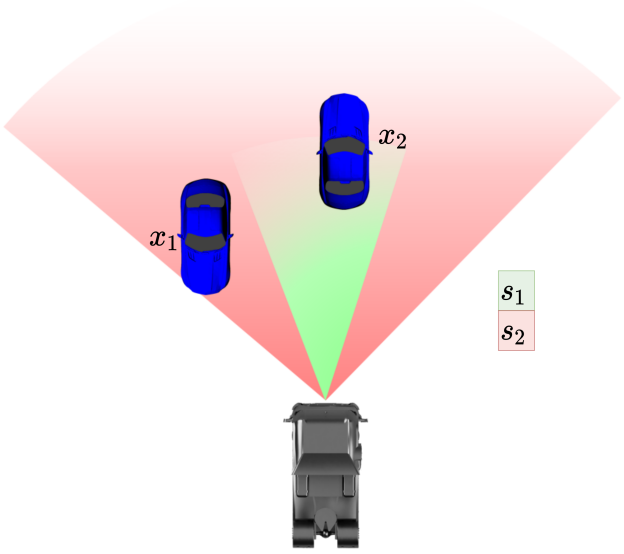


Figure 5.8: Example of two sensors s_1 and s_2 covering different FoVs and two tracked objects.

When using multiple sensors, the detection probability of each track is different for each sensor. This becomes obvious when looking at Figure 5.8, where the FoVs of two sensors s_1 and s_2 are visualized and the opacity represents their ability to detect objects, which is decreasing with higher distances. The detection probabilities of the tracked objects $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ vary for both sensors depending on the position. Sensor 1 is unable to detect track 1 since it is outside the FoV, which means the detection probability $p_{D,k}^{s_1}(\boldsymbol{x}_1) = 0$. Similarly, track 2 is at the edge of the range of sensor 1, which means $p_{D,k}^{s_1}(\boldsymbol{x}_2)$ will be low, while the detection probabilities of both tracks are higher for sensor 2. If the detection probability would be modeled constant, as it is usually done for single-sensor systems, the existence probabilities and weights of tracks that are not in the shared FoV of all sensors, would be vastly underestimated and the tracking would not work properly.

This is shown by an experiment in Figure 5.9a, in which an overtaking vehicle drives through several sensor FoVs. However, without consideration of the FoV in the detection probability model, stable tracking is only achieved in the overlapping FoV of all sensors (marked with orange circle). Since the detection capabilities of the vehicle perception sensors typically are different with respect to the distance, a similar effect occurs at higher distances, where some sensors are not anymore capable of detecting an object while others still can. Figure 5.9b shows an example track, whose trajectory is moving away from the ego vehicle. At a certain point it is outside the capabilities of all sensors except the LRR. If the detection probability is modeled with respect to the distance, the tracking is still possible at higher distances. If constant values are used, the track is lost when it is not anymore detected by the most sensors (marked with orange circle).

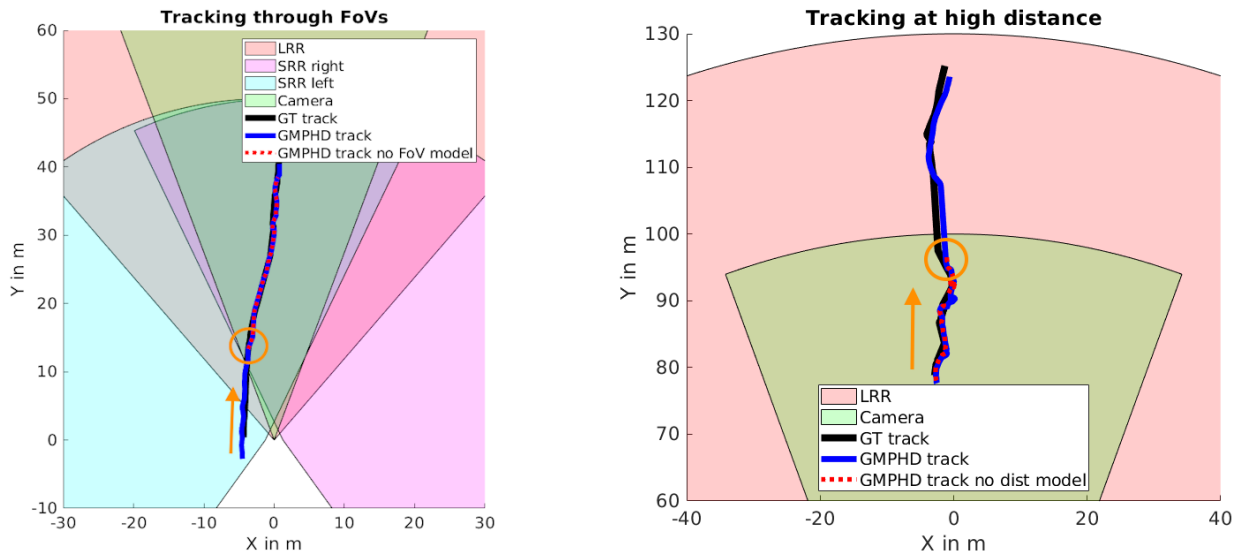
To solve the problems of partially overlapping FoVs, Vasic and Martinoli (2015), Granström et al. (2014) and Lindenmaier et al. (2022) model the detection probability according to FoV to solve tracking issues in

the boundary areas. Vasic and Martinoli (2015) and Granström et al. (2014) include the sensor range to this model as well, without modeling the distance dependent changes within the maximum sensor range. Chen et al. (2018) and Hendeby and Karlsson (2014) propose distance dependent models for sonar applications that are not directly applicable to automotive sensors and therefore highlight the need to provide such models in automotive applications. In addition to the development of such models, this thesis provides an evaluation based on real-world datasets to show the influence on a large database.

The overview in Figure 5.7 indicates that, in addition to the FoV and distance, other aspects may also influence the probability of detection and thus on the tracking performance. The following list shows the influences included in the proposed model, which is explained in detail in the next section:

- Track distance.
- Track classification.
- Operation domain.
- On/Off-road.

Vasic and Martinoli (2015), Granström et al. (2014) and Törő et al. (2021) furthermore show methods for modeling the detection probability depending on occlusion. However, occlusion is handled by the confirmation strategy in chapter 5.9 in this thesis.



(a) Overtaking car moving in direction of the orange arrow crossing multiple sensor FoVs. A stable tracking without FoV model is reached at the overlap of all sensor FoVs.

(b) Car moving away in the direction of the orange arrow at higher distance. The track is lost at orange circle without the distance model for the detection probability.

Figure 5.9: Example situations showing benefits of sensor-based parameter models.

5.5.1.1 Detection probability model

The detection probability model is a mathematical function that describes the detection probability of a track as precisely as possible depending on the sensor FoV, the distance, the class, the operation domain and if the track is on or off the road. Note, that due to a lack of measurement data, other environmental influences, like the weather, are not modeled in this thesis but still mentioned for future implementation. While environmental influences and the operational domain may change over time, the sensor properties remain constant. Therefore,

a mathematical model $p_{Dm,k}^{(s_p)}(\boldsymbol{x}_k)$ for the detection probability of track $\boldsymbol{x}_k$ is defined based on experimental measurements for each sensor s_p. Since each sensor has different detection capabilities depending on the track's class, a detection probability model $p_{Dm,k}^{(s_p,C_j)}(\boldsymbol{x}_k)$ is created for each class C_j separately. Based on the class probabilities of a track, the overall detection probability can be estimated.

Next to the FoV, which obviously sets limits to the detection probability, the position is a key parameter. Due to the decreasing resolution at higher distances, the detection probability will decrease with distance for all types of used perception sensors. In addition, some sensors, like automotive radars, can have direction-depending sensitivities and maximum detection ranges which can result in more complex detection probability models. To keep it generally applicable to different types of sensors, the proposed model is based on the distance and a radial FoV only.

In addition to the sensor properties, the operational domain is taken into account. Using the GNSS position and OpenStreetMap, the type of street closest to the ego vehicle can be used as an indicator of the operational domain. Each operational domain is evaluated separately and compared to the standard model. A reasonable fit is achieved by applying an offset for each operational domain to the standard model. The street types „highway_tertiary“ and the combination of „highway_primary“ and „highway_motorway_primary“ are used in this thesis, but others may be used in addition. Other street types are not considered, since too little data for analysis is available. For these street types, the detection probability is not influenced. According to the measurements, the Lidar-based detections are less influenced by different operation domains compared to the camera-based detections.

While roads are assumed to be "clean" environments mainly used by road users, areas next to the road might be more complex, with buildings, constructions, fauna and other types of unknown objects. Therefore, the detection capabilities might vary between tracks on the road and the ones off the road, that might be parking cars or pedestrians on sidewalks. Based on the road width provided by OpenStreetMap and the accurate GNSS position, each track can be evaluated to be on the road or off the road. An offset is now used to distinguish between the detection probability of tracks on and off the road.

Overall, the formula for the detection probability model $p_{Dm,k}^{(s_p)}(\boldsymbol{x}_k)$ is as follows:

$$p_{Dm,k}^{(s_p)}(\boldsymbol{x}_k) = \sum_{j \in C} p_{Dm,k}^{(s_p,C_j)}(\boldsymbol{x}_k) p_{\boldsymbol{x}_k}(C_j), \tag{5.15}$$

where $p_{\boldsymbol{x}_k}(C_j)$ is the track's probability to be of class j. Each of the classes models is modeled following:

$$\begin{aligned}
p_{Dm,k}^{(s_p,C_j)}(\boldsymbol{x}_k) &= (p_{dist}^{(s_p,C_j)}(\boldsymbol{x}_k) + p_{road}^{(s_p,C_j)}(\boldsymbol{x}_k) + p_{OD}^{(s_p,C_j)}(\boldsymbol{x}_k)) p_{FoV}^{(s_p,C_j)}(\boldsymbol{x}_k), \\
p_{dist}^{(s_p,C_j)}(\boldsymbol{x}_k) &= k_2^{(s_p,C_j)} d(\boldsymbol{x}_k)^2 + k_1^{(s_p,C_j)} d(\boldsymbol{x}_k) + k_0^{(s_p,C_j)}, \\
p_{road}^{(s_p,C_j)}(\boldsymbol{x}_k) &= \begin{cases} p_{OnRoad}^{(s_p,C_j)}, & \text{if } \boldsymbol{x}_k \text{ on road} \\ p_{OffRoad}^{(s_p,C_j)}, & \text{if } \boldsymbol{x}_k \text{ on road} \end{cases}, \\
p_{OD}^{(s_p,C_j)}(\boldsymbol{x}_k) &= \text{operational domain}, \\
p_{FoV}^{(s_p,C_j)}(\boldsymbol{x}_k) &= \begin{cases} 0, & \boldsymbol{x}_k \text{ out FoV} \\ 1, & \boldsymbol{x}_k \text{ in FoV} \end{cases}.
\end{aligned} \tag{5.16}$$

In this description, $p_{dist}^{(s_p,C_j)}(\boldsymbol{x}_k)$ is the second order polynomial describing the distance dependency with $d(\boldsymbol{x}_k) = \sqrt{p_x^2 + p_y^2}$ being the distance of the track's OBB center coordinates p_x and p_y. $p_{road}^{(s_p,C_j)}(\boldsymbol{x}_k)$ is the offset for the track being on or off the road, $p_{OD}^{(s_p,C_j)}(\boldsymbol{x}_k)$ the offset for the operational domain and $p_{FoV}^{(s_p,C_j)}(\boldsymbol{x}_k)$

Table 5.1: Operational domain parameters for detection probability of class "Car".

Street type	Offset Camera	Offset Lidar
highway_primary & highway_motorway_primary	0.15	0.006
highway_secondary	0.13	0.008
highway_tertiary	-0.17	-0.069
highway_unclassified & highway_residential	-0.04	0.038
Others	0	0

setting the limit for the sensors FoV. This mathematical model provides a flexible framework to include multiple external influences and thus increase the accuracy of the detection probability. Other influences, such as the weather or the daylight situation, can be added by additive variables in the future.

To verify this model and calculate the parameter values, the detection probabilities are experimentally determined for the class "Car". Using the Point-RCNN (Shi et al. 2019) detector for Lidar and Yolo-Mono-3D (Liu et al. 2021) for monocular camera, the detection probability is calculated using the training data from the KITTI tracking dataset: For each time-step, each of the ground truth tracks is compared with the respective measurements. If a measurement with an overlapping OBB is found for a ground truth track, it is counted as detected. Now, the percentage of detected tracks within the sensor FoV is calculated for distance sections of 10 m to obtain distance dependent detection probabilities.

As shown on top of Figure 5.10, the second order polynomial used by the model can accurately represent the distance dependent detection probability for both camera and Lidar. The BEV result of this polynomial model in combination with the sensor's FoV is shown at the bottom of Figure 5.10 and compared to the measured result.

In Figure 5.11, the distance offset models for the different operation domains are visualized on top, together with the measurements for the camera. Below, the offset model used to distinguish between tracks on and off the road is visualized. The plots show that an offset model can adequately represent the influence of both operation domain and if the track is on- or off-road. The offsets for the operational domain are summarized in Table 5.1.

Note, that the measurements show a drop in detection probability for very close ranges. The main reason for this seems to be the FoV, as many nearby objects are truncated in the camera image and partially occluded in the Lidar image. Therefore, the first distance chapter of 0-10 m is not taken into account to fit the model.

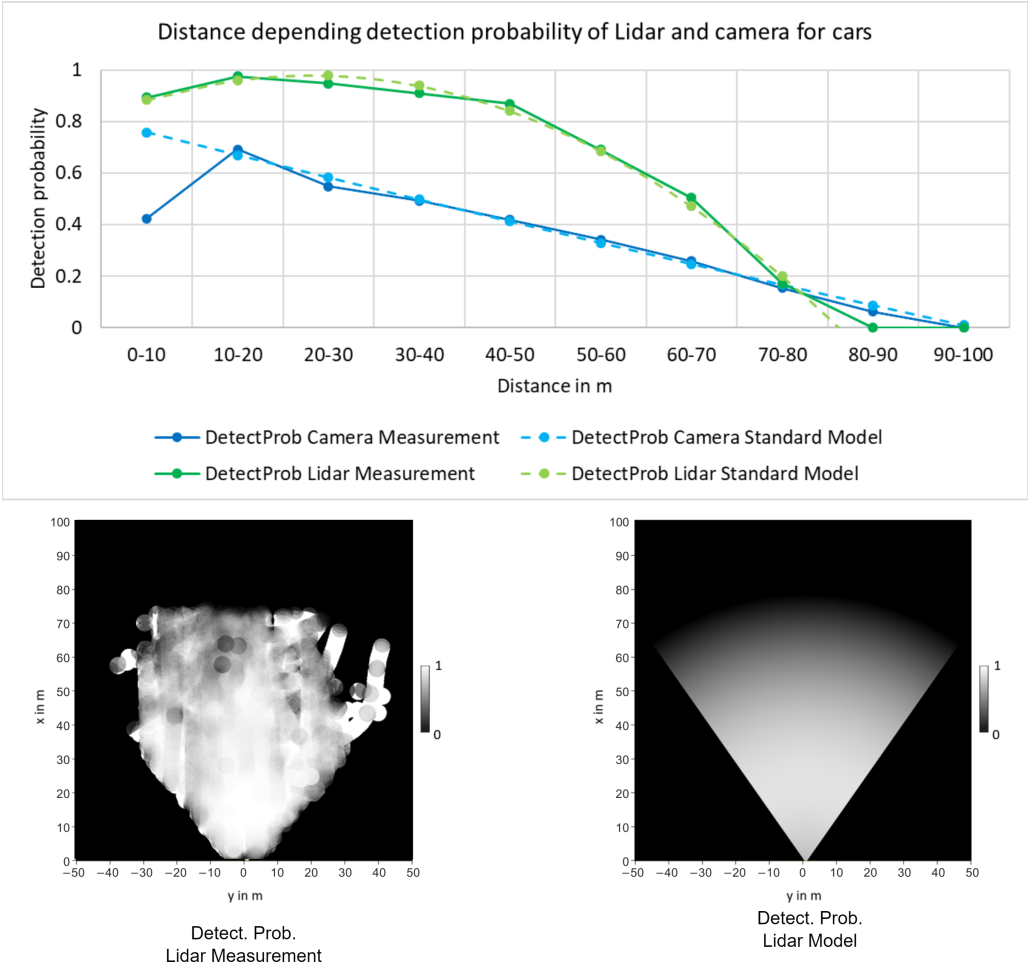


Figure 5.10: Detection probability distance model for class "Car". On top, detection probability models of camera and Lidar with respect to the distance is shown, on bottom the detection probability measurements and model are shown in BEV.

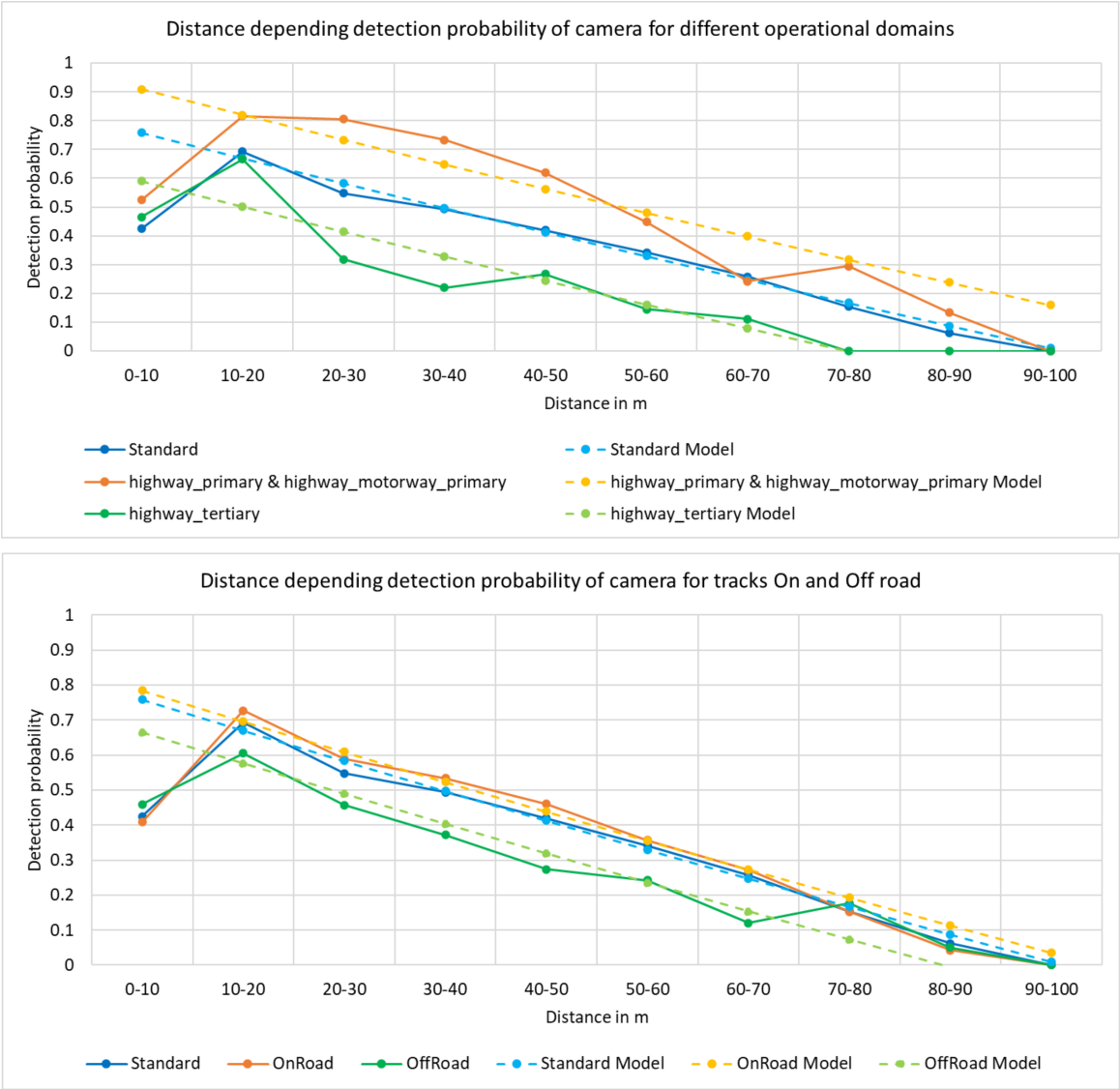


Figure 5.11: Models of detection probability for camera detector in different operational domains and tracks on/off the road for class "Car".

5.5.1.2 Detection probability implementation

When using a traditional KF approach, the application of the detection probability $p_{Dm,k}^{(s_p)}(\boldsymbol{x}_k)$ of track $\boldsymbol{x}_k$ is straightforward, since each track is handled separately. Therefore, the detection probability $p_{D,k}$ in equation (4.18) of chapter 4.3.2.2 changes to $p_{D,k} = p_{Dm,k}^{(s_p)}(\boldsymbol{x}_k)$.

Since the GM-PHD filter in contrast propagates the PHD of an RFS, this cannot directly be applied. The GM-PHD filter uses the detection probability $p_D(\boldsymbol{x})$ in the update step, but it is assumed to be constant with $p_D(\boldsymbol{x}) = p_D$ in the standard implementation (Vo and Ma 2006). Although there is a solution with the detection probability modeled as a mixture $p_{De,k}(\boldsymbol{x})$, as shown in chapter 4.4.4.4, this will greatly increase the computational requirements due to the high number of generated Gaussian components.

Since the literature (Lindenmaier et al. 2022; Granström et al. 2014; Vasic and Martinoli 2015; Törő et al. 2021; Chen et al. 2018; Hendeby and Karlsson 2014) and the experiments of chapter 6.7 suggest, that a non-constant model still works in real-world systems, a more simple solution is desired. Hendeby and Karlsson

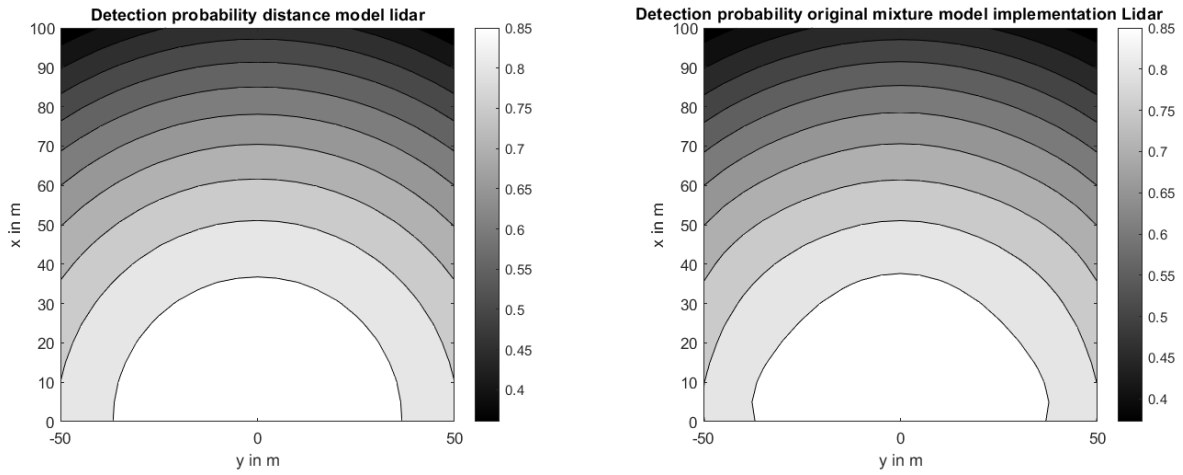


Figure 5.12: Comparison of detection probability model $p_{Dm,k}^{s_p}(\boldsymbol{x}_k)$ (left) and the mixture approximation $p_{De,k}(\boldsymbol{x})$ using 7 Gaussian components (right).

(2014) in addition argue that such a simplification can be described valid, if the detection probability varies slowly compared to the Gaussian components of the intensity function, which is the case for all regions except the FoV edges in the model. Therefore, the detection probability is modeled as

$$p_{D,k} = p_{Dm,k}^{(s_p)}(\boldsymbol{m}_{k|k-1}^{(j)}) \tag{5.17}$$

in this thesis.

In addition, a comparison to the mixture implementation of chapter 4.4.4.4 is provided. The model parameters are created based on the detection probability model $p_{Dm,k}^{s_p}(\boldsymbol{x}_k)$ from the previous section, where the difference between $p_{De}(\boldsymbol{x})$ and $p_{Dm,k}^{s_p}(\boldsymbol{x}_k)$ is minimized using numerical optimization. For this approximation, the offset values for operational domain and on/off-road are set to zero. Figure 5.12 shows the comparison of both the original detection probability and derived mixture model with 7 Gaussian components. With an increased number of components, the accuracy increases as well. However, the computational complexity of the update step also increases. Using this second implementation, an experimental comparison of the performance and computational requirements of both detection probability implementations are provided in chapter 6.7.4, which proves the efficiency of the simplified version.

5.5.2 Clutter density

Similar to the detection probability, the clutter density also has an influence on the intensity function. Although the influence is less critical, as it only comes into play when measurements are available and not when there are no detections (for example outside the FoV), it can still influence the performance of the tracker. As there is no literature investigating the influence of the clutter density for automotive multi-sensor systems, this thesis fills the gap by proposing a mathematical model and evaluating it based on real-world datasets.

The clutter density describes the number of expected clutter objects per area. In contrast to the detection probability, the GM-PHD standard filter equations do not have the assumption of a static value. Therefore, a mathematical model for the clutter density can be used directly.

The clutter density can be modeled similar to the detection probability. Therefore, the mathematical model for $\kappa_k^{(s_p)}(\boldsymbol{z}_k)$ is experimentally defined based on the same influences. For each class C_j a separate model exists, and the overall clutter density is composed using a weighted sum of the individual class functions.

The distance dependency is experimentally determined using the Point-RCNN (Shi et al. 2019) detector for Lidar and Yolo-Mono-3D (Liu et al. 2021) for monocular camera detection on the training data of the KITTI tracking dataset. The detections are compared to the GT objects for each time-step. If no overlapping GT object is found for a detection, it is counted as clutter. Figure 5.13 shows the measured clutter density of the class "Car" for both Lidar and camera. As shown, the clutter density has a peak value for medium distances and appears to be lower for both near and far distances. This behavior can be modeled using a sinusoidal function, which leads to the approximations shown in Figure 5.13.

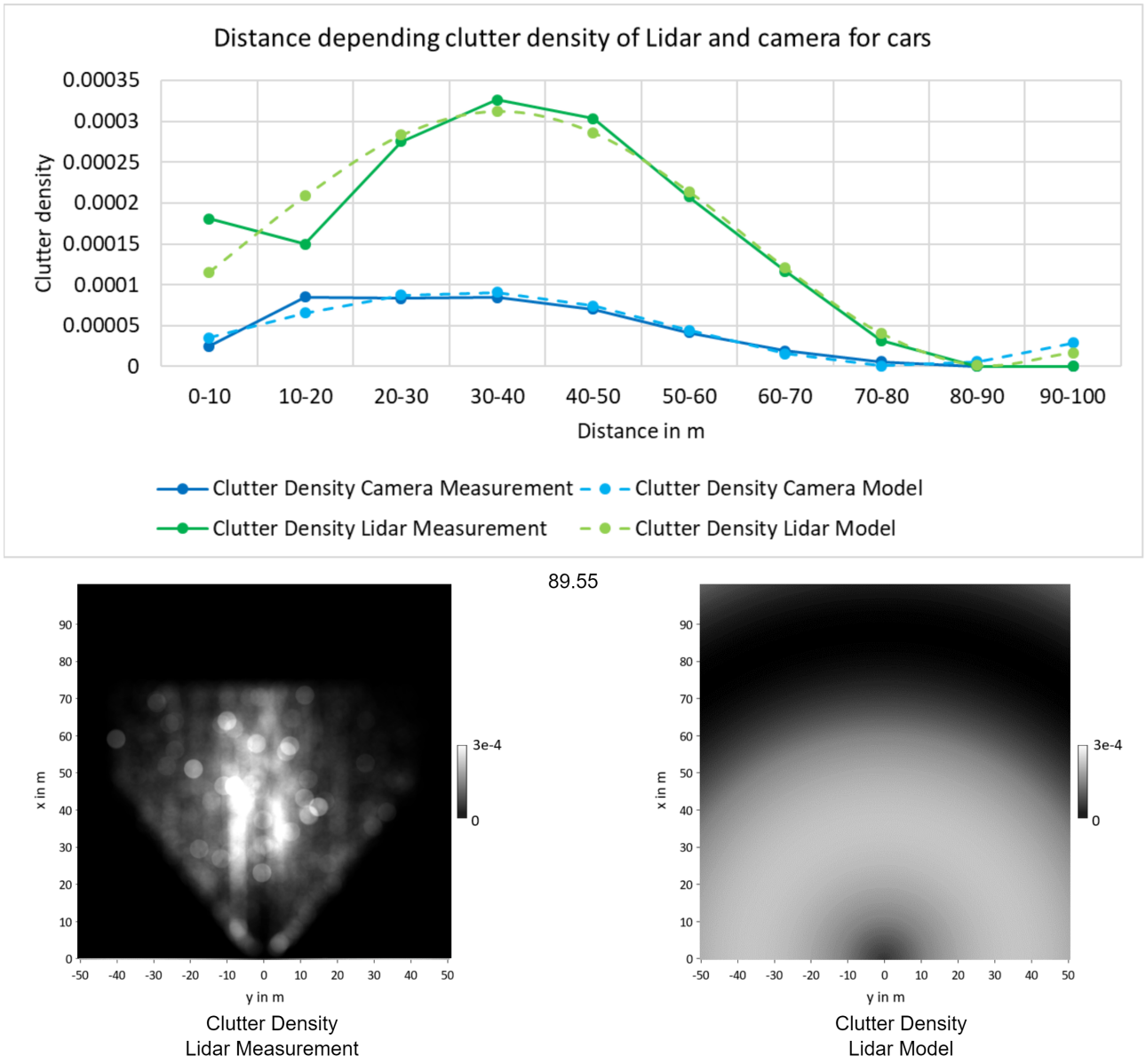


Figure 5.13: Clutter density distance model for class "Car". On top, the models of camera and Lidar with respect to the distance are shown, on bottom the Lidar measurement and model are shown in BEV.

The influences of the operation domain and objects on/off the road are modeled, too. Figure 5.14 shows the results of the measurements for camera detections of the class "Car". In contrast to the detection probability, the differences between operational domains and On/Off the road cannot be modeled by a constant offset. Instead, a constant factor shows a more accurate representation, as shown by the results in Figure 5.14. The factors of the operation domain are also shown in Table 5.2. The operational domain's influence on the clutter density appears to be higher for the camera, but still measurable for the Lidar-based detections.

Figure 5.13 shows that the mathematical model can reproduce the clutter density over the distance well. However, Figure 5.14 clearly shows that the modeling for individual operational domains introduces larger deviations. The clutter density peak value and the medium distance change with differen operational domains, while only the peak value can be modeled by the factor of the model. On-road and on highways, the peak has a larger mean distance than off-road and on tertiary roads. However, in order not to increase the complexity of the mathematical model too much, only the factor is used.

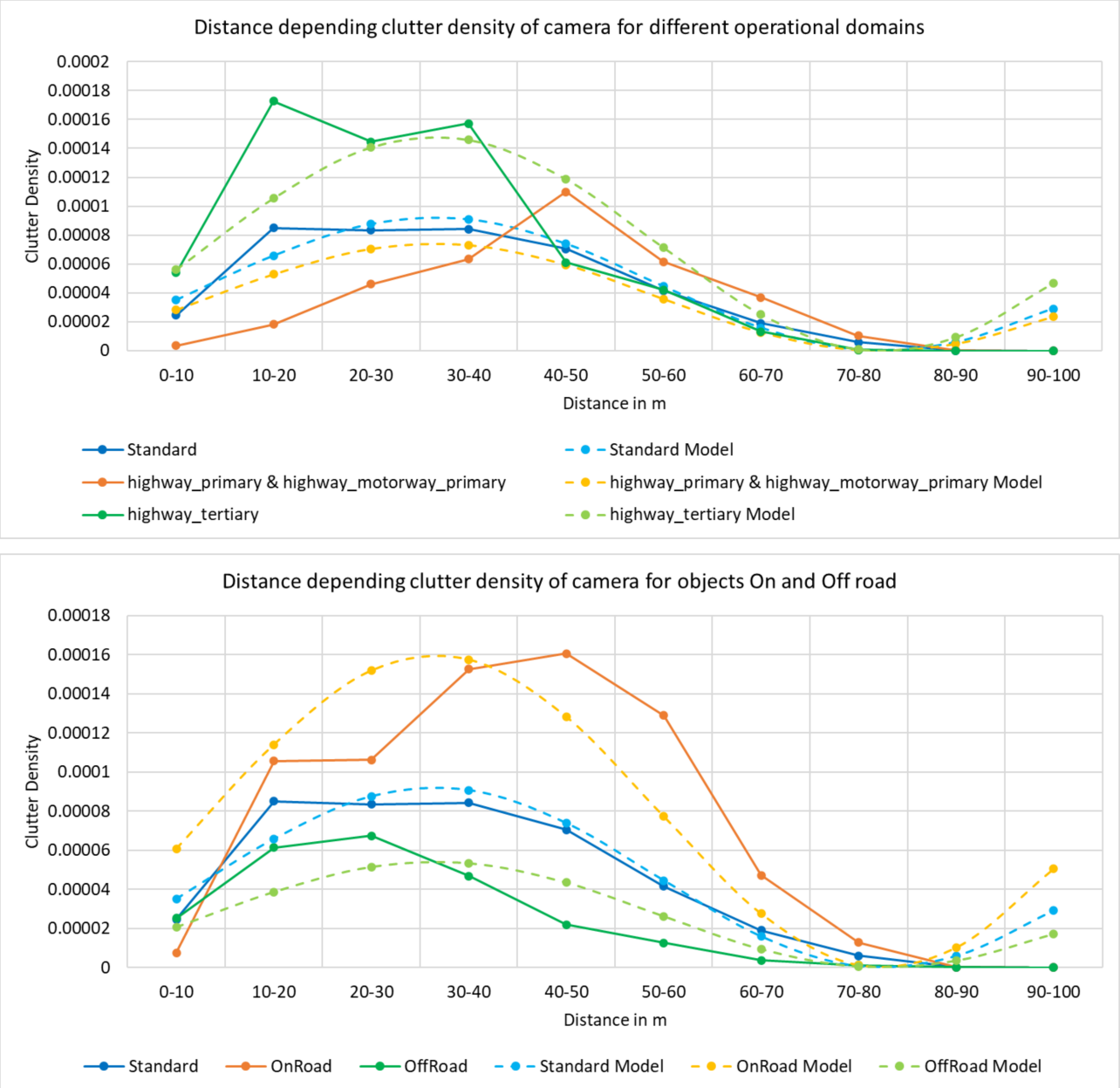


Figure 5.14: Models of clutter density for camera detector in different operational domains and tracks on/off the road for class "Car".

Table 5.2: Operational domain parameters for clutter density of class "Car".

Street type	Factor Camera	Factor Lidar
highway_primary & highway_motorway_primary	0.81	0.93
highway_secondary	0.71	0.59
highway_tertiary	1.61	1.06
highway_unclassified & highway_residential	1.45	1.18
Others	0	0

The overall mathematical description for the clutter density $\kappa_k^{(s_p)}(\boldsymbol{z}_k)$ of measurement $\boldsymbol{z}_k$ is described by

$$\kappa_k^{(s_p)}(\boldsymbol{z}_k) = \sum_{j \in C} \kappa_k^{(s_p,C_j)}(\boldsymbol{z}_k) p_{\boldsymbol{z}_k}(C_j), \tag{5.18}$$

where $p_{\boldsymbol{z}_k}(C_j)$ is the measurement's probability to be of class j and $\kappa_k^{(s_p,C_j)}(\boldsymbol{z}_k)$ is the clutter density of class j, which is described by

$$\begin{aligned}
\kappa_k^{(s_p,C_j)}(\boldsymbol{z}_k) &= \kappa_{dist}^{(s_p,C_j)}(\boldsymbol{z}_k)\kappa_{road}^{(s_p,C_j)}(\boldsymbol{z}_k) + \kappa_{OD}^{(s_p,C_j)}(\boldsymbol{z}_k), \\
\kappa_{dist}^{(s_p,C_j)}(\boldsymbol{z}_k) &= k_0 \sin(k_1 d(\boldsymbol{z}_k) + k_2) + k_0, \\
\kappa_{road}^{(s_p,C_j)}(\boldsymbol{z}_k) &= \begin{cases} \kappa_{OnRoad}^{(s_p,C_j)}, & \text{if } \boldsymbol{z}_k \text{ on road} \\ \kappa_{OffRoad}^{(s_p,C_j)}, & \text{if } \boldsymbol{z}_k \text{ on road,} \end{cases} \\
p_{OD}^{(s_p,C_j)}(\boldsymbol{z}_k) &= \text{operational domain.}
\end{aligned} \tag{5.19}$$

Here, $\kappa_{dist}^{(s_p,C_j)}(\boldsymbol{z}_k)$ is the distance-depending sinusoidal model, $\kappa_{road}^{(s_p,C_j)}(\boldsymbol{z}_k)$ the factor describing the dependency of the measurement being on or off the road and $p_{OD}^{(s_p,C_j)}(\boldsymbol{z}_k)$ the factor for the current operational domain. Overall, this model is a novel approach to improve the tracking performance of GM-PHD filters by taking into account the non-constant clutter density of real-world sensors.

5.6 Classification fusion

Perception sensors typically estimate the class of all detected objects. This information can be used to continuously update and correct the class of tracked objects. The fusion of the classification must be done in a meaningful way because ADAS functions rely on the correct class of the objects. Depending on the sensor, the quality of the classification can vary vastly: While cameras are typically excellent at estimating the correct class, radar sensors are struggling. Therefore, sensors with high confidence should be weighted more during the fusion. To achieve that, a classification confidence $c_{conf}^{(s_p)} \in [0,1]$ is assigned to each sensor s_p, representing the trust in the classification result. The classification fusion is done in the update step of a MOT filter when a sensor delivers new measurements. For each of the measurement $\boldsymbol{z}_j$, a class probability vector $\boldsymbol{c}_k^{(\boldsymbol{z}_j)} = \begin{bmatrix} p(\boldsymbol{z}_j|C_{1,k}) & \dots & p(\boldsymbol{z}_j|C_{n,k}) \end{bmatrix}$ is provided by the sensor or created based on the classification and the confidence. This is used to estimate the class probability vector $\boldsymbol{c}_k^{(i)} = \begin{bmatrix} p(C_{1,k}|Z_k) & \dots & p(C_{n,k}|Z_k) \end{bmatrix}$ of each track T_i at time-step k. Note, that depending on the approach, this class probability vector needs to be evaluated as pseudo-probability vector, since it does not represent the true probabilities, but rather a score.

In this section, four approaches to fuse the classification are proposed, while the results are compared in chapter 6. This thesis therefore provides an overview of the suitability of various methods for the classification fusion within a framework in which the GM-PHD filter is used. Since the classification is typically not considered in this context in the literature, this comparison can provide guidance to choose an appropriate approach. Most of the proposed approaches use the confidence values. In this thesis, the set of classes

$C = \{C_{Bg}, C_{Car}, C_{Ped}, C_{Cyc}\}$ with Background, Car, Pedestrian and Cyclist is used, while in general any class set could be used.

The max-confidence classification fusion is a simple fusion scheme, where the statement with the currently highest confidence is used as the result. For each track, the current maximum confidence $c_{conf,max,k}$ is stored. When the track is updated, first the current maximum track confidence is updated using a damping ratio $c_{conf,d}$. Next, the confidence of the updating sensor $c_{conf}^{(s_p)}$ is compared with the current maximum confidence and if it exceeds this value, the track is set to the updates class and confidence:

$$c_{conf,max,k} = c_{conf,max,k-1} * c_{conf,d}, \tag{5.20}$$

$$c_{conf,max,k} = \begin{cases} c_{conf}^{(s_p)}, & c_{conf,max,k} < c_{conf}^{(s_p)} \\ c_{conf,max,k}, & c_{conf,max,k} >= c_{conf}^{(s_p)} \end{cases}, \tag{5.21}$$

$$\boldsymbol{c}_k^{(i)} = \begin{cases} \boldsymbol{c}_k^{(\boldsymbol{z}_j)}, & c_{conf,max,k} < c_{conf}^{(s_p)} \\ \boldsymbol{c}_{k-1}^{(i)}, & c_{conf,max,k} >= c_{conf}^{(s_p)} \end{cases}. \tag{5.22}$$

The advantage of this approach is an easy implementation, as well as the simple parameter tuning, since only the sensor confidences and the damping ratio can be adjusted. When having measurements from multiple sensors, it is likely that the low confidence ones will not have an impact at all. One disadvantage is, that the approach does not use any filtering over time, so errors of the highest confidence sensor will directly transform to errors in the track's class.

The voting scheme for classification fusion is a simple, additive solution for classification estimation. The approach proposed in this thesis is inspired by neural network ensemble approaches, that use multiple classifiers adding up for the final result, as shown by Brock et al. (2016). Here, the probabilities of multiple classifiers are summed up and normalized afterward to calculate overall class probabilities. For each update, the confidence weighted class probabilities are added to the sum vector $\tilde{\boldsymbol{c}}_k^{(i)}$, which is used to calculate the result:

$$\tilde{\boldsymbol{c}}_k^{(i)} = \tilde{\boldsymbol{c}}_{k-1}^{(i)} + \boldsymbol{c}_{conf}^{(s_p)} \boldsymbol{c}_k^{(\boldsymbol{z}_j)}, \tag{5.23}$$

$$\boldsymbol{c}_k^{(i)} = \frac{\tilde{\boldsymbol{c}}_k^{(i)}}{\|\tilde{\boldsymbol{c}}_k^{(i)}\|_1}. \tag{5.24}$$

The advantage of the approach is the simple calculation scheme with a low number of parameters, while still doing a filtering over time. A disadvantage is that if the class is changed (e.g. triggered by a wrong association) the error can remain for a long time.

The Bayesian classification fusion that was proposed in chapter 4.5.1 for single sensor usage, can also be used for the fusion. One major drawback of this approach is, that the sensor's confidence value $c_{conf}^{(s_p)}$ is not used in its standard form. As a consequence, sensors with low and high confidence values can have the same impact on the estimation and thus lead to wrong results. To overcome this issue, this thesis proposes an implementation that generates a pseudo probabilities vector $\hat{\boldsymbol{c}}_k^{(\boldsymbol{z}_j)}$ for each measurement $\boldsymbol{z}_j$ using the sensor's confidence value:

$$\hat{\boldsymbol{c}}_k^{(\boldsymbol{z}_j)} = \boldsymbol{c}_k^{(\boldsymbol{z}_j)} c_{conf}^{(s_p)} + \mathbf{1}_n \frac{1 - c_{conf}^{(s_p)}}{|C|}. \tag{5.25}$$

Here, $\mathbf{1}_n$ is a vector consisting of n elements, set to 1. With this formulation, the measurement's class probabilities are weighted by the classification confidence $c_{conf}^{(s_p)}$ and added to an equal remaining probability weighted

by $1 - c_{conf}^{(s_p)}$. Therefore, the class probabilities converge in the case of small confidences to equal pseudo probabilities of $\frac{1}{|C|}$ with $|C|$ being the number of classes.

The four-class problem of this thesis uses the following transition model:

$$\boldsymbol{P}_t = \begin{bmatrix} 0.91 & 0.05 & 0.02 & 0.02 \\ 0.05 & 0.91 & 0.02 & 0.02 \\ 0.02 & 0.02 & 0.91 & 0.05 \\ 0.02 & 0.02 & 0.05 & 0.91 \end{bmatrix} \tag{5.26}$$

The advantage of the Bayesian fusion is, that the results are very close to true probabilities, calculated in a recursive way. In addition, the transition model can represent more complex interrelationships compared to the other approaches.

The Dempster-Shafer classification fusion uses the DST classification tracking scheme proposed in chapter 4.5.2 to combine the classification provided by multiple sensors with different confidences. The DST uses a power set 2^C, which results in $2^{|C|} = 16$ hypotheses for the four classes used in this thesis. As shown by Aeberhard (2017) and the example shown by Koks and Challa (2003), the power set can be reduced to "useful sets" in order to reduce the computational requirements. Here, the following power set of 8 hypotheses is used, that consists of the four basic hypotheses, two hypotheses for VRU and non-VRU objects, one for non-background objects and the "unknown" hypothesis including all classes:

$$p^C = \{C_{Bg}, C_{Car}, C_{Ped}, C_{Cyc}, C_{\overline{VRU}}, C_{VRU}, C_{\overline{Bg}}, C_{all}\}, \tag{5.27}$$
$$C_{\overline{VRU}} = \{C_{Bg}, C_{Car}\}, \tag{5.28}$$
$$C_{VRU} = \{C_{Ped}, C_{Cyc}\} \tag{5.29}$$
$$C_{\overline{Bg}} = \{C_{Car}, C_{Ped}, C_{Cyc}\}, \tag{5.30}$$
$$C_{all} = \{C_{Bg}, C_{Car}, C_{Ped}, C_{Cyc}\}. \tag{5.31}$$
$$\tag{5.32}$$

To perform the fusion, the masses $m_{k,\boldsymbol{z}_j}^{(s_p)}$ of all defined hypotheses need to be constructed based on the measurement's classification probabilities. (Aeberhard 2017) shows a method, which incorporates several assumptions and system knowledge for a proper fusion. This thesis, however, proposes a more general approach, where the masses purely defined by the sensor's classification probabilities and the classification confidence. This requires less system knowledge and allows universal application to different setups. A basic mass $b_{k,\boldsymbol{z}_j}^{(s_p)}(A)$ of hypothesis A is defined by a multiplication of sensor confidence $c_{conf}^{(s_p)}$ and class probability $p(C_a)$, if the hypothesis only includes class C_a. Otherwise, it is defined by a sum of $(1 - c_{conf}^{(s_p)})$ times the class probability for all hypotheses containing C_a. Then, the final mass $m_{k,\boldsymbol{z}_j}^{(s_p)}(A)$ is calculated by a normalization of the basic weight to ensure the sum of the weights is equal to 1:

$$b_{k,\boldsymbol{z}_j}^{(s_p)}(A) = \begin{cases} p(C_a) c_{conf}^{(s_p)}, & \text{if } A = \{C_a\} \\ \sum_{C_a \cap A = C_a} p(C_a)(1 - c_{conf}^{(s_p)}), & \text{if } A \neq \{C_a\} \end{cases}, \tag{5.33}$$

$$m_{k,\boldsymbol{z}_j}^{(s_p)}(A) = \frac{b_{k,\boldsymbol{z}_j}^{(s_p)}(A)}{\sum_{B \subseteq P^C} b_{k,\boldsymbol{z}_j}^{(s_p)}(B)}. \tag{5.34}$$

To propagate the masses through time, equation (4.57) is used, while equation (4.58) is used to generate pseudo probabilities as output to ADAS applications. The advantage of the Dempster-Shafer fusion is the explicit modeling of combinations and unknown hypotheses.

Overall, the Dempster-Shafer fusion is the most complex approach, while the voting scheme is the simplest option with the lowest amount of tuneable parameters. The Dempster-Shafer fusion has more degrees of freedom in terms of parameter tuning and has the highest computational requirements, but also a high potential due to the modeling of multiple and unknown hypotheses. The four presented approaches are compared in terms of accuracy, which is shown in chapter 6.

5.7 GM-PHD filter implementation

Within the proposed fusion framework, the GM-PHD filter needs to keep track of the classification and the unique ID of the tracks in addition to propagating the PHD over time. Therefore, the implementation of the GM-PHD filter follows the basic structure of the proposals from Panta et al. (2006) and Clark et al. (2006), who use tags to determine the unique object ID. However, some adaptations are made to optimize runtime, for better state estimation, and the possibility of classification propagation. The following paragraphs summarize the steps of the filter based on the work from Clark et al. (2006) and show the innovations and differences.

5.7.1 Tags and classification extensions

The approach used in this thesis adopts the method of using tags to keep preserving the track's unique IDs over time. As described by Clark et al. (2006), an unique identifier, or tag, $\tau_{k|k}$ at time step k is assigned to each Gaussian component forming a set

$$\tau_{k|k} = \left\{ \begin{array}{ccc} \tau_{k|k}^{(1)}, & \ldots, & \tau_{k|k}^{(J_{k|k})} \end{array} \right\}, \tag{5.35}$$

where $J_{k|k}$ is the number of Gaussian components at time step k. For a consecutive numbering of the tags, the variable τ_{max} is introduced, which contains the tag with the current highest number. In addition to the implementation from Clark et al. (2006), this thesis proposes an extension for the classification. A classification distribution $\boldsymbol{c}_{k|k}^{(i)}$ is assigned to each Gaussian component that makes up the set:

$$C_{k|k} = \left\{ \begin{array}{ccc} \boldsymbol{c}_{k|k}^{(1)}, & \ldots, & \boldsymbol{c}_{k|k}^{(J_{k|k})} \end{array} \right\} \tag{5.36}$$

with

$$\boldsymbol{c}_{k|k}^{(j)} = \left[\begin{array}{ccc} p(C_1) & \ldots & p(C_n) \end{array} \right]. \tag{5.37}$$

This classification set allows the tracking of the classes for each component over time.

5.7.2 Step 0: Initialization

In the implementation used here, the initial number of Gaussian components is $J_0 = 0$ and the tags and classification sets being empty with $\tau_0 = \emptyset$ and $C_0 = \emptyset$. The steps one to five are repeated after the initialization.

5.7.3 Step 1: Prediction

The prediction step consists of three parts and is shown in equation (4.35). Since the size of the prediction time-steps can be different, the process noise matrix $\boldsymbol{Q}_k$ and the dynamics matrix $\boldsymbol{A}_k$ are calculated as described in chapter 4.6.2. Different motion models are applicable to calculate $\boldsymbol{A}_k$, but this implementation uses the constant acceleration model shown in chapter 5.3. In addition, the survival probability $p_{S,k}$ is calculated for each time-step using a base survival probability $p_{S,base}$ and the time difference from the last time step Δt as follows:

$$p_{S,k} = p_{S,base}^{\Delta t}. \tag{5.38}$$

The proposed implementation does not model the spawn process of equation (4.35), which ignores some scenarios for simplicity and sets $v_{\beta,k|k-1}(\boldsymbol{x}) = 0$. For the birth model γ_k in equation (4.35), a new adaptive approach is presented here. Since new objects can appear anywhere in the monitored space, but the computational effort should be kept low and thus no high number of random Gaussian components should be created, adaptive birth models depending on the measurements are considered. Houssineau and Laneuville (2010) and Ristic (2013) propose adaptive birth processes, which create new Gaussian components based on all the measurements. However, this inevitably creates an overlap of birth components with already tracked components in many areas. These overlaps do not truly represent the birth of new targets. When trying to create an adaptive birth strategy, the best case would be to only add Gaussian components for measurements in areas where no components are present at the current timestamp. This reduces the influence of the birth targets on the update step while reducing the number of Gaussians and thus reducing the computational requirements.

Lindenmaier et al. (2022) show an approach using the sum of the updated component weights of the measurements in the previous time-step. This creates high weight birth components for measurements that have created a low sum of update weights in the previous time-step and thus can be assumed to be non-tracked targets. The proposed approach of this thesis follows a similar idea in using the update weight sum, but instead adds components with constant birth weights rather than modeling them depending on the weight sum. By using constant birth weights, an increased robustness against ghost objects is desired, since no high weight components can be directly created by the model.

In the presented approach, a birth measurement set $Z_{\gamma,k-1}$ is generated in the previous update step $k-1$, which includes all measurements with a sum of generated weights $w_{sum,k-1}^{(j)}$ below a threshold t_{w_γ}:

$$Z_{\gamma,k-1} = \left\{ \boldsymbol{z}_{\gamma,k-1}^{(j)} \in Z_{k-1} \,\forall\, w_{sum,k-1}^{(j)} < t_{w_\gamma} \right\}. \tag{5.39}$$

The threshold is found experimentally and set to $t_{w_\gamma} = 0.01$. For the sum of weights, the measurement weights $q_{k-1}^{(l)}(\boldsymbol{z})$ from equation (4.42) are used combined with the predicted weights $w_{k-1|k-2}$ from the previous time step:

$$w_{sum,k-1}^{(j)} = \sum_{l=1}^{J_{k-1|k-2}} w_{k-1|k-2}^{(l)} q_{k-1}^{(l)}(\boldsymbol{z}). \tag{5.40}$$

The sum of weights $w_{sum,k-1}^{(j)}$ indicates the influence of the j-th measurement on the PHD update. If the influence is very low, it is assumed that no Gaussian component of the intensity function is in close range to the measurement. Therefore, $Z_{\gamma,k-1}$ is a subset of the measurement set Z_{k-1}, that only contains measurements in areas where no Gaussian components are present in the intensity function. Next, a gaussian mixture $\hat{\gamma}_{k-1}$ is generated based on these measurements with the mean $\hat{\boldsymbol{m}}_{\gamma,k-1}^{(j)} = \begin{bmatrix} p_x^{(j)} & p_y^{(j)} & 0 & 0 & 0 & 0 & l^{(j)} & w^{(j)} & h^{(j)} & \phi^{(j)} \end{bmatrix}^T$

based on the measurement $\boldsymbol{z}_{\gamma,k-1}^{(j)} = \begin{bmatrix} p_x & p_y & l & w & h & \phi \end{bmatrix}^T$. The covariance $\hat{\boldsymbol{P}}_{\gamma,k-1}^{(j)} = \boldsymbol{P}_0$ is defined by the initial covariance $\boldsymbol{P}_0$ and $\hat{w}_{\gamma,k-1}^{(j)} = w_0$ by the initial weight w_0. Overall, the mixture is defined by

$$\hat{\gamma}_{k-1} = \sum_{i=1}^{\hat{J}_{\gamma,k-1}} \hat{w}_{\gamma,k-1}^{(i)} \mathcal{N}(\boldsymbol{x}; \hat{\boldsymbol{m}}_{\gamma,k-1}^{(i)}, \hat{\boldsymbol{P}}_{\gamma,k-1}^{(i)}), \tag{5.41}$$

which is then predicted to time step k using the GM-PHD prediction to form γ_k.

For each of the birth Gaussian components, a new birth tag is added and the set of birth tags at the current time step, as proposed by Clark et al. (2006). Each tag is initialized with a new unique ID based on τ_{max}, which is increased by J_{γ_k} afterward:

$$\tau_{k|k-1} = \tau_{k-1} \cup \left\{ \tau_{\gamma_k}^{(1)}, \quad \ldots, \quad \tau_{\gamma_k}^{(J_{\gamma_k})} \right\}, \tag{5.42}$$

$$\tau_{\gamma_k}^{(j)} = \tau_{max} + j. \tag{5.43}$$

$$\tag{5.44}$$

Similarly, a classification distribution is added for each Gaussian component. The initial classification is calculated by updating an uniform classification distribution $\boldsymbol{c}_U$ with the measurement classification probability vector $\boldsymbol{c}_{k-1}^{(\boldsymbol{z}_{\gamma,k-1}^{(j)})}$ using the classification fusion from chapter 5.6. This fusion is described as propagation function $f_c(\boldsymbol{c}, \boldsymbol{c}^{(z)})$ here, which represents either of the four presented classification fusion approaches:

$$C_{k|k-1} = C_{k-1} \cup \left\{ \boldsymbol{c}_{k|k-1}^{(1)}, \quad \ldots, \quad \boldsymbol{c}_{k|k-1}^{(J_{\gamma_k})} \right\}, \tag{5.45}$$

$$\boldsymbol{c}_{k|k-1}^{(j)} = f_c\left(\boldsymbol{c}_U, \boldsymbol{c}_{k-1}^{(\boldsymbol{z}_{\gamma,k-1}^{(j)})}\right). \tag{5.46}$$

5.7.4 Step 2: Update

The update step overall follows the procedure from Clark et al. (2006), but includes some adaptations for improved tracking results for ADAS applications.

This thesis proposes a gating process similar to the application for KF-based tracking shown in chapter 4.3.2, to only update nearby Gaussian components and therefore reducing the computational complexity. In typical GM-PHD implementations, many Gaussian components are added during the update step due to the update of each prior Gaussian component with each measurement Gaussian component, where many of these components have very low weights and would be pruned in the next step anyway. To reduce this number of low weighted components, a gating process based on the minimum of the squared Mahalanobis distance $D_{M,pos}$ and the Euclidean distance $D_{Euclid,pos}$ is proposed. Since gating in ADAS applications can be interpreted spatially, the gating is based solely on the position in X and Y coordinates. The minimum of the squared Mahalanobis distance $D_{M,pos}$ and the Euclidean distance $D_{Euclid,pos}$ defines the gate, which are calculated using the position difference $\boldsymbol{z}_{p,zm}^{(i)}$ and the position covariance $\boldsymbol{P}_p^{(i)}$, that only include the elements related to the X and Y position:

$$\boldsymbol{z}_{p,zm}^{(i)} = \begin{bmatrix} (z_1 - m_{k|k-1,1}^{(i)}) & (z_2 - m_{k|k-1,2}^{(i)}) \end{bmatrix}^T, \tag{5.47}$$

$$\boldsymbol{P}_p^{(i)} = \begin{bmatrix} P_{k|k-1,11}^{(i)} & P_{k|k-1,12}^{(i)} \\ P_{k|k-1,21}^{(i)} & P_{k|k-1,22}^{(i)} \end{bmatrix}. \tag{5.48}$$

The distances are then calculated with

$$D_{M,pos}(z_{p,zm}^{(i)}, P_p^{(i)}) = \left[\boldsymbol{z}_{p,zm}^{(i)}\right]^T \left[\boldsymbol{P}_p^{(i)}\right]^{-1} \left[\boldsymbol{z}_{p,zm}^{(i)}\right], \tag{5.49}$$

$$D_{Euclid,pos}(z_{p,zm}^{(i)}) = \sqrt{\left[\boldsymbol{z}_{p,zm}^{(i)}\right] \left[\boldsymbol{z}_{p,zm}^{(i)}\right]^T}. \tag{5.50}$$

Applying the gating based on those distances, the equation (4.40) changes to

$$v_{D,k}(\boldsymbol{x};\boldsymbol{z}) = \sum_{j \in V_k(\boldsymbol{z},\gamma_{DA})} w_k^{(j)}(\boldsymbol{z}) \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{m}_{k|k}^{(j)}(z), \boldsymbol{P}_{k|k}^{(j)}\right), \tag{5.51}$$

where only components of the gating indices $V_k(\boldsymbol{z}, \gamma_{DA})$ are taken into account. For these gating indices, either the squared Mahalanobis distance or the Euclidean distance is below the gating threshold γ_{DA}:

$$V_k(\boldsymbol{z}, \gamma_{DA}) = \left\{ i = 1, \ldots, J_k \mid \min(D_{M,p}(\boldsymbol{z}_{p,zm}^{(i)}, \boldsymbol{P}_p^{(i)})^2, D_{Euclid,p}(\boldsymbol{z}_{p,zm}^{(i)})) \leq \gamma_{DA} \right\}. \tag{5.52}$$

In the proposed implementation, the pseudo measurement $\hat{\boldsymbol{z}}$ is used for the update in equation (5.51), which is created by the procedure described in chapter 5.4 to be able to use measurements with geometries that do not match OBB.

In addition, the detection probability and clutter density are calculated by the developed models in chapter 5.5.1 and 5.5.2. Therefore, the following applies:

$$p_{D,k} = p_{D,k}^{(s_p)}(\boldsymbol{m}_{k|k-1}^{(j)}, \boldsymbol{c}_{k|k-1}^{(j)}), \tag{5.53}$$

$$\kappa_k(\boldsymbol{z}) = \kappa_k^{(s_p)}(\boldsymbol{z}, \boldsymbol{c}^{(\boldsymbol{z})}). \tag{5.54}$$

This leads to a replacement of equation (4.39) by

$$v_k(\boldsymbol{x}) = \sum_{j=1}^{J_{k|k-1}} \left[(1 - p_{D,k}^{(s_p)}(\boldsymbol{m}_{k|k-1}^{(j)}, \boldsymbol{c}_{k|k-1}^{(j)})) w_{k|k-1}^{(j)} \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{m}_{k|k-1}^{(j)}, \boldsymbol{P}_{k|k-1}^{(j)}\right)\right] + \sum_{\boldsymbol{z} \in Z_k} v_{D,k}(\boldsymbol{x};\boldsymbol{z}) \tag{5.55}$$

and a replacement of equation (4.41) by

$$w_k^{(j)}(\boldsymbol{z}) = \frac{p_{D,k}^{(s_p)}(\boldsymbol{m}_{k|k-1}^{(j)}, \boldsymbol{c}_{k|k-1}^{(j)}) w_{k|k-1}^{(j)} q_k^{(j)}(\boldsymbol{z})}{\kappa_k^{(s_p)}(\boldsymbol{z}, \boldsymbol{c}^{(z)}) + p_{D,k}^{(s_p)}(\boldsymbol{m}_{k|k-1}^{(j)}, \boldsymbol{c}_{k|k-1}^{(j)}) \sum_{l=1}^{J_{k|k-1}} w_{k|k-1}^{(l)} q_k^{(l)}(\boldsymbol{z})}. \tag{5.56}$$

The effects of these changes are discussed in the respective chapters, while experiments are provided in chapter 6 to prove the functionality. All other calculation steps remain the same as described in chapter 4.4.3. The tags and classifications are added for each new Gaussian component as follows:

$$\tau_{k,u} = \left\{ \tau_{k|k-1} \cup \tau_{k|k-1}^{z_1} \cup \cdots \cup \tau_{k|k-1}^{z_{|Z_k|}} \right\}, \tag{5.57}$$

$$\tau_{k|k-1}^{z} = \left\{ \tau_j \in \tau_{k|k-1} \mid j \in V_k(\boldsymbol{z}, \gamma_{DA}) \right\}, \tag{5.58}$$

$$C_{k,u} = \left\{ C_{k|k-1} \cup C_{k,u}^{z_1} \cup \cdots \cup C_{k,u}^{z_{|Z_k|}} \right\}, \tag{5.59}$$

$$C_{k,u}^{z} = \left\{ f_{\boldsymbol{c}}\left(\boldsymbol{c}_k^{(j)}, \boldsymbol{c}^{(z_{\gamma,k-1}{}^{(j)})}\right) \mid j \in V_k(\boldsymbol{z}, \gamma_{DA}) \right\}. \tag{5.60}$$

For each Gaussian component created by a measurement, a tag is added that has the same value as the tag of the updated component. Similarly, for each gaussian component created by a measurement, the classification vector is updated by the measurement's classification vector $\boldsymbol{c}^{(z_{\gamma,k-1}{}^{(j)})}$ using the classification update function $f_c(\boldsymbol{c}, c^{(z)})$, that implements either of the four proposed classification fusion approaches.

5.7.5 Step 3: Pruning

The pruning step from Clark et al. (2006) is implemented without adaptations, which means that all components with low weights below a truncation threshold τ are eliminated. The classification set $C_{k,u}$ is pruned in a similar way to the other properties.

5.7.6 Step 4: Merging

The merging procedure in general follows the one proposed by Clark et al. (2006) and has the goal to merge Gaussian components that have a low distance based on a distance criterion. Some implementations (Clark et al. 2006; Vo and Ma 2006) use the following distance criterion to calculate the distance $D(\mathcal{N}_i||\mathcal{N}_j)$ between components i and j:

$$D(\mathcal{N}_i||\mathcal{N}_j) = (\boldsymbol{m}_k^{(i)} - \boldsymbol{m}_k^{(j)})^T (\boldsymbol{P}_k^{(i)})^{-1} (\boldsymbol{m}_k^{(i)} - \boldsymbol{m}_k^{(j)}). \tag{5.61}$$

However, similar to Granström and Orguner (2012), in this implementation the distance is formed using the Kullback-Leiber Divergence (KLD) D_{KL}, which is defined as follows:

$$D_{KL}(\mathcal{N}_i||\mathcal{N}_j) = \frac{1}{2}\left(\mathrm{tr}(\boldsymbol{P}_j^{-1}\boldsymbol{P}_i) - k + (\boldsymbol{m}_j - \boldsymbol{m}_i)^T \boldsymbol{P}_j^{-}1(\boldsymbol{m}_j - \boldsymbol{m}_i) + \ln\left(\frac{\det \boldsymbol{P}_j}{\det \boldsymbol{P}_i}\right)\right). \tag{5.62}$$

Here, k is the dimension of the covariance matrices $\boldsymbol{P}_i$ and $\boldsymbol{P}_j$. The KLD can be interpreted as a measure which indicates the information loss when $\mathcal{N}_i$ is approximated by $\mathcal{N}_j$. Therefore, it is well suited as a criterion whether two Gaussian components should be merged, since it indicates the possible information loss. Note, that the KLD is asymmetric and should thus not be used reversely.

When merging multiple components, the tag $\tau_k^{(i)}$ of the component with the highest weight $w_k^{(i)}$ is kept for the merged component. If components with the same tag still exist after the merge procedure, the component with the highest weight keeps the tag and all others get a new one assigned, similar to the proposal from Clark et al. (2006).

The classification is merged similarly to the mean. Therefore, the merged classification vector $\tilde{\boldsymbol{c}}_k^{(l)}$ is calculated using the weighted average of all merge components L.

$$\tilde{\boldsymbol{c}}_k^{(l)} = \frac{1}{\tilde{w}_k^{(l)}} \sum_{i \in L} w_k^{(i)} \boldsymbol{c}_k^{(i)}. \tag{5.63}$$

5.7.7 Step 5: Track extraction

In the last step, the tracks are extracted from the PHD estimation and handed over to the track confirmation strategy proposed in chapter 5.9. The tracks consist of a state vector, a classification vector and an ID extracted from the Gaussian mixture, the tag set and the classification set. To be interpreted as a track, the weight of a Gaussian component has to exceed a weight threshold $w_{\min}$ creating the track set T_k:

$$T_k = \left\{(\tau_k^{(i)}, \boldsymbol{m}_k^{(i)}, \boldsymbol{c}_k^{(i)}, \tilde{p}_k^{(i)}(\exists)) \,|\, w_k^{(i)} > w_{\min}\right\} \tag{5.64}$$

To provide an existence probability for further processing steps, the weight of the Gaussian component can be interpreted as pseudo existence probability with $\tilde{p}_k^{(i)}(\exists) = \min(w_k^{(i)}, 1)$.

5.8 Kalman filter implementation

The Kalman filter implementation is interchangeable with the GM-PHD filter implementation, since it uses the same interfaces. To provide a meaningful comparison, the Kalman filter uses the same motion model like the GM-PHD implementation. Therefore, the main differences are in the implementation of the prediction and update routine, as well as the management of the track list. Each element in the track list consists of the track's state, its existence probability and its classification vector.

Similar to the GM-PHD implementation, a CA model is used for the state prediction. The prediction and update is implemented as described in chapter 4.3.2 In addition, the existence probability is estimated, as explained in chapter 4.3.2 using a constant survival probability and a constant clutter probability, which should not be mixed up with the clutter density.

In the update step, a gating based on Mahalanobis distance and data association using the GNNDA approach are used, as described in chapter 4.3.2.1. The Mahalanobis distance is used as metric for the cost matrix of the DA, which is then solved by the Hungarian algorithm (Kuhn 1955; Munkres 1957). The state of each track that has an assigned measurement is then updated using the respective measurement model. In addition, the existence probability is updated according to equation (4.16) and the classification vector is also updated using one of the methods proposed in chapter 5.6. The existence probability of each track without an associated measurement is in turn updated as non-detected according to equations (4.17). The detection probability used for the existence estimation is provided by the parameter model from chapter 5.5.1.1.

All detections, which have not been associated to a track, are added as new elements to the track list.

In order to handle overlapping bounding boxes and to delete obsolete tracks, the "prune and merge" procedure from the GM-PHD filter implementation is adopted after each update cycle. The tracks with existence probabilities below a threshold are removed. Then, the tracks with low KLD are merged to deal with overlapping tracks originated from the same detections. Since this process requires weights, the existence probability is used as the weight. When merging the existence probability of a merge set M, it is done as a weighted average, effectively resulting in:

$$\begin{aligned} w_j &= p_j(\exists) \\ p_i(\exists) &= \frac{\sum_{j \in M} w_j p_j(\exists)}{\sum_{j \in M} w_j} \end{aligned} \tag{5.65}$$

The combination of the states and the classification is done in the same way as in the GM-PHD filter implementation, using the weights w_j. Now, the tracks are handed over to the track confirmation module for further processing.

5.9 Track confirmation strategy

The MOT filter (GM-PHD or KF), which is the core of the fusion architecture, generates a list of tracks for each time step that contains the currently tracked objects. Each of the objects has a state, a class, a unique ID, and an existence probability. The existence probability is decisive in determining whether an object exists or not, and thus needs to be considered by ADAS functions. However, there are situations in which the modeling of existence using a probability reaches its limits. The following situations are examples where reliance on existence probability is not sufficient:

- If a previously tracked object is occluded over several time-steps and therefore not visible for multiple sensors, the existence probability will drop fast. Such a drop is also possible due to the spooky effekt (Vo and Vo 2012). The object is, however, still existent at the occluded location. If the occlusion is resolved,

the object will be tracked again, but it will take time to raise the existence probability and the unique ID is changed.

- If one of multiple sensors continuously fails to detect an object due to some systematic error, the existence probability is reduced and can potentially be in the range of the threshold for forwarding the track.
- If multiple sensors create a clutter object at a certain location randomly or due to a systematic error, a track with an existence probability above the threshold could be created. The existence probability may decrease after a short time, but this clutter track can potentially cause a false positive action by the ADAS function.

A possible solution for occlusion is the explicit modeling, as shown by Aeberhard (2017). Another solution is the integration of an occlusion model to the GM-PHD detection probability model (Vasic and Martinoli 2015; Granström et al. 2014; Törő et al. 2021). Since the framework of this thesis should work with different MOT filters, the occlusion problem is handled in the track confirmation strategy, which is applied independent of the used filter.

The strategy confirms tracks based on several criteria over time and only passes the confirmed ones to the ADAS function. It can handle ID changes, remove false positive clutter tracks and tracks with lower than expected existence probabilities. It consists of a confirmation list that is updated after each MOT update cycle by the MOT update track set T_k. Each element in the confirmation list consists of a confirmation track $\hat{T}_k^{(i)}$, an ID alias, the unobserved time, the first appeared time t_{fa} and a confirmation flag, as shown in the example in Figure 5.15. The unobserved time is the time, since the ID of the track was last updated by the MOT tracker.

Figure 5.15 shows an example scenario with two cars around the ego vehicle. Arrows show the trajectories of the tracks provided by the MOT filter. One car is directly in front of the ego vehicle, but the provided existence probability is low. The other car is overtaking and occluded for a short duration, where the tracks are lost. In addition, in the right top corner, there is a false positive track. This scenario therefore represents different errors that possibly can occur for a MOT system.

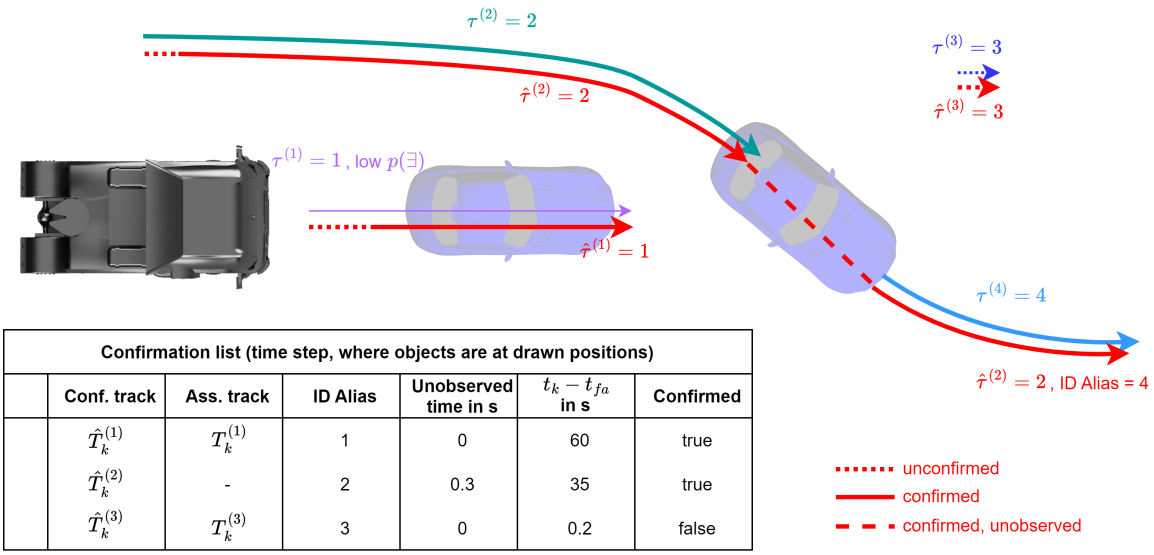
In each update cycle, the tracks of the confirmation list are tried to be updated. Here, the alias ID of the stored track is compared with the ID of the tracks in the MOT list. If no match is found, the unobserved time is increased and the state is predicted using the motion model. In Figure 5.15, this is the case during occlusion of the overtaking car with $\hat{\tau}^{(2)} = 2$. Otherwise, the unobserved time is set to 0.

Next, ID switches are searched for all tracks in the confirmation list, that do not yet have an assignment from the MOT list. If an unused track from the update track set falls within an Euclidean distance threshold from a non-updated track in the confirmation track list, an ID change is assumed. In this case, the ID alias is set to the new ID, which is indicated in Figure 5.15 after the occluded car is tracked again. In this way, objects which have been lost for a short time due to e.g. occlusion can be found again while preserving their original unique ID. For all tracks in the update track set that are not yet assigned to a track in the confirmation list, a new element is created in the confirmation list.

The next step checks for all elements in the list whether they can be marked as confirmed. To be confirmed, one of the following criteria must be met:

- $p(\exists) > p_{\exists,min} \wedge t_k - t_{fa} > t_{min}$
- $t_k - t_{fa} > t_{conf}$

To be confirmed, a track must either have an existence probability $p(\exists)$ greater than a threshold $p_{\exists,min}$ and at the same time be active for a minimum duration t_{min} by comparing with the time of the current time step t_k, or be active for a greater duration t_{conf} regardless of the existence probability. In Figure 5.15, the car with low existence probability and $\hat{\tau}^{(1)} = 1$ therefore is confirmed since it is tracked for a long duration, while the false positive track with $\hat{\tau}^{(3)} = 3$ is not confirmed and therefore not forwarded to ADAS functions. All tracks are then checked for deletion using the unobserved time. If the unobserved time is above a threshold, they are



Confirmation list (time step, where objects are at drawn positions)

	Conf. track	Ass. track	ID Alias	Unobserved time in s	$t_k - t_{fa}$ in s	Confirmed
	$\hat{T}_k^{(1)}$	$T_k^{(1)}$	1	0	60	true
	$\hat{T}_k^{(2)}$	-	2	0.3	35	true
	$\hat{T}_k^{(3)}$	$T_k^{(3)}$	3	0	0.2	false

Figure 5.15: Example situation with track confirmation strategy. The trajectories of the confirmation list are drawn in red, while the trajectories of the track set have different colors. The confirmation list table shows example values at the time where the objects are at the cars positions.

removed from the list. The threshold for already confirmed tracks is higher than the threshold for unconfirmed tracks.

Overall, this confirmation strategy can filter out some of the errors that are created by MOT systems, as shown in Figure 5.15. In particular, situations where ID changes occur or tracks are lost for a short period of time can be resolved. In addition, it can reduce the amount of false positive tracks transmitted to an ADAS application. Therefore, the confirmation strategy can help to improve the overall system performance.

6 Evaluation of multi-object detection and tracking

This chapter summarizes the experiments and results of the developed approaches for object detection and multi-sensor multi-object tracking. The experiments are conducted with the KITTI dataset and a newly created truck dataset, which are presented in chapter 6.1. The metrics for object detection experiments used are then presented in chapter 6.2, followed by the results of the two-stage Lidar detection algorithm (chapter 6.3) and the results of the camera-based downward looking object detection approach (chapter 6.4). Chapter 6.5 then shows the metrics used for MOT evaluation, followed by the experiments for MOT, which, in addition to a simple filter comparison in chapter 6.6, shows several experiments to evaluate the proposed fusion framework in chapter 6.7. The respective experiments are explained in detail in the corresponding chapters.

6.1 Datasets

Two datasets are used to evaluate the methods and approaches developed in this thesis. The popular KITTI dataset proposed by Geiger et al. (2012) is used since it offers a large amount of ground truth data and enables comparability to numerous works in the literature. For validation on trucks and the integration of the current generation of sensors, an additional truck dataset is created. The main task of this dataset is the validation of algorithms developed with the KITTI dataset for truck-specific applications and using current generation series sensors. In addition, it allows the integration of additional perception sensors for tracking and fusion. An overview and comparison between both the KITTI and the truck dataset is given in Table 6.1, where the interface of the sensors is either specified as "raw" or "object". Raw data is defined as a sensor's output without any object detection algorithm running at sensor level. For cameras, raw data are images, while for Lidar sensors, raw data are point clouds. Object data, on the other hand, is defined as a list of 3D objects created by a detection algorithm. Both data sets also have data from a GNSS-based localization system. A more detailed description of the sensors can be found in the following chapters on the datasets.

The KITTI dataset consists of more scenes and therefore a larger amount of data than the truck dataset. The sensors of the KITTI dataset are hardware synchronized which results in a simultaneous recording of the scene, while the truck dataset uses sensors, that have asynchronous measurements with known time stamps. The sensor setup and the sensor's FoVs are visualized in Figure 6.1. As shown by the Figure, both datasets have a forward facing camera and Lidar, where the KITTI dataset has a larger FoV. The truck setup relies more on radars, with two corner SRRs and a central LRR.

For both datasets, the maximum evaluation distance is set to 70 m. Above that distance, both ground truths and tracks are ignored. On the KITTI dataset, the sensor performance massively degrades above this distance, while an accurate labeling was not possible for higher distances on the truck dataset due to the limited Lidar point cloud density.

Table 6.1: Comparison of KITTI MOT dataset and truck dataset.

		KITTI MOT dataset	Truck dataset
Number of scenes		21	3
Measurement timing		Simultaneous firing	Asynchronous
Sensors	Lidar	1 (raw)	1 (raw)
	Camera	4 (raw, only 1 used)	1 (object)
	Radar	-	3 (object)
	Localization	1	1

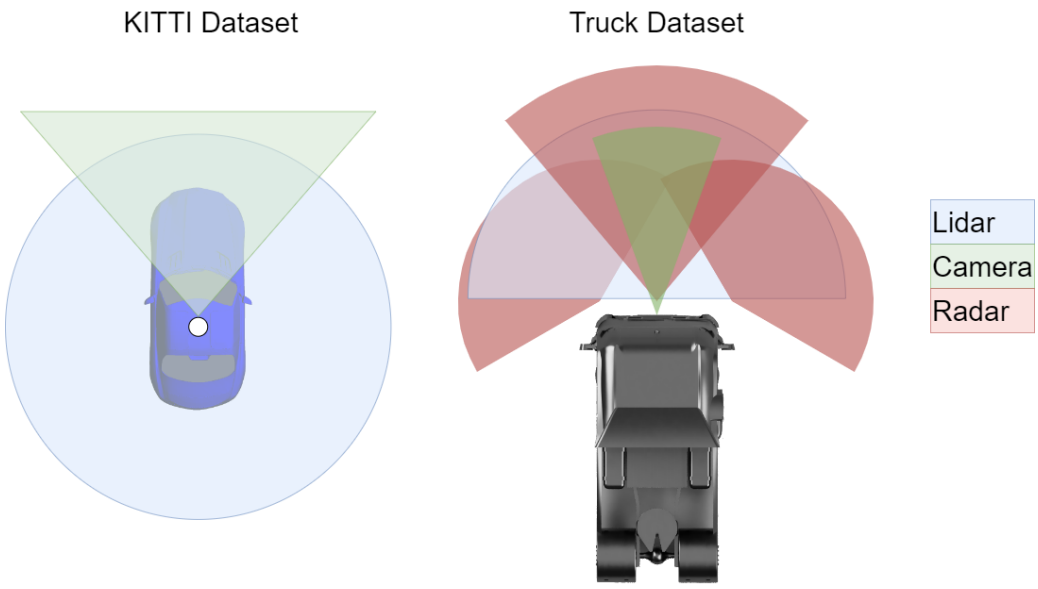


Figure 6.1: Comparison of the sensor setups and FoV of the KITTI and the truck dataset.

6.1.1 KITTI dataset

The KITTI vision benchmark suite (Geiger et al. 2012) is a large-scale real-world dataset recorded in the area of Karlsruhe and covering mainly urban scenarios. The dataset provides recordings from a timely and spatially aligned sensor kit, which consists of a Velodyne HDL-64E 360° Lidar, four video cameras and a high-precision localization system, alongside all necessary calibration data. The four cameras are mounted as two stereo pairs, one as RGB and the other as grayscale. The used localization system is a "state-of-the-art OXTS RT 3003 localization system which combines GPS, GLONASS, an IMU and RTK correction signals" (Geiger et al. 2012, p. 3354). The dataset includes benchmarks with ground truth data for various tasks, including object detection and object tracking. In this thesis, both the MOT benchmark and the 3D object detection benchmark are used. A clear drawback of the dataset is the uniform conditions of recording. As stated by Arnold et al. (2019), p. 3793, the dataset is recorded in daylight scenes and standard weather conditions. Therefore, the evaluations on this dataset do not represent general performance indicators.

Since the proposed fusion framework works with object interfaces and the KITTI dataset only provides raw data, object detection approaches are applied to the raw data for further processing, as summarized by Table 6.2. Both detection approaches used create OBBs and are able to estimate the dimensions of the objects. In this thesis, only one of the four cameras is used, since monocular camera systems are common for trucks.

The ground truth data provided for the KITTI benchmarks includes "DontCare" regions, which are areas that are challenging to annotate or are not crucial for certain evaluation tasks. These areas and everything outside the camera's FoV are excluded from the MOT evaluation.

Table 6.2: Overview of sensor interfaces for the KITTI dataset.

Sensor	Detection approach	Geometric Representation	Dimension estimation capability
Lidar (Velodyne HDL-64E)	Point-RCNN (Shi et al. 2019)	OBB	yes
Camera (Point Grey FL2-14S3C-C)	Yolo-Mono-3D (Liu et al. 2021)	OBB	yes

6.1.2 Truck dataset

In order to evaluate the tracking and fusion algorithms based on truck sensors, a new dataset is created. In particular, this dataset includes sensors comparable to series equipment with integrated object detection to prove the validity and performance of the presented approaches for use in trucks.

The truck dataset contains recordings of a central LRR, two corner SRR sensors, a perception camera and a front Lidar, as shown in Figure 6.1. A Trimble BX928 GNSS system is used for localization. Table 6.3 shows an overview of all perception sensors and their interfaces. As shown, different geometric object representations exist throughout the sensor set, including OBB, point and L-shape representations that need to be combined. The SRR sensors and the Lidar are able to estimate the dimensions of an object, while the LRR and camera only provide points and widths.

The Lidar sensor is particularly necessary for the generation of ground truth labels, as its point clouds enable a 3D representation of the environment. The labeling tool "SUSTech POINTS" (Li et al. 2020) is used for the annotation process. The GT OBB labels are created by hand using the Lidar point cloud for spatial mapping and the camera image as visual aid, which is a similar procedure as others, like the KITTI dataset, use.

The Lidar achieves distance accuracies of ± 3 cm for ranges up to 200m on 10 % reflectivity targets. Since the labels are generated based on the point clouds with these errors and are created by hand, the ground truth labels of the truck dataset have to be considered as approximate truth. Similar to the KITTI dataset, "DontCare" regions are created for areas that are challenging or impossible to annotate or not relevant for evaluation. These areas are also marked by 3D OBBs and the areas are excluded from the evaluations. Visualizations of these areas can be found in the example images in Table 6.4.

The truck dataset is small compared to KITTI and contains 3 scenes from the different ODDs highway, rural and urban. It includes 2393 annotations over all time-steps with a majority being cars. The scene length between 15.4 s and 21.1s is comparable to many from the KITTI dataset, although KITTI also includes some with a higher duration. Table 6.4 shows an overview over the individual scenes and their properties.

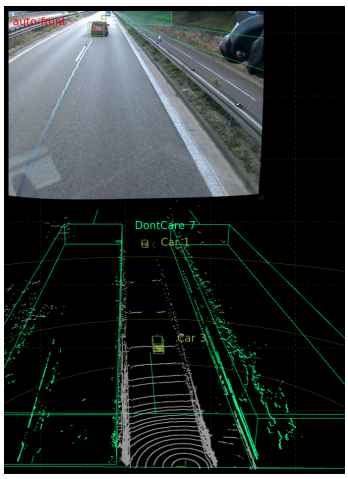
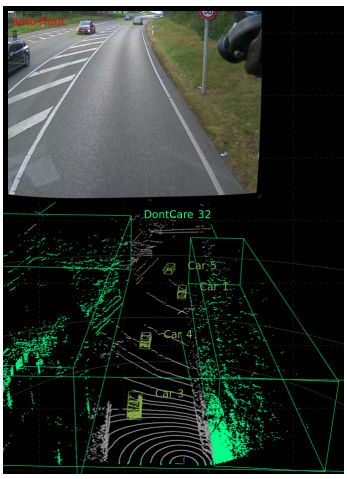
Due to the smaller size, the truck dataset is less meaningful, but it can still serve as confirmation or questioning of the results on the KITTI dataset. The truck dataset also provides the opportunity to study the fusion of more than two sensors with different FoVs. In addition, the sensors are mounted on a different mechanical platform with different mounting positions, as well as other vibrations and motion influences.

For this dataset, the detection sensors are assumed to have bigger uncertainties in detection, localization, and classification than the methods used on the KITTI dataset, so performance of the fusion framework in difficult conditions can be studied. By combining the two datasets, experiments can be conducted using both the well-studied KITTI dataset and validating these using a realistic truck dataset with current sensors close to series.

Table 6.3: Overview of sensor interfaces for the truck dataset.

Sensor	Detection approach	Geometric Representation	Dimension estimation capability
Lidar (Velodyne VLP-32c)	proposed approach chapter 3.2.3	OBB	yes
Camera	built in	Point	no
LRR	built in	Point & width	no
SRR (left & right)	built in	L-shape	yes

Table 6.4: Overview of individual scenes of the truck dataset and visualization using a separate context camera and the Lidar point cloud.

	 Highway scene	 Urban scene	 Rural scene
Description	Highway scene with multiple cars overtaking the ego vehicle.	Urban scene with preceeding and oncoming vehicles, multiple parking vehicles and traffic at intersections.	Rural scene with preceeding and oncoming vehicles and an intersection.
Scene length	15.4 s	21.1 s	18.0 s
Car tracks	5	29	25
Pedestrian tracks	0	1	2
Overall annotations	320	1258	815

6.2 Metrics for object detection

This chapter explains the basic metrics for evaluating object detection methods, which generate an OBB and a confidence level for each detection. GT OBBs are also required for the evaluation, with which the detections can be compared. The similarity between a GT shape B_{GT} and a detection shape B_d is measured using the Intersection over Union (IoU), also referred to as Jaccard index. This metric is calculated by dividing the intersecting area or volume by the union area or volume of the two shapes. Figure 6.2 visually shows the IoU of 2D bounding boxes, while the IoU of 3D OBBs is defined by

$$IoU_{3D} = \frac{intersecting\ volume}{united\ volume} = \frac{B_d \cap B_{GT}}{B_d \cup B_{GT}}. \tag{6.1}$$

A detected object is defined as True Positive (TP), if its IoU with a GT object exceeds a threshold. For the KITTI benchmark, the standard thresholds defined by the authors are 0.7 for cars and to 0.5 for pedestrians. Detections, that have no associated GT object above the threshold, are defined as False Positive (FP). On the other hand, GT objects without an associated detection OBB are defined as False Negative (FN).

The two basic metrics for the detection performance are precision and recall. The precision P is defined as the sum of TPs over the sum of all detections, which is the combination of TPs and FPs:

$$P = \frac{TP}{TP + FP}. \tag{6.2}$$

The recall R is defined as the sum of TPs over all GT objects, which is the combination of TPs and FNs:

$$R = \frac{TP}{TP + FN}. \tag{6.3}$$

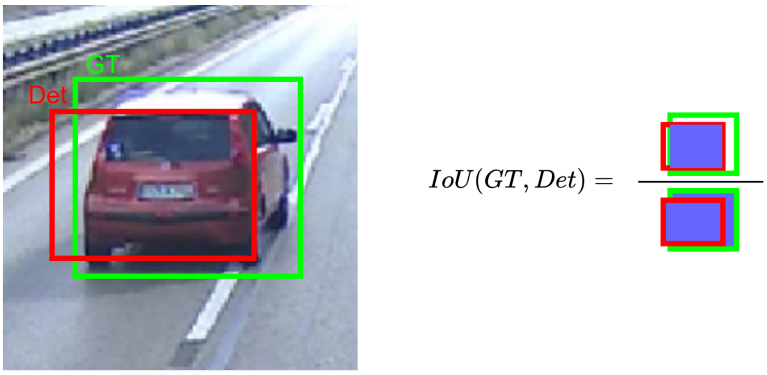


Figure 6.2: IoU for a 2D bounding box in image coordinates. Note, that these metric can also be applied to other types of shapes and volumes, like OBBs.

The precision is therefore a metric for the quality of the detected objects, while recall is a metric for the number of detections.

The confidence level of a detection can be used to filter out detections of low confidence, which is used to control the tradeoff between precision and recall. Low thresholds typically lead to higher recall and lower precision values, while high thresholds typically lead to lower recall and higher precision values. The precision-recall curve can visualize this trade-off between precision p and recall r depending on the chosen threshold for the confidence level, as shown in Figure 6.3. It is derived using multiple confidence thresholds for the detections, while the interpolated precision-recall curve ensures monotony and is defined as follows:

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r}>=r} p(\tilde{r}). \tag{6.4}$$

The Average Precision (AP) is the most popular metric for both 2D and 3D object detection and is described by Everingham et al. (2010) for image-based detections. It is the standard evaluation metric for various datasets like the KITTI detection benchmark. The AP is calculated by numerically integrating over a given interpolated precision-recall curve $p_{interp}(r)$:

$$AP = \frac{1}{|R|} \sum_{r \in R} p_{interp}(r). \tag{6.5}$$

Everingham et al. (2010) propose a calculation using $R_{11} = \{0, 0.1, 0.2, ..., 1\}$. However, Simonelli et al. (2020) recommend a new calculation method using $R_{40} = \{1/40, 2/40, 3/40, ..., 1\}$, which results in a more fair comparison by eliminating the first bin. This method is recently used by the KITTI benchmark and in this thesis. For multi class evaluation, the mAP can simply be evaluated by averaging over the AP of a set of classes C where AP_i is the AP of class C_i:

$$mAP = \frac{1}{|C|} \sum_{i=1}^{|C|} AP_i. \tag{6.6}$$

6.3 Evaluation of the two-stage Lidar detection algorithm

The Lidar object detection algorithm proposed in chapter 3.2.3 consists of two calculation steps, where the first detects objects using traditional geometric methods. This allows to generate separate point cloud segments for each physical object. The second step then estimates the class and the 3D OBB dimensions based on that segment using a NN.

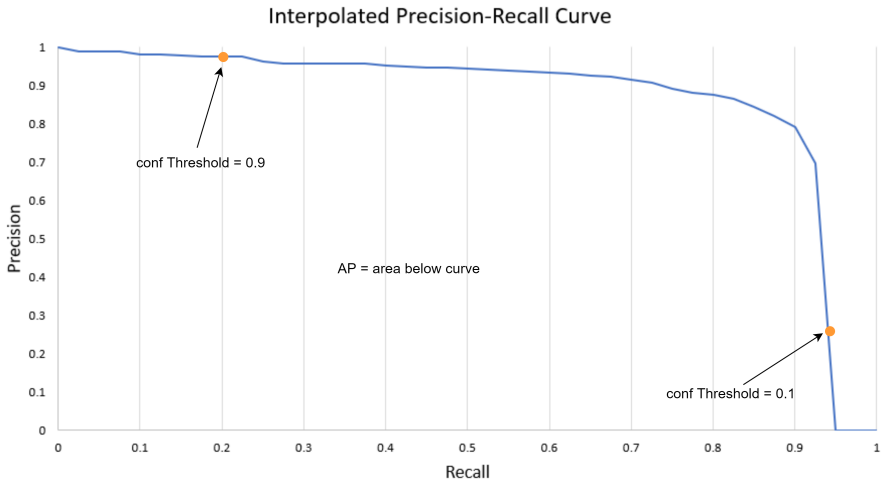


Figure 6.3: Exemplary interpolated precision-recall curve showing the tradeoff between both values. The area below the curve is the AP and is calculated by numerically integrating the curve.

The evaluation of the approach is done using the KITTI detection dataset. Since the GT of the test set is not available, this is done on a 20 % test split of the training data. Table 6.5 shows the results on the 2D, 3D and BEV benchmark. The approach achieves a mAP score of 79.33 % on the 3D detection benchmark (car, easy)[1], which is comparable to approaches like VoxelNet (Zhou and Tuzel 2018) with an mAP of 81.97%. A comparison to other approaches can be done using Table 3.3 from chapter 3.2.2. The achieved scores are not as high as the best currently available approaches, but this two-stage algorithm additionally detects background objects. This is important in situations, where instances of unknown classes appear in dangerous positions on the road, which are not trained.

For this reason, the first stage of the network is evaluated separately. The output of the first stage is a point cloud segment and an OBB based on traditional algorithms for each physical object in the point cloud. In order to evaluate these outputs, the unclassified recall is calculated. This value expresses the recall of the all detections without considering the class. Therefore, based on the first stage's output, a TP is counted in case of large overlap of a segment's point cloud with a GT OBB of any class. A detection is counted as TP, if more than 5 Lidar points of this detection lie inside a GT OBB, with less than 20 % of the Lidar points being outside. The experimentally derived minimum point amount serves as a threshold for ensuring an object physically exists, while the overlap threshold ensures to rule out segmentation errors. The detections are divided into Easy, Medium and Hard for each of the classes using the same constraints as the KITTI dataset[1]. The unclassified recall values are also shown in Table 6.5. The presented approach achieves unclassified recalls of up to 96.53 (car, easy)[1], which shows that an enormous amount of all objects is correctly detected. This also shows the suitability for safety critical applications, where the correct detection of a road user or obstacle is mandatory.

In addition to the evaluation of the full object detection pipeline, the classification from the second stage is evaluated separately. This allows the comparison to other deep learning approaches working on point cloud segments. Therefore, the evaluation metric is the accuracy of the classification without taking into account the detection of the first stage, which is defined as the percentage of correctly classified point clouds. The comparison is done on the STC (Teichman et al. 2011) and the ModelNet40 dataset (Wu et al. 62015) and shown in detail by Bader et al. (2021). Both datasets offer point cloud segments with GT class labels. The ModelNet40 segments are based on CAD data, while the STC is based on Lidar measurements. The STC is

[1]Easy: Min. bounding box height: 40 Px, Max. occlusion level: Fully visible, Max. truncation: 15 %
Moderate: Min. bounding box height: 25 Px, Max. occlusion level: Partly occluded, Max. truncation: 30%
Hard: Min. bounding box height: 25 Px, Max. occlusion level: Difficult to see, Max. truncation: 50%

Table 6.5: AP and unclassified recall results of the two-stage approach on the KITTI Dataset. Note that the unclassified recall values are calculated differently and should therefore not be compared with the AP results.

	Car			Pedestrian			Cyclist		
	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
AP 2D	90.67	76.32	71.78	39.30	31.55	29.36	57.49	45.92	43.65
AP BEV	87.98	73.19	68.52	43.75	35.06	30.94	54.78	43.78	41.54
AP 3D	79.33	60.58	55.80	43.24	34.72	30.66	52.51	41.54	39.25
unclassified R	96.53	89.26	87.21	58.961	53.15	50.37	83.16	78.56	75.46

organized in tracks and therefore provides additional information, like velocity, for each segment. However, the presented approach only uses the point cloud segments. For both datasets, the evaluation shows a great trade-off between inference time and accuracy, which is important for real-time applications. As shown by Table 6.6, the accuracy on the ModelNet40 dataset is comparable to (Shabat et al. 2018), which has a 7-times higher runtime on an NVIDIA RTX4000 GPU. Based on the analysis from Bader et al. (2021), it is assumed that for a real-world system, 32 object segments need to be classified within 50 ms as a minimum requirement, which results in a maximum runtime of 1.56 ms per point cloud segment. Within this constraint, the proposed approach clearly achieves the highest accuracy while still having the lowest runtime on subsampled point cloud sizes of 2048.

Table 6.6: Comparison of classification accuracy and inference time using the ModelNet40 (Bader et al. 2021).

Method	Accuracy [%]	Runtime (inference) [ms]
3DmFV (Shabat et al. 2018)	91.6	4.4
LightNet parenZhi et al. (2017)	86.9	N/A
VoxNet (Maturana and Scherer 2015)	83	N/A
VRN Ensemble (Brock et al. 2016)	95.54	460.4
RotationNet (Kanezaki et al. 2018)	**97.37**	N/A
PointNet (Charles et al. 2017a)	89.2	1.44
PointNet++ (Charles et al. 2017b)	90.7	2.02
PointGrid (Le and Duan 2018)	92.0	14.91 (original paper)
proposed approach (2048 Pts)	91.45	**1.07**

Since the STC consists of segmented real-world Lidar data, it is well suited to examine performance. As shown in Table 6.7, the proposed network also clearly outperforms heuristic and SVM-based approaches on this dataset, even though some of them use additional tracking information. To prove the robustness against occlusion, which is a common issue with Lidar point cloud segments, the classification accuracy is tested with simulated occlusion of the STC objects as described by Bader et al. (2021). As shown in Table 6.7, the approach is stable against occlusion, with the results on 40 % occluded point cloud segments still outperforming several other approaches and the accuracy declining by less than 3%.

Overall, the investigations show, that the proposed two-stage Lidar detection approach provides a state-of-the-art accuracy for object detection while still being able to run in real-time. The classification is stable against the common issue of occlusion and outperforming other approaches within runtime constraints for real-time application. The evaluation of the unclassified recall shows, that almost all objects of relevance are detected by the first stage, which proves the reliability for safety critical systems in contrast to other deep learning approaches.

Table 6.7: Comparison of classification accuracy and inference time using the STC (Bader et al. 2021).

Method	Acc [%]	Inf. time [ms]
Segment classifier only (Teichman et al. 2011)	93.1	N/A
ADBF classifier[1] (Teichman et al. 2011)	97.9	N/A
SVM[12] (Lin et al. 2018)	95.5	N/A
SVM[3]	94.66	0.34
heuristic approach[3]	88.39	**<0.01**
proposed approach (512)	**96.85**	0.43
proposed approach (20 % occlusion) (512 points)	95.51	0.79
proposed approach (40 % occlusion) (512 points)	94.27	0.79

[1]Using tracking information, like speed, velocity, ...
[2]Only using balanced subset of STC with only cars, pedestrians and cyclists
[3]Inhouse developments of Daimler Trucks for comparison

6.4 Evaluation of the camera detection algorithm for downward looking cameras at trucks

The camera detection algorithm proposed in chapter 3.3.3 uses a 2D object detector, a CNN on the cropped detections for estimation of the dimensions and orientation and a projection to the road surface to calculate the position of the OBB.

For the evaluation of the approach, a dataset with 765 images of a downward looking camera, that include objects is recorded and annotated as described by Zhang (2021). The proposed approach is applied to the images to generate 3D OBBs. Now, the distance in X-direction of the closest edge of the detections are compared to the ground truth OBBs. Figure 6.4 shows the distance error of the real-world detections over the closest distance of the ground truth objects. More than 83 % of all measurements are within an absolute error of < 0.3 m, with a standard deviation of 0.257 m and an average error of 0.185 m. Since 0.3 m is about half the width of a typical pedestrian OBB, it is possible to detect a 50 % overlap with the driving path with a high chance. These results show that for this camera position, a multi-stage approach can achieve distance estimations with errors at low decimeter level.

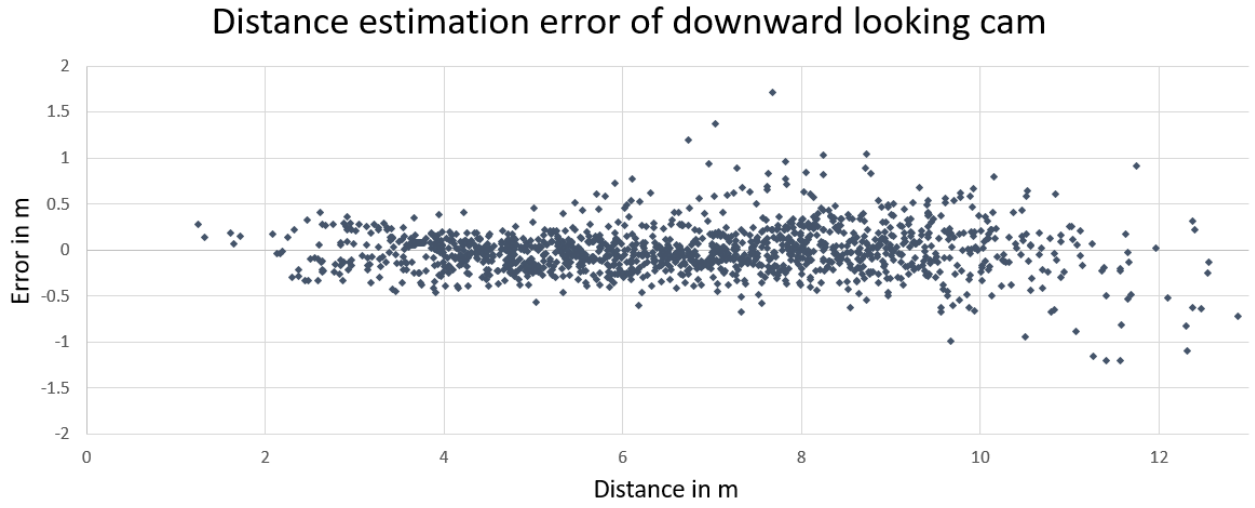


Figure 6.4: Distance error of camera detections to GT from Lidar over the object distance.

Using the created dataset, the 3D detection performance is evaluated for the class "Car". The other classes are ignored due to the low number of examples. An mAP of 95.4 % on 2D image coordinates and a mAP of 45.28 % on 3D OBBs is achieved. Other monocular camera approaches achieve lower scores between 4.47 % (Li et al. 2019) and 18.28 % (Liu et al. 2021) on the KITTI 3D detection benchmark for "car,easy"[1], as shown in Table 6.8. Note, that a direct comparison is not possible, since the datasets have massive differences in terms of object positions and distances. However, it indicates that the performance of the developed approach within its FoV operates on a level with state-of-the-art approaches. The utilization of this camera position for object detection therefore offers a huge potential for monitoring the close range area in front of the truck.

Table 6.8: Comparison of mAP scores of detection approaches on the KITTI dataset and the proposed approach on truck data.

Approach	Dataset	Car		
		Easy	Mod	Hard
YOLO3D (Liu et al. 2021)	3D KITTI	18.28	12.06	8.42
GS3D (Li et al. 2019)	3D KITTI	4.47	2.90	2.47
MonoGRNet (Qin et al. 2021)	3D KITTI	9.61	5.74	4.25
Mono3D++[a] (He and Soatto 2019)	3D KITTI	10.6	7.9	5.7
downward cam approach	3D truck		45.28	
YOLO3D (Liu et al. 2021)	2D+AOS KITTI	91.43	78.50	58.80
Deep MANTA (Chabot et al. 2017)	2D+AOS KITTI	98.83	93.31	82.95
Deep3DBox (Mousavian et al. 2017)	2D+AOS KITTI	94.62	89.88	76.40
GS3D (Li et al. 2019)	2D+AOS KITTI	85.79	75.63	61.85
MonoGRNet (Qin et al. 2021)	2D KITTI	88.65	77.94	63.31
downward cam approach	2D truck		95.4	

[a]Results not officially listed

6.5 Evaluation metrics for object tracking

The prerequisite for comparing developed approaches on MOT datasets is a meaningful metric that covers all differences from the ground truth. Depending on the dataset and the task, different metrics are used in this thesis. For the KITTI dataset, the Multiple Object Tracking Accuracy (MOTA) metric (Bernardin and Stiefelhagen 2008) and the Higher Order Tracking Accuracy (HOTA) metric (Luiten et al. 2021) are often used, both of which are calculated using the intersection over union as base distance. In addition to these traditional metrics, however, there are others, such as the Second Order Optimal Sub-Pattern Assignment (OSPA$^{(2)}$) metric, created for arbitrary sets and probability distributions.

Rezatofighi et al. (2020) compare these metrics in terms of their trustworthiness. This is based on the robustness of the result to parameter variation, how meaningful they behave in sanity tests and analytical properties, like their mathematical consistency. In the experiments of Rezatofighi et al. (2020), the traditional metrics seem to be less robust to parameter variations, get worse results in the sanity tests and cannot fulfill the criteria for mathematical consistency. In particular, the identity criteria, and the triangle inequality may be violated (Rezatofighi et al. 2020). The identity criteria states that a metric $d(\boldsymbol{x}, \boldsymbol{y})$ of GT $\boldsymbol{x}$ and prediction $\boldsymbol{y}$ is $d(\boldsymbol{x}, \boldsymbol{y}) = 0$ only if $\boldsymbol{x} = \boldsymbol{y}$. The triangle inequality ensures that if a prediction $\boldsymbol{z}$ is close to a prediction $\boldsymbol{y}$, which is close to GT $\boldsymbol{x}$, the prediction $\boldsymbol{z}$ is also close to $\boldsymbol{x}$ by stating $d(\boldsymbol{x}, \boldsymbol{z}) \leq d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{z})$. The mathematical inconsistency of traditional metrics is rooted in the threshold for generating true positives (Rezatofighi et al. 2020). Overall, these drawbacks can potentially lead to false interpretations during evaluations when purely relying on the

[1]Includes only objects of class "car" with 2D bounding box height of minimum 40 pixels and without occlusion

traditional metrics, especially since the labels of the truck datasets used in this thesis are handmade based on measurements and therefore rather represent an approximate truth than a ground truth.

Rezatofighi et al. (2020) show some alternative metrics, of which especially the OSPA$^{(2)}$ metric stands out with good results while being mathematically consistent. To be able to compare the tracking results with state-of-the-art approaches on the KITTI dataset, the HOTA metric is still used in this thesis, while the OSPA$^{(2)}$ metric is used in addition and for comparison under mathematical consistency. The OSPA$^{(2)}$ metric is based on the OSPA metric, while all of them (including OSPA) are using a base distance measuring the similarity between two objects. The following paragraphs give an overview of common base distances, while Figure 6.5 provides an overview of the Euclidean distance $d_{b,euclid}$, the IoU and the Generalized Intersection over Union (GIoU), which are explained in the following paragraphs. In the example image, a GT bounding box in green is compared to a detected bounding box "Det" in red.

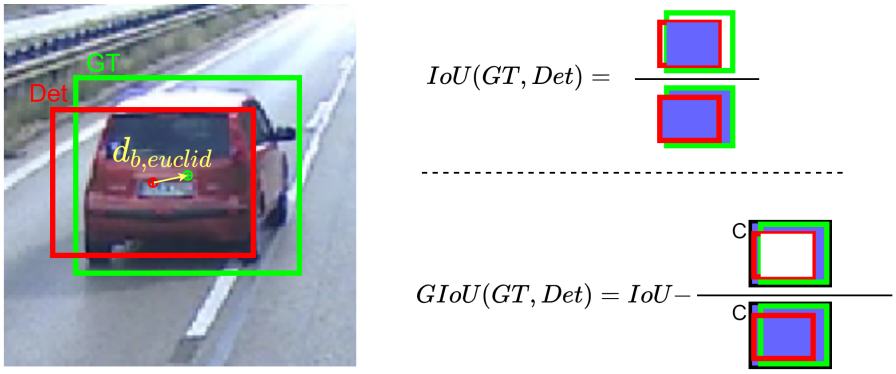


Figure 6.5: Proposed base distances in comparison for a 2D bounding box in image coordinates. Note, that these metrics can also be applied to other types of shapes and volumes, like OBBs.

The Euclidean distance is a natural choice for a base metric, since it is the distance between two points in Euclidean space and can therefore directly give the distance between two state vectors:

$$d_{b,euclid}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \sqrt{\sum_{l=1}^{N} (\boldsymbol{x}(l) - \hat{\boldsymbol{x}}(l))^2}. \tag{6.7}$$

The Euclidean distance is intuitive and works well for determining the distance between two points. However, it is not suitable for evaluating the similarity of geometric shapes. In Figure 6.5, it is shown on the left side comparing the OBB centers.

The Intersection over Union (IoU), which was introduced in chapter 6.2, can be used to calculate the base metric $d_{b,IoU}$ Due to the normalization, the IoU is invariant to the scale (Rezatofighi et al. 2019). The IoU distance $d_{b,IoU}$ of two shapes A and B with $IoU \in [0, 1]$ is defined by

$$d_{b,IoU}(A, B) = 1 - IoU(A, B). \tag{6.8}$$

The IoU is among the most popular metrics in the area of object detection (Rezatofighi et al. 2019), but is not suited for the evaluation of non-overlapping shapes, since the distance for these cases will always be $IoU = 0$, regardless of the actual spatial distance.

The Generalized Intersection over Union (GIoU) (Rezatofighi et al. 2019) compensates this weakness by introducing a third shape C from the same type of A and B, which is defined as the smallest convex shape enclosing both A and B. The GIoU is then defined as the IoU minus the area or volume of C excluded the Union of A and B, normalized to the area or volume of C. It is therefore also invariant to the scale, always lower bound to the IoU and ranging from $GIoU \in (-1, 1]$ (Rezatofighi et al. 2019). The base distance $d_{b,GIoU}(A, B)$ is then defined by

$$d_{b,GIoU}(A, B) = \frac{1 - GIoU(A, B)}{2}, \tag{6.9}$$

$$GIoU(A, B) = IoU - \frac{|C \setminus (A \cup B)|}{|C|}. \tag{6.10}$$

$$\tag{6.11}$$

The GIoU is a good indicator for evaluating both overlaps and non-overlaps, if a scale invariance is desired. In general, other metrics based on the IoU are possible, as shown with different loss functions by Zheng et al. (2020).

6.5.1 The OSPA metric

If a whole set of states needs to be evaluated in the context of multiple object filtering, a single value of distance to the ground truth is desired. However, single target distances, like the previously discussed base distances, do not fulfill this requirement. Instead, both the states and the number of objects need to be incorporated. The Optimal Sub-Pattern Assignment (OSPA) metric (Schuhmacher et al. 2008) is designed to give a single distance between two finite sets of objects, and was developed to overcome the weaknesses of the Optimal Mass Transfer (OMAT) (Hoffman and Mahler 2004) metric. The OSPA metric is calculated using a base distance d_b between two states $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ or two shapes A and B. Each of the proposed base distances can be used. The cut-off base distance d_c with $c > 0$ is then limiting the base distance using the cut-off parameter c:

$$d_c(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \min\{c, d_b(\boldsymbol{x}, \hat{\boldsymbol{x}})\} \tag{6.12}$$

For two sets $X = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\}$ and $\hat{X} = \{\hat{\boldsymbol{x}}^{(1)}, \ldots, \hat{\boldsymbol{x}}^{(m)}\}$ with $m \leq n$, the OSPA distance $d_p^{(c)}$ of order $1 \leq p \leq \inf$ is defined as follows:

$$d_p^{(c)}(X, \hat{X}) = \left(\frac{1}{n} \left(\min_{\pi \in \Pi_n} \sum_{i=1}^{m} d_c(\boldsymbol{x}^{(i)}, \hat{\boldsymbol{x}}^{(\pi(i))})^p + c^p(n - m) \right) \right)^{\frac{1}{p}}. \tag{6.13}$$

Π_n is the number of permutations, which means that the global minimum distance of m-m assignments is calculated. If using higher orders, outliers are more penalized, however, in this thesis the order of $p = 1$ is used. For a more profound error analysis, the metric can be divided into cardinality error $d_p^{(c,card)}$ and localization error $d_p^{(c,loc)}$:

$$d_p^{(c,loc)}(X, \hat{X}) = \left(\frac{1}{n} \min_{\pi \in \Pi_n} \sum_{i=1}^{m} d_c(\boldsymbol{x}^{(i)}, \hat{\boldsymbol{x}}^{(\pi(i))})^p \right)^{\frac{1}{p}}, \tag{6.14}$$

$$d_p^{(c,card)}(X, \hat{X}) = \left(\frac{c^p(n - m)}{n} \right)^{\frac{1}{p}}. \tag{6.15}$$

This division allows to determine, if the main contribution to the overall error is based on the state or cardinality estimation. For a proper analysis with this metric, a suitable cut-off value c has to be chosen, since large values of c can lead to a high penalization of the cardinality error compared to the localization error.

Note that the OSPA metric is only suitable for the evaluation of multi-object filter problems, and not tracking problems. It compares the distance of two sets separately for each time step and thus only covers localization and cardinality, without considering the ID of individual tracks. ID switches and track continuity are therefore not evaluated.

6.5.2 The OSPA$^{(2)}$ metric

To overcome the limitations of the OSPA metric for tracking applications, the OSPA$^{(2)}$ uses a base distance measuring the distance between tracks instead of set elements (Beard et al. 2020). The time-averaged OSPA distance between two tracks f and g is the base distance. Here, $|D|$ is the cardinality of D and $D_f \subseteq \mathbb{T}$ and $D_g \subseteq \mathbb{T}$ are sets of time instants, where the corresponding object has a state (Rezatofighi et al. 2020). $\mathbb{T}$ is a time-window with fixed length T, in which the track evaluation and averaging is performed. The averaging is done over $D_f \cup D_g$ to avoid calculation errors when parts of either track have no valid state:

$$\tilde{d}^{(c)}(f,g) = \begin{cases} \sum_{t \in D_f \cup D_g} \frac{d^{(c)}(\{f(t)\},\{g(t)\})}{|D_f \cup D_g|}, & \text{if } D_f \cup D_g \neq \emptyset \\ 0, & \text{if } D_f \cup D_g = \emptyset \end{cases} \tag{6.16}$$

with

$$d^{(c)}(\phi,\psi) = \begin{cases} 0, & |\phi| = |\psi| = 0 \\ c, & |\phi| \neq |\psi| \\ \min(c, d_b(\phi^{(1)}, \psi^{(1)})), & |\phi| = |\psi| = 1 \end{cases}. \tag{6.17}$$

Now, the OSPA$^{(2)}$ distance $\check{d}_p^{(c)}$ is calculated similar to the standard OSPA distance with $m \leq n$ and $1 \leq p \leq inf$ (Beard et al. 2020):

$$\check{d}_p^{(c)}(f,g) = \left(\frac{1}{n} \left(\min_{\pi \in \Pi_n} \sum_{i=1}^{m} \tilde{d}^{(c)}(f^{(i)}, g^{(\pi(i))})^p + c^p(n-m) \right) \right)^{\frac{1}{p}}. \tag{6.18}$$

Overall, the OSPA$^{(2)}$ metric incorporates errors from localization and the shape (depending on base metric), as well as cardinality, track fragmentation and identity switching (Beard et al. 2020). If the metric is plotted against time, the time window $\mathbb{T}$ may include the last T valid time-steps.

6.5.3 The HOTA metric

The Higher Order Tracking Accuracy (HOTA) (Luiten et al. 2021) metric is a state-of-the art MOT metric, that is used at the KITTI dataset to compare approaches to each other. Therefore, using this metric allows the direct comparison of the presented approaches to others.

(Luiten et al. 2021) classifies errors in MOT applications in the following categories:

- Detection errors: The system either misses detecting an object that is present in the GT data or detects an object that is not present in the GT data.
- Association errors: The unique identifier (ID) of a detection by the tracker changes when the corresponding GT ID does not or the ID of a detection does not change, when the corresponding GT ID does.
- Localization errors: The spatial error of a detection and the corresponding GT object. In this thesis, it is defined as the IoU between two 3D OBBs.

Classification errors are not directly evaluated by the HOTA metric. However, if the metric is applied separately to objects of each individual class, classification errors are implicitly represented.

The HOTA metric is designed to create an equal weighting between detection, association and localization error, since previous metrics tend to overemphasize either of them (Luiten et al. 2021). Another advantage of this metric is the ability to split the result into sub-metrics including the localization, which can help to understand the tracking performance in more detail.

Similar to the metrics described for object detection in chapter 6.2, a detection is defined TP, if the IoU with a GT object exceeds the threshold α. A GT object without a matched detection is a FN, and a detection without a matched GT object is a FP. Each TP is further evaluated for association in a similar way. A set of True Positive Association (TPA) is defined as the TPs that share the same GT ID (gtID) and detection ID (dtID) with the TP c (Luiten et al. 2021):

$$\text{TPA}(c) = \{k\}, k \in \{\text{TP}|\text{dtID}(k) = \text{dtID}(c) \wedge \text{gtID}(k) = \text{gtID}(c)\}. \tag{6.19}$$

Similarly, a set of False Positive Association (FPA) is the set of TP detections that share the dtID with c but not the gtID, combined with the set of FP detections that share the dtID with c:

$$\begin{aligned} \text{FPA}(c) =& \{k\}, \\ & k \in \{\text{TP}|\text{dtID}(k) = \text{dtID}(c) \wedge \text{gtID}(k) \neq \text{gtID}(c)\} \\ & \cup \{FP|\text{dtID}(k) = \text{dtID}(c)\}. \end{aligned} \tag{6.20}$$

A set of False Negative Association (FNA) is the set of TP detections that share the gtID with c but not the dtID, combined with the set of FN detections that share the gtID with c:

$$\begin{aligned} \text{FNA}(c) =& \{k\}, \\ & k \in \{\text{TP}|\text{dtID}(k) \neq \text{dtID}(c) \wedge \text{gtID}(k) = \text{gtID}(c)\} \\ & \cup \{FN|\text{gtID}(k) = \text{gtID}(c)\}. \end{aligned} \tag{6.21}$$

Based on the detection measurements TP, FP, and FN and the association measurements TPA, FPA and FNA, the HOTA is calculated by integrating over a range of IoU thresholds α (Luiten et al. 2021):

$$\text{HOTA} = \int_0^1 \text{HOTA}_\alpha \, d\alpha \approx \frac{1}{19} \sum_{\alpha \in \{0.05, 0.1, \ldots, 0.95\}} \text{HOTA}_\alpha. \tag{6.22}$$

The HOTA_α value is calculated based on both the evaluation of the association and the detection given the IoU threshold α:

$$\text{HOTA}_\alpha = \sqrt{\frac{\sum_{c \in \{\text{TP}\}} \mathcal{A}(c)}{|\text{TP}| + |\text{FN}| + |\text{FP}|}}. \tag{6.23}$$

Each of the TPs is evaluated for the correct association:

$$\mathcal{A}(c) = \frac{|\text{TPA}(c)|}{|\text{TPA}(c)| + |\text{FNA}(c)| + |\text{FPA}(c)|}. \tag{6.24}$$

The HOTA metric can further be decomposed into the following sub-metrics for in-depth evaluation. The formulas for the calculation can be found in the description from Luiten et al. (2021):

- The localization accuracy "LocA" is a measure for the pure spatial alignment between the tracker output and the GT.
- The detection accuracy "detA" is a measure for the pure detection alignment between the tracker output and the GT data, which can further be divided into the detection precision and detection recall.
- The association accuracy "AssA" is a measure for the pure association accuracy, which can further be divided into the association precision and association recall.

6.5.4 MOT metric summary

The presented metrics have different approaches, advantages and disadvantages, which are summarized in the following list:

- The HOTA metric is particularly useful for comparing tracking approaches against state-of-the-art for public datasets, such as KITTI. The metric is well established here and is used by many datasets. It allows an intuitive interpretation by presenting the result in the range [0,1] and does not require any parameters. By decomposition into sub-metrics, partial aspects can also be interpreted. Compared to the other metrics, however, no statement can be made about the accuracy with non-overlapping bounding boxes and due to thresholding, the metric is not mathematically consistent.
- The OSPA$^{(2)}$ metric satisfies the criteria of mathematical consistency (Rezatofighi et al. 2020) and allows the application of different basis metrics. Thus, the metric is not limited to bounding boxes and can also compute results for non-overlapping geometric sets. The result of the metric is a distance and depends on the parameters T and c, which must be set manually. If the IoU or the GIoU are used as the base metric, the best choice is for c is $c = 1$. A single value for a sequence can be determined if $\mathbb{T}$ contains all time-steps of this sequence or the average result for all time-steps is used.
- In contrast to the other two metrics, the OSPA metric only calculates the distance between two sets. Thus, it does not allow any statement about the quality of the association when using continuous tracks. The metric is therefore particularly suitable for evaluating detection and filtering without a label, but can also be used for tracking situations when the association errors are not to be taken into account. For the analysis of whole sequences, an average value can be calculated over all time-steps.

Each of the presented metrics attempts to evaluate very complex scenarios using single values. In the case of MOT with sensor fusion, correct interpretation can therefore still be challenging, since the quality of the complete environment detection around the ego-vehicle is determined with the help of a single value.

On the level of the ADAS functions, individual situations must also be analyzed to find weaknesses and errors. However, for a basic comparison of the performance of different tracking and fusion approaches using a large dataset, such metrics are the best option.

6.6 Simple multi-object filter comparison

Chapter 4 shows the basics of MOT with different filtering approaches, like the Kalman filter and some RFS filters. These are to be compared to get a first clue about the performance with real-world data and to make an informed decision on which filter to use in the fusion framework. Similar comparisons between different types of RFS filters can be found in the literature (Vo et al. 2009; Clark et al. 2006; Lu et al. 2017), but most of them focus on simulated data that may not be comparable to real-world data from Lidar sensors or cameras in automotive environments. Therefore, this thesis performs real-world tests using the KITTI dataset, which allows more substantiated statements due to the higher amount of data compared to the truck dataset.

In contrast to the evaluations in chapter 6.7, this evaluation compares the pure multi-object filtering capabilities. Therefore, the RFS filters, like the GM-PHD filter or GM-CPHD filter, are implemented without an additional framework to propagate unique IDs and to retrieve full track trajectories. The OSPA metric is used for comparison, which excludes any ID-related information from the evaluation. In addition, the filters are implemented "plain" without additional pre- or post-processing. This overall reduced complexity should help to get an impression about the pure capabilities of the filters without the influence of any heuristics of the implemented framework. The KF is an exception, as it is not able to perform multi-object filtering without a basic tracking framework with data association.

The comparison only uses objects of the class "Car" to simplify the dataset and avoid classification influences. The measurements from the Lidar detector Point-RCNN (Shi et al. 2019) are used for a realistic measurement representation. The filters are implemented in Matlab, and share the same implementation structure, which is based on the implementation of Vo (2023).

The comparison is carried out as point tracking, which means that the measurements are represented as single, 2-dimensional points that follow a 2D constant velocity state space model with the following state vector $\boldsymbol{x}$ and measurement vector $\boldsymbol{z}$. Note that for this first and simple comparison, simplified state and measurement vectors are used compared to the developments from Chapter 5. Only the basic filter performance is compared here, without the need to estimate additional properties such as the dimensions:

$$\boldsymbol{z}_K = \begin{bmatrix} p_x & p_y \end{bmatrix}^T, \tag{6.25}$$

$$\boldsymbol{x}_k = \begin{bmatrix} p_x & p_y & v_x & v_y \end{bmatrix}^T. \tag{6.26}$$

Therefore, all filters follow a linear Gaussian model and use the same measurement and noise covariance $\boldsymbol{Q}$ and $\boldsymbol{R}$, as well as the same initial covariance $\boldsymbol{P}_0$. As far as possible, the other parameters, like the detection probability p_D, the survival probability p_s, and others are the same for all filters. Therefore, the filters can be compared using the same setup and parameters. The following paragraphs give an overview of the implementation details of all compared filters.

Kalman filter: The KF implementation uses a GNNDA based on the Hungarian algorithm and the Mahalanobis distance to use a similar setup as used by the implementation proposed in chapter 5.8. This is supported by a preceding gating process as described in 4.3.2 and by Bader (2019). For each track, the existence probability is calculated. This leads to a good comparability, since some RFS approaches use similar concepts, while for others an abstraction of the weights can be used. Tracks with an existence probability below a threshold are deleted, while tracks with a value above a threshold are reported to be confirmed. The states of these confirmed tracks are then fed into the evaluation.

PHD and CPHD filter: The PHD (Vo and Ma 2006) and CPHD (Mahler 2007a) filters are implemented as Gaussian mixture filters based on the implementation from Vo (2023). For this test, no adaptive birth intensity, sensor based parameter model, classification or tags for preserving the ID are used. Instead, a birth model with uniform distribution of 4 Gaussian components over the observed area is used. The mean values of the n Gaussian components with the highest weights are extracted as valid objects and used by the evaluation, with n being the rounded estimated cardinality.

CBMeMBer filter: The CBMeMBer filter (Vo et al. 2009) is also implemented as a Gaussian mixture approximation based on the implementation from Vo (2023). For this test, the birth process is modeled by a Bernoulli density with 4 components distributed uniformly over the observed area. Similar to the PHD Filter, the mean values of the n Bernoulli components with the highest existence probability r are extracted as valid objects and used by the evaluation, with n being the rounded estimated cardinality.

The comparison is done using the OSPA metric, since it is not possible to use the HOTA or OSPA$^{(2)}$ metric without unique IDs for each track. The OSPA distance, the OSPA localization, the OSPA cardinality and the average processing time per frame in milliseconds are compared for all filters. The results are compared for each of the 21 training set sequences separately. All evaluation metrics are compared using the average over all frames of each sequence, with an OSPA cutoff value of $c = 2.5$. All objects inside "DontCare" regions or outside the FoV defined by the KITTI dataset are excluded from the evaluation. Figure 6.7 shows the results on the training set of the KITTI tracking dataset for each of the 21 sequences. The top plot shows the comparison of the OSPA distance for the four filters, while the second shows the OSPA localization sub-metric and the third

the OSPA cardinality. The last plot shows the average runtime per frame using the MATLAB implementation on an AMD Ryzen 7 5700U CPU.

The OSPA distance computed by the KF tends to exceed that of the RFS filters across the majority of sequences. In particular, in sequences 8 and 18, significant deviations from the RFS filters are observed, primarily due to inaccurate cardinality estimation, as shown by the third plot. Both are sequences that include fast, oncoming vehicles, which are difficult for the gating and association process after initialization, where the velocities are not yet known. Since RFS filters jointly estimate the cardinality and states and do not rely on association, this is less of an issue. Note, that these issues can be handled in advanced KF frameworks with proper initialization and exceptions. However, this experiment should provide a comparison of the basic performance without any special handling and advanced techniques.

Within the group of RFS filters, the PHD filter has the lowest OSPA distance in most sequences. In most of these sequences, both the OSPA cardinality and the OSPA localization show lower values compared to the other filters, which indicates the suitability for both existence and state estimation. The bottom plot shows that the KF is clearly the fastest one for each sequence. This is no surprise, since the KF uses less mathematical operations in the update step and the data association is still fast for the used number of measurements per frame in this example. On average, the PHD filter and the CPHD filter follow, while the CBMeMBer filter is the slowest. This is shown by Table 6.9, which shows the average results of the runtime and the mean OSPA values over all 21 sequences. The runtimes in the range of some milliseconds show that each of these filters is capable of real-time online processing.

Table 6.9: Overall mean results of filter comparison on KITTI dataset.

	OSPA Dist	OSPA Loc	OSPA Card	Mean Runtime [ms]
KF	1.17	0.23	0.94	2.61
CBMeMBer	0.80	0.20	0.60	7.73
PHD	0.66	0.16	0.51	4.31
CPHD	0.85	0.21	0.64	6.15

As shown be Table 6.9, the PHD filter on average outperforms the other implementations in all measures of the OSPA metric, while the CBMeMBer filter achieves the second-highest performances. This only partially meets the results from Vo et al. (2009), who show the best results with the CPHD filter for simulated low clutter situations with Gaussian mixture implementations of the filters. The difference between simulated and real measurement data could be the reason in this experiment: The ground truth cardinality of the used real-world data frequently changes within most sequences. Figure 6.6 shows the GT OSPA cardinality for sequence 4 as an example in combination with the estimations from the CPHD (left) and PHD (right) filters. As indicated by the red-marked areas, the CPHD filter tends to estimate the cardinality more stable, but also reacts slowly to cardinality changes, resulting in long periods of false cardinality estimates. It often takes multiple time steps to adapt the estimated cardinality to the quickly changing number of objects. This is a main contribution to the lower result compared to the PHD filter.

The KF underperforms compared to the other filters and clearly achieves the lowest results. The main difference is the poorer estimation of the cardinality. However, the KF already provides track labels within this implementation. For an in-depth comparison, the filters have to be evaluated on a metric that includes labels, in order to also evaluate the errors from the label extraction process of the RFS based filters, which is shown in chapter 6.7.

In summary, this comparison shows the potential of RFS filters and specifically the PHD filter for real-world MOT problems in automotive environments. RFS-based filters can outperform classical KF pipelines for online tracking and are still real-time capable if Gaussian mixture implementations are used. Therefore, the GM-PHD filter is further evaluated against a classic KF approach in the more sophisticated framework proposed in chapter 5.

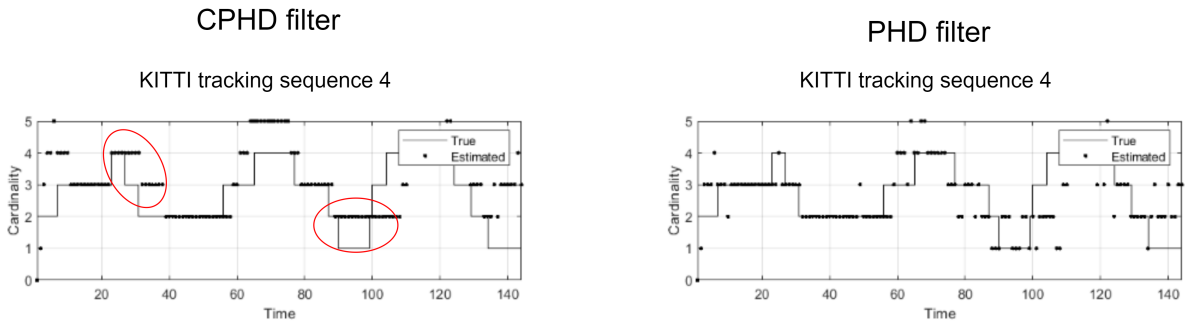


Figure 6.6: Comparison of cardinality estimation of GM-CPHD and GM-PHD filter on KITTI dataset sequence 4. Situations of the GM-CPHD filter reacting slowly to changes are marked in red.

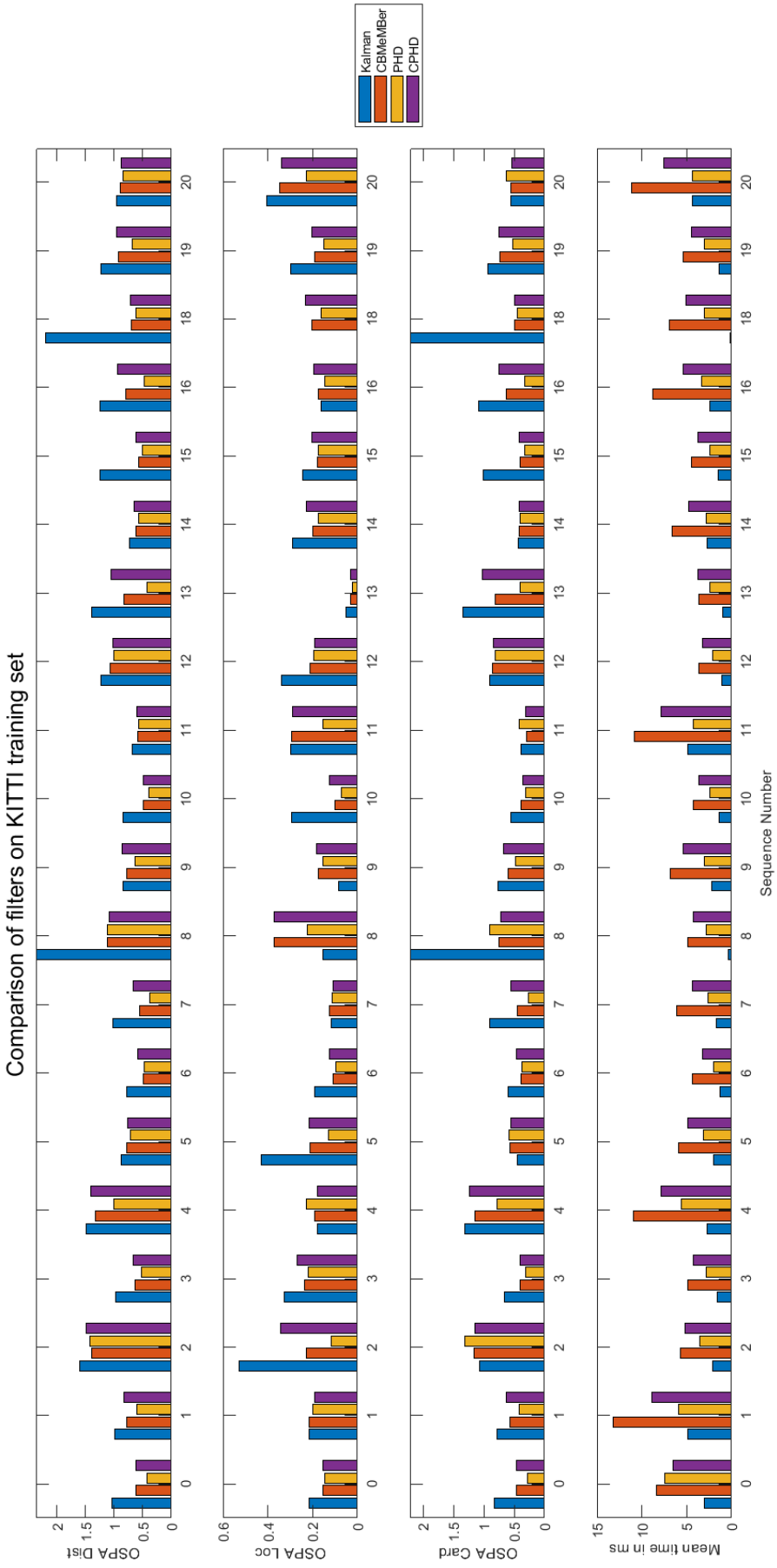


Figure 6.7: Comparison of filter approaches on the KITTI dataset using objects of class "Car". The OSPA results and the average runtime per time-step are shown for each sequence.

6.7 Fusion framework evaluation

This chapter evaluates the fusion framework proposed in chapter 5 both using the KITTI dataset and the new truck dataset. It shows the effectiveness of the developed approaches and gives a comparison between the KF and the GM-PHD filter in an advanced multi-sensor multi-object framework. This is in particular valuable, since direct comparisons of both approaches are rare in literature and have not been done on multiple real-world automotive datasets. Since the comparison is applied to multiple datasets, multiple sensor combinations and single sensors with and without the proposed developments, it offers a rich decision basis for the usage of the approaches in real-world systems, which is currently missing in literature. Table 6.10 shows an overview of the performed experiments and indicates on which datasets the experiments are applied.

Various experiments are applied on the KITTI dataset. Since the dataset is widely used and well researched, it is particularly well suited for comparisons of new approaches and the literature. Since the GT data are not available for the test set, the training set is used for evaluation. The evaluations on the KITTI dataset focus on the class "Car", since it is by far the most frequent in the dataset and the used detection approaches do not have frequent misclassifications. Therefore, all detections from other classes are excluded prior tracking, which makes the tracking process a single-class problem. After the tracking is performed, an offline filtering step removes all tracks with less than five confirmed time-steps or an average weight or existence probability below 0.5. The analysis on the KITTI dataset mainly uses the HOTA metric to be comparable to the literature and the OSPA$^{(2)}$ metric for confirmation of the results. For all OSPA$^{(2)}$ calculations, a cutoff value of $c = 1$ is used, since this is the maximum possible distance between two objects when using the base distance GIoU. The order $p = 1$ is used to not penalize outliers by a higher order and the window length of $T = 10$ is found empirically.

Compared to the KITTI dataset, the truck dataset differs in several ways, as shown in chapter 6.1. Instead of OBBs, the SRR sensors deliver L-shaped objects, while the camera and LRR deliver point objects with an estimated width. For the Lidar sensor, the algorithm proposed in chapter 3.2.3 is used, which generates OBBs. One major difference for the evaluation is, that the truck dataset is calculated as a multi-class problem, since the sensors produce frequent misclassifications. Therefore, all detections of different classes are tracked with the same filter and the classification is estimated by the framework. Due to the low number of cyclist and pedestrian examples, the metric is still only calculated for the class "car". However, the classification accuracy is implicitly evaluated, since misclassifications with "car" tracks are taken into account. The smaller dataset size and the differences in labeling also have to be noted.

In contrast to the KITTI dataset analysis, where the HOTA metric on image coordinates is mainly used, the truck dataset analysis only uses the OSPA$^{(2)}$ with the base metric GIoU. The reason is that some of the sensors used do not detect bounding boxes, but only points in space, which makes the correct bounding box estimation difficult to compare. Therefore, OSPA$^{(2)}$ metric is mainly used, since the GIoU includes bounding box information combined with distance information. Here, the same hyperparameters as for the KITTI dataset with $c = 1$, $p = 1$ and $T = 10$ are used. The investigations using the truck dataset also focus more on the current series generation of perception sensors in a more general setup compared to the KITTI dataset. The used sensors have, in general, lower detection capabilities with a higher amount of clutter compared to the highly optimized sensors and detectors of the KITTI dataset.

Table 6.10: Overview of the experiments on KITTI and truck dataset, alongside with their goals.

Experiment	KITTI	Truck	Goals
Comparison of GM-PHD and KF	✓	✓	• Comparison of overall performance ceiling within the proposed framework. • Comparison of the performances for different single sensors and fusion setups.
Evaluation of sensor-based parameter models	✓	✓	• Evaluation of effectiveness of different levels of parameter models. • Comparison of influence on different sensors and datasets.
Evaluation of detection probability simplification and gating	✓	✗	• Evaluation of performance and runtime differences between proposed detection probability implementation and original one. • Experimental proof that the filter is working as expected under proposed assumptions. • Evaluation of performance and runtime differences using proposed gating process. • Evaluation of runtime advantages of proposed approaches.
Sensor failure analysis	✗	✓	• Test of influence of single sensor failure in shared FoV. • Prove of system robustness in sensor failure scenario. • Only applied to truck dataset due to higher amount of sensors.
Comparison of operational domains	✗	✓	• Comparison of performance for different operational domains. • Applied to truck dataset due to clearly distinguishable domains.
Comparison of classification fusion approaches	✗	✓	• Comparison of performance using different classification fusion approaches. • Comparison of runtime of different classification fusion approaches. • Only applied to truck dataset since this is calculated as multi-class problem.

6.7.1 Evaluation areas

Although the evaluation with an MOT metric gives a result for the evaluation of the overall system, misleading effects may occur. In particular, if the FoV of the sensor set and the GT does not match each other, unavoidable errors will be created. This also complicates the comparison of sensor sets with different FoVs or the evaluation of tracks outside the labeled area.

Therefore, a simple approach to create evaluation areas is proposed here. This allows to only evaluate specific areas instead of the whole surrounding of the ego vehicle. Before the metric is applied, a preprocessing step filters out both ground truth and tracking results outside the evaluation areas for each time-step. Therefore, all objects outside the evaluation area are ignored and have no influence on the results.

Two evaluation areas are used on the truck dataset, which are the full area shown on the left in Figure 6.8, which is a $\pm 90°$ FoV and a narrow area on the right, which has $\pm 20°$ FoV, both with a maximum distance of 70 m. The full area represents the Lidars FoV which is used for the GT label generation, while the narrow area is based on the camera and LRR sensor's central region and is shared by all sensors except a small blind spot of the SRRs. The evaluation is limited to 70 m since the Lidar's point density drops to a low level above that distance, where a correct generation of GT labels cannot be ensured.

The following procedure ensures to ignore the objects outside the evaluation area: For each of the ground truth element, the closest track within a cutoff value using on the GIoU is found to form a pair. For these pairs, the ground truth is the baseline for both objects to be labeled as inside or outside the evaluation area. The pairwise approach can resolve ambiguous situations at the edges of the evaluation areas, that can occur if a track lies outside and the corresponding ground truth inside the area or vice versa. Next, all remaining GT and track objects are evaluated to be inside or outside the area separately.

During the experiments on the truck dataset, the full area is used except the experiments from chapter 6.7.5. The KITTI Dataset does not require this filtering, since both detections and GT objects only occur in the shared FoV of the camera.

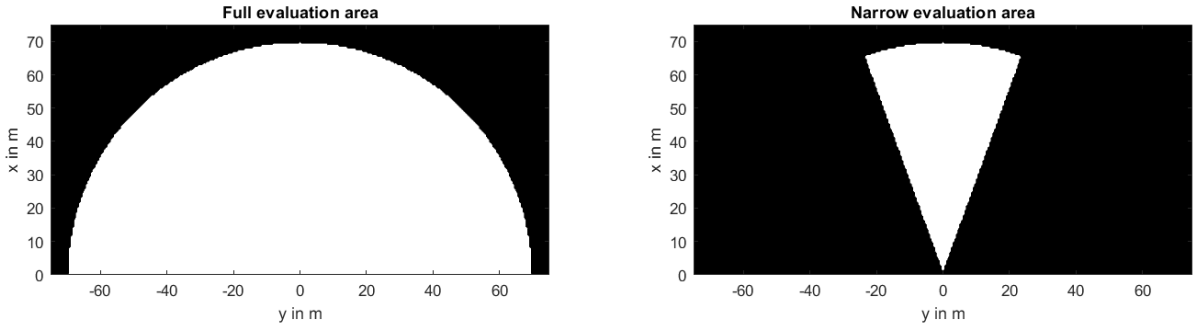


Figure 6.8: Visualization of the evaluation areas. Full area on the left side, narrow area on the right side.

6.7.2 Comparison of GM-PHD and Kalman filter using real-world data

First, the overall performance of the GM-PHD and the KF is compared within the proposed framework from chapter 5. Previous comparisons do not provide direct comparisons of these filters with real-world automotive data in a realistic MOT framework with multiple sensors. On the KITTI dataset, both filters are evaluated in three input scenarios: Using camera only, Lidar only and combined detections from Lidar and camera.

On the KITTI dataset, this comparison is using the HOTA metric to be comparable to the literature. Figure 6.9 shows the overall HOTA performance over the 21 sequences of the KITTI tracking training set on the class "Car" in image coordinates similar to the results on the KITTI leaderboard. The evaluation shows that the HOTA performance of the GM-PHD filter is higher for Lidar only and fused data. For camera only data, however, the KF performs slightly better. A possible explanation is the low amount of clutter in the camera data, which minimizes the chance of false associations and increases the performance of the KF.

The results of the fused tracker are in the range of the top ten KITTI leaderboard (07/2022) of the multi-object tracking challenge and comes close to the HOTA of 77.80 % from Wu et al. (2022), which uses a complex data

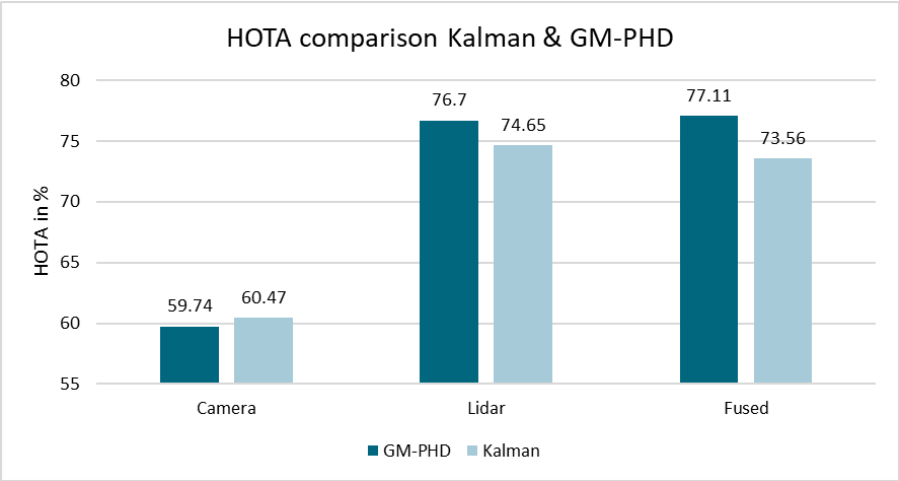


Figure 6.9: Comparison of GM-PHD and KF using HOTA metric on the KITTI dataset.

association scheme in combination with a KF. The proposed GM-PHD achieves a similar result, but is much more general and can be easily applied to other applications and sensor setups.

When analyzing the HOTA sub-metrics in Figure 6.10, it is noticeable that the spatial alignment represented by the localization accuracy (LocA) is comparable for both filters. Therefore, the state estimation of the approaches performs on a similar level, while the differences in the results are originated in the existence estimation and the track continuity. There are noticeable differences in the association accuracy (AssA) sub-metric and the detection accuracy (DetA). Therefore, the existence estimation and the track continuity seem to be the main advantage of the GM-PHD filter compared to the KF. This is no surprise, since it is estimated on a global basis for the GM-PHD filter compared to the separate estimation of the KF.

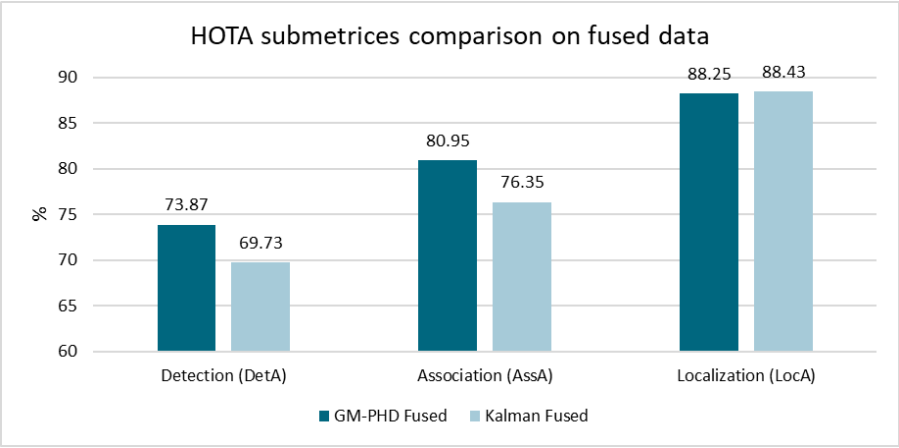


Figure 6.10: Comparison of GM-PHD and KF using the HOTA submetrics on the KITTI dataset for deeper insight.

The performance advantage of the GM-PHD filter is confirmed by an analysis using the OSPA$^{(2)}$ metric in combination with the GIoU base metric on the BEV of the OBBs. For a better visual comparison to the HOTA metric, a OSPA$^{(2)}$ score of $1 - OSPA^{(2)}$ is used in Figure 6.11. It confirms the performance advantage of the GM-PHD filter in the Lidar and fusion scenario, with the slight advantage of the KF for the camera only scenario.

Similar experiments are also conducted using the truck dataset with multiple setups. These include the single sensor performances and their fusion. Since the GT labels are generated based on the Lidar raw data, the

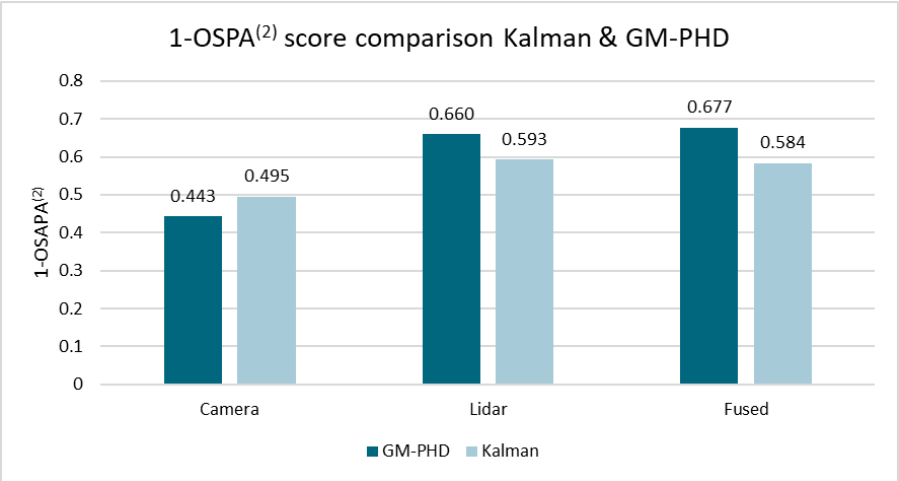


Figure 6.11: Comparison of GM-PHD and KF using the mean OSPA$^{(2)}$ score on the KITTI dataset. Displayed as 1-OSPA$^{(2)}$ for visual comparison to HOTA.

fusion of all sensors except the Lidar is evaluated as well. The OSPA$^{(2)}$ with GIoU as base metric is used for evaluation.

All tested setups use the developed parameter model for the detection probability, as well as the proposed Dempster-Shafer classification fusion method from chapter 5.6. As shown by Figure 6.12, the GM-PHD filter with constant models for detection probability and clutter density outperforms the KF in most of the setups. The LRR and camera setups are the only ones, where the KF achieves the higher scores. These sensors detect a low amount of objects with low clutter, which is easier to handle for association schemes. Similar to the KITTI dataset, a noticeable margin can be found for the fusion situations, where the GM-PHD filter outperforms the KF.

It must be noted that this experiment is meant to compare the tracking and fusion capabilities of the GM-PHD and the KF. Due to different FoVs, different object representations and capabilities in classification, it does not make sense to compare individual sensors (e.g. LRR vs. camera) based on the presented results.

Overall, the GM-PHD filter implementation achieves higher scores on the KITTI dataset in the comparison, when comparing the Lidar only and fusion results using both the HOTA and OSPA$^{(2)}$ metric. The experiments on the truck dataset confirm the ones from the KITTI dataset, where the GM-PHD outperforms the KF in most setups, in particular the fusion setups. In certain configurations with a single sensor, the KF may offer superior performance, but in the majority of cases and especially in fusion setups, the GM-PHD filter proves to be the more effective choice. Therefore, the proposed GM-PHD filter should be favored in such a feature level fusion system.

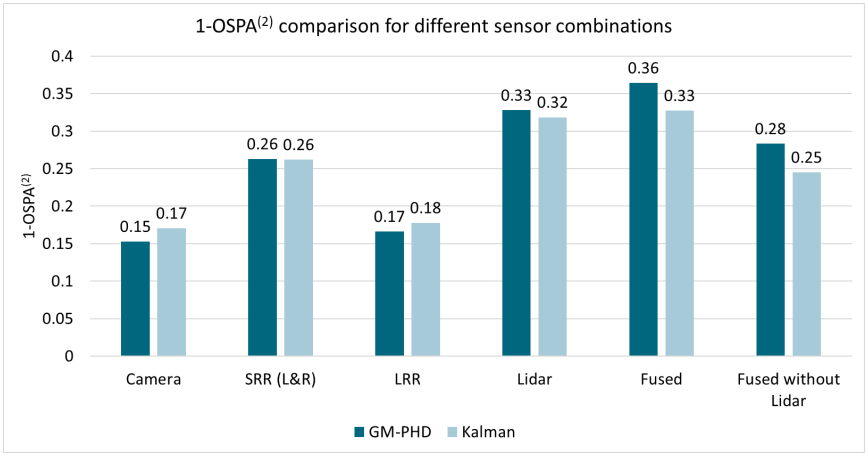


Figure 6.12: Comparison of GM-PHD and KF using the OSPA$^{(2)}$ metric on the truck dataset with different sensor combinations.

6.7.3 Evaluation of sensor-based parameter models

The sensor-based parameter models developed in chapter 5.5 are tested and compared to the standard implementation with constant values for the GM-PHD filter. In this comparison, the influence of the detection probability and clutter density models are investigated on the KITTI dataset and the detection probability model in addition on the truck dataset. Both the detection probability model and the clutter density model consist of multiple parts, which are the distance dependency, the On/Off road dependency and the operational domain dependency. In order to measure the influence of each part of the model separately, the functions are tested separately and in combination. Note, that the parameters are estimated using the same dataset as used for the test, since no additional data source is available with GT information.

First, the comparison is done using the KITTI dataset. Figure 6.13 shows the HOTA results of the constant model compared to the different parts of the detection probability model and the combination of all of them. The results are provided for the camera only, Lidar only and the fused situation. The figure makes clear, that the sensor-based parameter models can improve the results for the camera setup compared to the constant model. However, the improvements are minimal for the Lidar only and the fused situation.

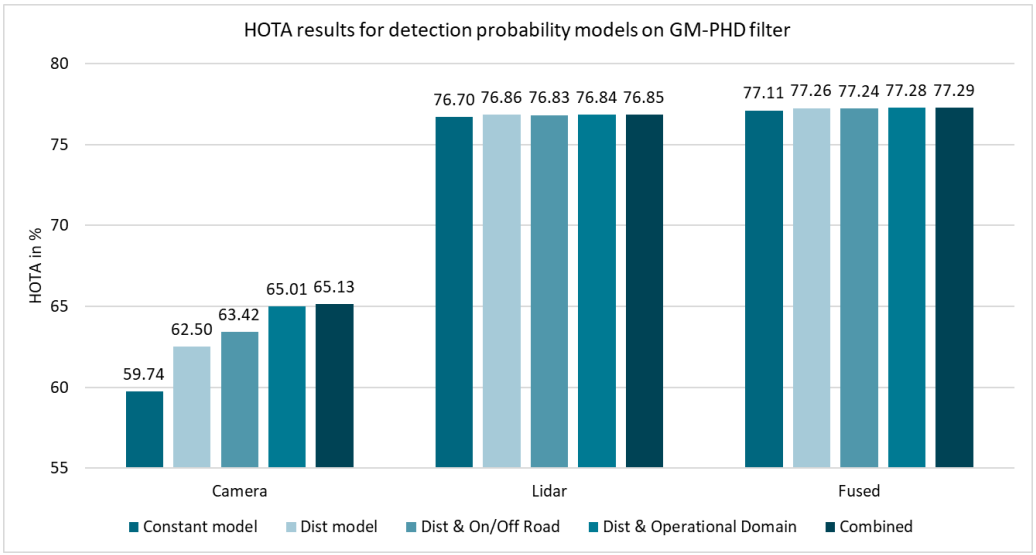


Figure 6.13: Comparison of detection probability model impact on HOTA results of the KITTI dataset.

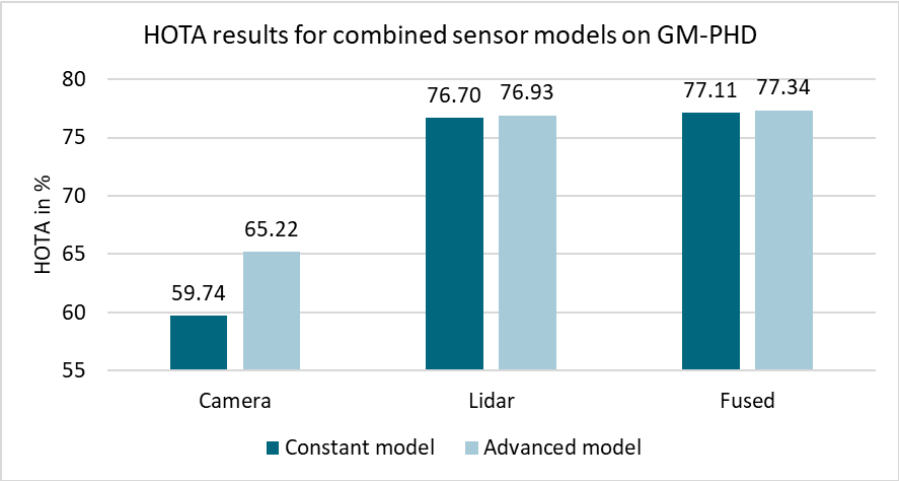


Figure 6.15: Influence of combined parameter models on HOTA results of the KITTI dataset.

A possible explanation is the good overall quality of the Lidar detections for the whole observation space, which does not need such detailed models. Since the quality of the Lidar measurements in terms of detection probability and localization accuracy is higher compared to the camera, it is also the dominant detector for the fused results. For the camera only situation, the developed models lead to bigger improvements.

A similar situation is shown for the clutter density model in Figure 6.14. The advanced clutter density models can also lead to small gains in most situations. The biggest improvement is also achieved for the camera only situations. However, the improvements are smaller compared to the detection probability model.

In Figure 6.15, a comparison of the constant model to the combination of all detection probability models and clutter density models is shown for camera only, Lidar only and fusion situation. It gets clear, that the overall combination improves the result in all situations, while the biggest improvement is achieved for the camera only situation. For the Lidar and fused results, the improvements are small in comparison.

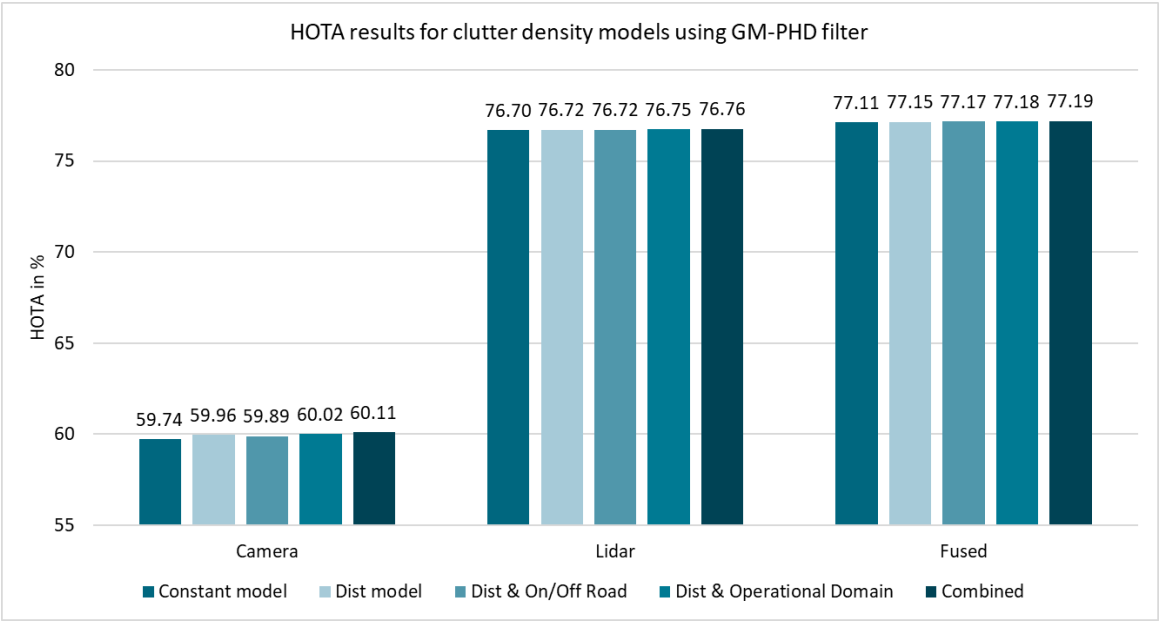


Figure 6.14: Comparison of clutter density model impact on HOTA results of the KITTI dataset.

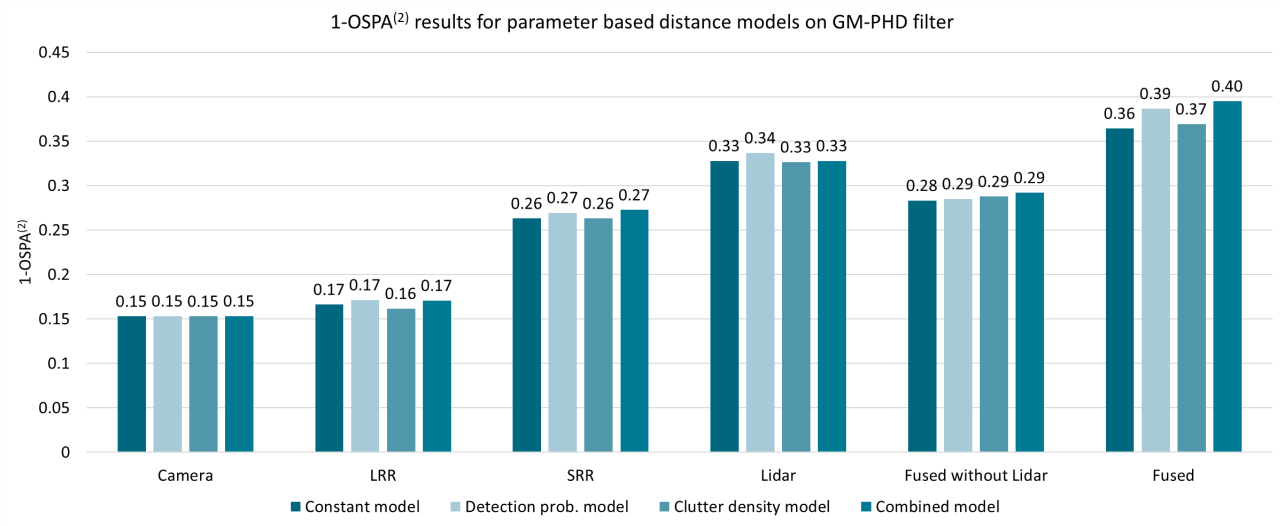


Figure 6.16: Influence of distance-depending detection probability model on the truck dataset.

The following experiments show the influence of the parameter model on the truck dataset. In contrast to the KITTI dataset, only a reduced model that purely depends on the distance and FoV is used for both detection probability and clutter density, due to the much lower amount of data. For future datasets with more data available, other parts such as the On/Off road influence can be included as well. When applying the distance model, the results increase for most of the single sensor cases and by a substantial amount for the fused case, as shown by Figure 6.16. In general, the influence of the clutter density model is small compared to the detection probability and even shows a slight negative effect on the LRR only setup. The distance-based detection probability allows a correct fusion of sensors, that share a similar radial FoV but have different capabilities regarding distance. Note, that even the constant model used here does already incorporate the FoV properties of the sensor, since this is a prerequisite for a working fusion of partially non-overlapping FoVs. Otherwise, the results would drop dramatically due to a wrong estimation of the detection probability outside the FoV of a sensor.

Overall, the sensor-based parameter models show the potential to improve tracking results, but the effectiveness highly depends on the sensor setup and parameter type. Even though the clutter density models can improve the results, their influence is small for both tested datasets. The detection probability model can achieve a higher positive influence on both datasets. For heterogeneous sensor setups of lower quality, such as used by the truck dataset, in particular the distance-depending detection model shows benefits. Here, the differences in the detection capabilities between sensors are greater and therefore need to be modeled accordingly. However, the improvement is small for situations with high-quality detectors that deliver good and homogeneous results over the whole observation space, like for the KITTI dataset. Note, that the sensor-based parameter models also have the potential to increase the accuracy of the KF based implementation in a similar manner.

6.7.4 Comparison of the proposed GM-PHD implementation to original implementation

The implementation proposed in chapter 5.7 differs from the original implementation proposed by Vo and Ma (2006) and Clark et al. (2006) at the following points: As discussed in chapter 5.5.1.1, the implementation of the detection probability uses assumptions and simplifies the process to reduce the runtime and to allow the use of sensor-based parameter models. In addition, a gating process is introduced as shown in chapter 5.7. Here, the influences of these implementation details are investigated to prove that the filter is still functional for automotive applications.

Table 6.11: Comparison of runtime and HOTA for different gating distances.

	HOTA in %	Runtime in ms
Kalman with gate $\gamma_{DA} = 5$	74.65	0.381
GM-PHD with gate $\gamma_{DA} = 5$	76.70	0.704
GM-PHD with gate $\gamma_{DA} = 15$	76.69	0.757
GM-PHD without gate	76.69	1.582

First, the proposed gating process is evaluated. Table 6.11 compares the average runtime for one prediction and update step, as well as the HOTA result for different gate distances. The Lidar only situation with constant parameter models is used for comparison. The KF is included for the runtime comparison as well. The maximum number of Gaussian components for the prune process is set to 1000 for this test. The experiments were performed using a computer with an Intel Core I7-9700K CPU. As shown by the comparison, the GM-PHD filter with a gate threshold of $\gamma_{DA} = 5$ achieves a similar HOTA score while reducing the runtime by a factor of 2. The KF with a gate of 5 needs approximately half the runtime compared to the GM-PHD filter. Note, that these values may vary with different implementations or different input data.

In addition, the proposed detection probability implementation of the distance-depending probability model is compared to the original implementation from Vo and Ma (2006), which uses a mixture model for the detection probability. For the mixture model of the detection probability, seven Gaussian components are used and a numerical optimization with squared distance is used to fit the components to the distance model. The visualization of the two models is shown in Figure 5.12 of chapter 5.5.1.2. Table 6.12 compares the results of both implementations for the Lidar only situation on the KITTI dataset. Both the combination with and without gating are tested. As shown by the table, the runtimes are massively reduced using the proposed implementation, while the HOTA results even increase.

As shown by both the experiments, the HOTA results do not suffer from the simplifications, while the gains in runtime are significant. Both measures combined can speed up the computation by a factor of approximately 20 in our exemplary situation. It is important to note, that while the simplifications seem to work well in the tested real-world scenarios, it is not given, that the implementations will work in all situations.

Table 6.12: Comparison of runtime and HOTA for exponential mixture and proposed detection probability model.

	HOTA in %	Runtime in ms
Proposed distance model with gate $\gamma_{DA} = 5$	76.86	0.715
Original mixture model with gate $\gamma_{DA} = 5$	75.12	7.313
Proposed distance model without gate	76.84	1.629
Original mixture model without gate	75.81	15.015

6.7.5 Sensor failure analysis

In real-world scenarios, sensors may become damaged or fail. In such situations, the MOT should continue to work as good as possible without adjustments. The influence of the failure of individual sensors is shown on the fusion results for the truck dataset with the GM-PHD filter, since multiple sensors have overlapping FoVs here. In this experiment, the Lidar is excluded since it is none of the current generation sensors and therefore achieves much higher accuracies. Here, the narrow evaluation area is used since sensors are directly compared to each other and this area is shared by all of them.

The result of the fusion is calculated, excluding one sensor at a time. As shown in Figure 6.17, the results of omitting single individual sensors lead to performance degradation, but the system remains operational, and the overall scores still exceed the performances of the best single sensor, which is the left SRR. Note, that these results only apply for the narrow evaluation area. This is also the reason, why the score from the Lidar missing scenario here differs from the previous evaluation in Figure 6.12 where the full evaluation area is used.

The results also indicate the influence of the different single sensors on the overall result: The influence of the SRR left is the highest, since the performance in case of deactivating it for the fusion drops the most. In contrast, the influence of the LRR and camera are lower. One reason is the geometrical model of a point and width delivered by these sensors. Therefore, they do not generate benefit to the bounding box dimension estimation, which is part of the evaluation. In addition, the strengths of the LRR in higher distances are not evaluated due to the limited distance of 70 m. The influence of the right SRR is smaller in the driving examples, since most relevant objects are occurring on the left side in right-hand traffic.

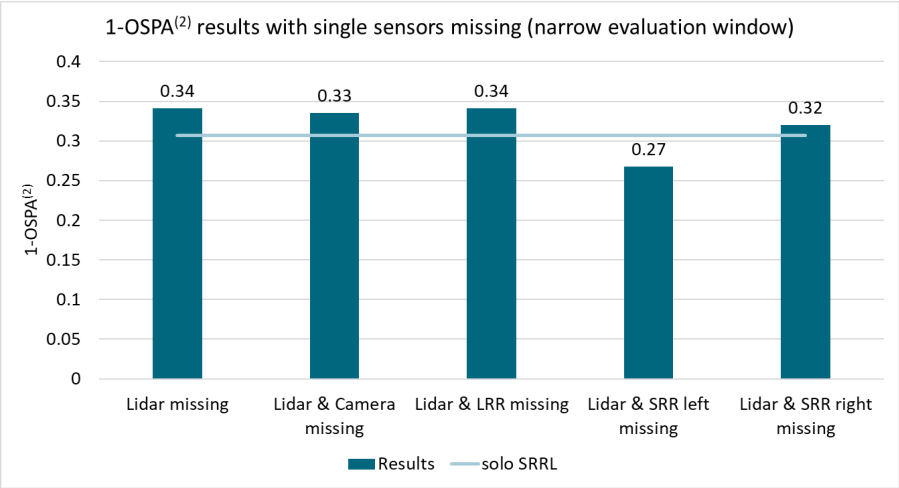


Figure 6.17: Sensor failure analysis with different sensors missing, using the narrow evaluation window on the truck dataset.

6.7.6 Comparison of performance in different operational domains

Since the truck is operating in different operational domains, they are evaluated separately with the GM-PHD filter. The dataset consists of a highway scene, an urban scene and a rural scene, which are compared here. As shown by Figure 6.18, the average OSPA$^{(2)}$ result is much better for the highway, exceeding the ones from the urban and rural scenes. This can be explained by the cleaner environment with fewer reflections that create clutter for Lidar and radar and simpler situations to learn for machine learning approaches.

This result underlines the functional safety approach of dividing autonomous driving into different levels, since even from a tracking and fusion perspective, some situations work better than others.

6.7.7 Comparison of classification approaches

The different classification approaches proposed in chapter 5.6 are compared to each other using the fusion situation of all sensors with the GM-PHD filter. The OSPA$^{(2)}$ result is shown in Figure 6.19 and clearly shows the advantage of the Dempster-Shafer classification fusion over the other approaches. Although it is a simple approach, the voting scheme achieves the second-highest result in this experiment. However, the influence of the classification method is relatively low compared to the influence of other factors, such as a missing sensor.

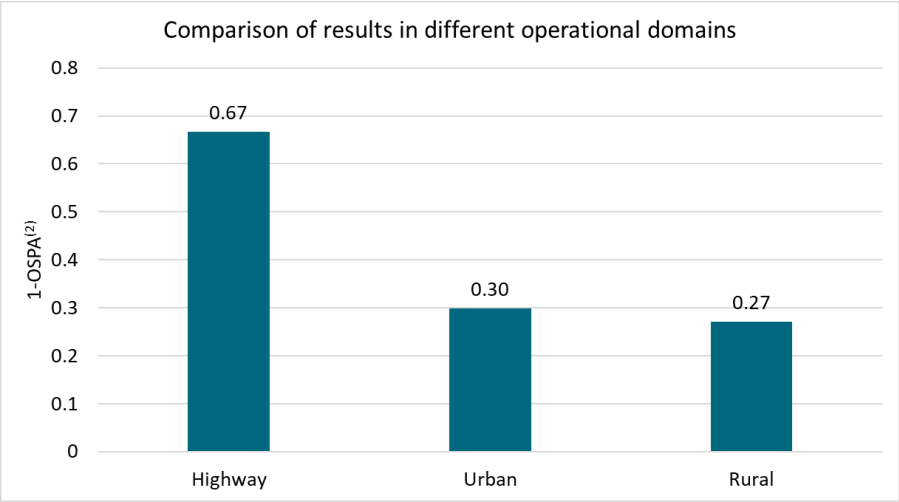


Figure 6.18: Comparison of $OSPA^{(2)}$ results in different ODDs of the truck dataset.

Overall, the Dempster-Shafer classification fusion should be used when maximum accuracy is desired, while the Voting-Scheme should be used when focusing on a low complexity.

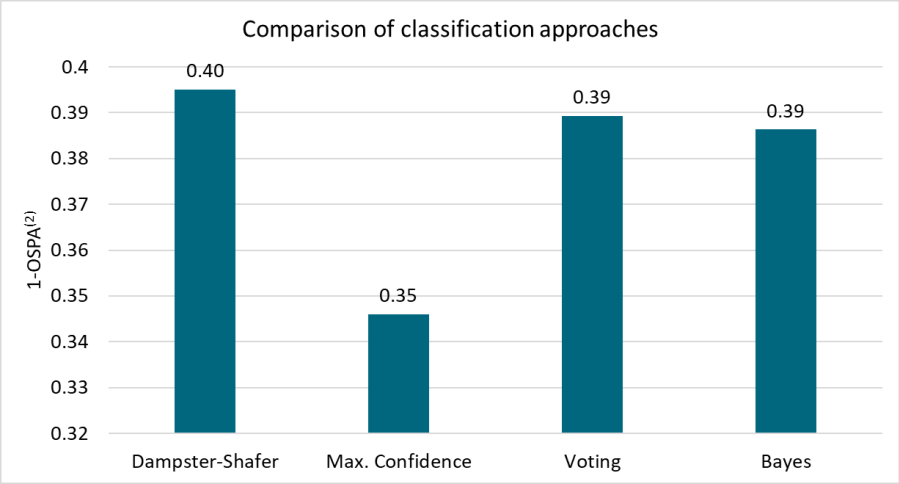


Figure 6.19: Comparison of different classification approaches on Fused Truck dataset with GM-PHD filter.

7 Conclusion and future work

The field of environment perception is subject to constant change due to ongoing research, which is driven in particular by modern developments in the field of neural networks. In the context of this thesis, a wide variety of detection, tracking, and fusion approaches for different applications have been discussed, developed, implemented and compared. Overall, a full framework for multi-sensor multi-object detection and tracking was proposed, that offers compatibility with different current and next generation sensor setups with different object detection approaches and supports both the KF and GM-PHD filter, as well as multiple classification fusion schemes. This allows a comparison of the KF and GM-PHD in the same framework on real-world data provided by the KITTI dataset and a newly recorded truck dataset, which was not possible previously. In addition, several improvements for the GM-PHD filter are developed and tested that enable it's usage in real-world automotive applications.

7.1 Conclusion

The methods developed can be used to improve and expand current-generation systems. In particular, the following conclusions can be drawn with regard to ADAS systems. Note that these conclusions only apply to the situations and experiments carried out in this thesis:

- Lidar-based detection methods can improve environment perception. The developed two-stage approach provides a safe and accurate approach to object detection, achieving an mAP of 79.33 % on the KITTI 3D detection benchmark. The unclassified recall of 96.53% shows that a high percentage of objects can be correctly detected even if the NN fails to classify it correctly, which leads to advantages in terms of safety.
- Monocular cameras in combination with neural networks and projection enable accurate object detection at close range for downward looking applications for trucks despite the cabin movement, achieving an mAP of 95.4 % using 2D image coordinates and a mAP of 45.28% using 3D OBBs. The average distance error of 0.185 m shows that the projection method offers good accuracy. Overall, this approach creates a new possibility for detecting objects at close range for trucks.
- The developed truck dataset with current generation of sensors enables situation analysis and evaluation of detection and tracking algorithms for trucks in real-world scenarios.
- The developed framework for multi-sensor multi-object tracking is a universal, sensor-independent approach that enables the use of different multi-object filters and classification fusion approaches. The framework works with both Kalman and GM-PHD filter implementations and is tested using both the KITTI dataset and the truck dataset. Most importantly, it enables a direct comparison of the filters on real-world data.
- The GM-PHD filter can outperform the Kalman filter in most situations on both the KITTI dataset and the truck dataset, but in particular in fusion situations: The GM-PHD filter achieves a HOTA score of 77.11 % compared to the 73.56% of the KF on the KITTI dataset. On the truck dataset, the GM-PHD filter achieves a 1-OSPA$^{(2)}$ score of 0.36 compared to 0.33 of the KF for fused data. The KF can only achieve better results for scenarios with single sensors and few detections, such as camera only.
- The sensor-based parameter models provide an improvement of the result in most situations for the GM-PHD filter on the KITTI and the truck dataset. For good and homogeneous detection situations, the improvements are only marginal, while for more heterogeneous setups, like on the truck dataset, the

improvements can be higher. In the fusion situation, a 1-OSPA$^{(2)}$ result of 0.40 compared to 0.36 with a constant model is achieved on the truck dataset.

- The developed implementation of the GM-PHD filter that includes gating and the proposed sensor-based parameter models can reduce the runtime significantly. Compared to a reference implementation without gating and with a mixture model implementation for the detection probability, the runtime is reduced by a factor of more than 20 while slightly increasing the HOTA score.
- The developed GM-PHD implementation uses tags in combination with a track extraction scheme to preserve the track's unique IDs, which enables the usage of the GM-PHD filter in real-world applications for tracking.
- Using the developed spatial measurement matching algorithm, a MOT with detections of different geometric objects, like OBBs, point objects or L-shapes, can be realized.
- The GM-PHD-based implementation remains operable even when individual sensors fail. For each single sensor failure, the fusion result of the remaining sensors is better than the best single sensor.
- The MOT results on highways are higher compared to rural and urban scenarios. In this scenario, less clutter affects the tracking.
- The proposed integration of the classification to the GM-PHD is a new and elegant way of propagating this property within a MOT framework, and is necessary for real-world applications.
- The Dempster-Shafer fusion of classification achieves the best MOT results with a 1-OSPA$^{(2)}$ score of 0.40 compared to 0.39 of the voting scheme, which achieves the second-highest score but is much simpler.

Overall, the proposed multi-sensor multi-object framework with the GM-PHD filter using the proposed adaptations and the Dempster-Shafer classification fusion can outperform current KF-based frameworks for truck applications. Such a framework can therefore be recommended in particular for heterogeneous setups with different sensors that have different capabilities. In addition, the proposed framework is completely versatile usable, which is in particular interesting for truck applications, since damaged sensors or different sensor setups and mounting positions can be used with the same software stack.

7.2 Future work

The truck dataset used in this thesis covers three scenarios of limited length. In future analysis, a larger truck dataset with a bigger variety of operational domains, weather, illumination, and places should be used to verify the results. In addition, this can help to improve the filter parameters, the detection results and enable the development of more complex parameter models.

The current GM-PHD implementation does ignore the spawn of targets in the prediction step. A spawn model can be developed for the filter to model spawn events that naturally happen in real-world scenarios, such as a driver getting out of a car, and to resolve under-segmented objects.

The current implementation of the GM-PHD filter uses the mechanism of label tags to propagate the track ID. However, other developments in the field of labeled RFS filters, like the Generalized Labeled Multi Bernoulli Filter (Vo and Vo 2013) implicitly handle this issue. Therefore, such a filter could be included in the comparison in addition.

In general, environmental detection, and therefore also MOT, will continue to be the subject of research and development in the coming years, as better and better ADAS and AD systems are pursued. The trend here shows an ever-increasing proportion of NNs, which currently mainly affects detection. At some point, parts of tracking could also be taken over by NN, which should then be compared with the methods presented here.

8 Acronyms

ACC	Advanced Cruise Control
AD	Autonomous Driving
ADAS	Advanced Driver Assistance Systems
AEBS	Advanced Emergency Braking System
AMCW	Amplitude Modulated Continuous Wave
AP	Average Precision
BEV	Bird's Eye View
BSIS	Blind Spot Information System
CA	Constant Acceleration
CBMeMBer	Cardinality Balanced Multi Bernoulli Filter
CNN	Convolutional Neural Network
CPHD	Cardinalized Probability Hypothesis Density
CTRA	Constant Turn Rate Acceleration
DA	Data Association
DST	Dempster-Shafer Theory
EKF	Extended Kalman Filter
FISST	Finite-Set Statistics
FMCW	Frequency Modulated Continuous Wave
FN	False Negative
FNA	False Negative Association
FoV	Field of View
FP	False Positive
FPA	False Positive Association
GIoU	Generalized Intersection over Union
GM-PHD	Gaussian Mixture Probability Hypothesis Density
GNNDA	Global Nearest Neighbor Data Association
GNSS	Global Navigation Satellite System
GSR	General Safety Regulation
GT	Ground Truth
HOTA	Higher Order Tracking Accuracy
ICP	Iterative Closest Point
IID	Independent Identically Distributed
IMM	Interacting Multiple Model
IMU	Inertial Measurement Units
IoU	Intersection over Union
JPDAF	Joint Probabilistic Data Association Filter
KF	Kalman Filter
KLD	Kullback-Leiber Divergence
LRR	Long Range Radar
LSTM	Long Short-Term Memory
mAP	mean Average Precision
MEMS	Micro-Electromechanical Systems
MHT	Multi Hypothesis Tracker
MLP	Multi Layer Perceptrons
MOIS	Moving Off Information System
MOT	Multi-Object Tracking

MOTA	Multiple Object Tracking Accuracy
MRR	Medium Range Radar
MVFE	Modified Voxel Feature Encoder
NN	Neural Network
NNDA	Nearest Neighbor Data Association
OBB	Oriented Bounding Box
ODD	Operational Design Domains
OMAT	Optimal Mass Transfer
OSPA	Optimal Sub-Pattern Assignment
OSPA$^{(2)}$	Second Order Optimal Sub-Pattern Assignment
PDAF	Probabilistic Data Association Filter
PDF	Probability Density Function
PHD	Probability Hypothesis Density
RANSAC	Random Sample Consensus
ReLU	Rectified Linear Unit
RFS	Random Finite Set
SAE	Society of Automotive Engineers
SMC	Sequential Monte Carlo
SRR	Short Range Radar
STC	Stanford Track Collection
SVD	Singular Value Decomposition
SVM	Support Vector Machine
ToF	Time of Flight
TP	True Positive
TPA	True Positive Association
UKF	Unscented Kalman Filter
VCH	Visible Convex Hull
VRU	Vulnerable Road User

Bibliography

Aeberhard, M. (2017): Object-Level Fusion for Surround Environment Perception in Automated Driving Applications. Dissertation at Technical University of Dortmund. Faculty of Electrical Engineering and Information Technology, Dortmund.

Aeberhard, M., Paul, S., Kaempchen, N., Bertram, T. (2011): Object existence probability fusion using dempster-shafer theory in a high-level sensor data fusion architecture. In: *2011 IEEE Intelligent Vehicles Symposium (IV)* IEEE, pp. 770–775. ISBN: 978-1-4577-0890-9. DOI: 10.1109/IVS.2011.5940430.

Aggarwal, C. C. (2018): Neural Networks and Deep Learning. Springer International Publishing, Cham.

Amit, Y., Felzenszwalb, P., Girshick, R. (2020): Object Detection. In: *Computer Vision: A Reference Guide*. Ikeuchi, K. [Ed.] Cham: Springer International Publishing, pp. 1–9. ISBN: 978-3-030-03243-2. DOI: 10.1007/978-3-030-03243-2_660-1.

Anderson, B. D. O., Moore, J. B. (1979): Optimal filtering. Prentice-Hall, Englewood Cliffs, N. J. *Prentice-Hall information and system sciences series*.

Arnold, E., Jarrah, O. Y. A., Dianati, M., Fallah, S., Oxtoby, D., Mouzakitis, A. (2019): A Survey on 3D Object Detection Methods for Autonomous Driving Applications. *IEEE Transactions on Intelligent Transportation Systems*, 20, 10, pp. 3782–3795, DOI: 10.1109/TITS.2019.2892405. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8621614.

Arras, K. O., Mozos, O. M., Burgard, W. (2007): Using Boosted Features for the Detection of People in 2D Range Data. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation* pp. 3402–3407. ISBN: 1-4244-0601-3. DOI: 10.1109/ROBOT.2007.363998. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4209616.

Asvadi, A., Premebida, C., Peixoto, P., Nunes, U. (2016): 3D Lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes. *Robotics and Autonomous Systems*, 83, pp. 299–311, DOI: 10.1016/j.robot.2016.06.007. URL: https://www.sciencedirect.com/science/article/pii/S0921889016300483.

Bader, C. (2019): System for Detection, Tracking and Classification of multiple Objects for Collision Detection based on LiDAR Sensors. Masterthesis at FH Joanneum Graz. Graz.

Bader, C., Dingler, S., Schwieger, V., eds. (2021): PVENet: Point Voxel Encoder Network for Real-Time Classification of Lidar Point Cloud Segments

Bader, C., Schwieger, V. (2024): Advancing ADAS Perception: A Sensor-Parameterized Implementation of the GM-PHD Filter. *Sensors*, 24, 8. DOI: 10.18419/opus-14662. URL: https://www.mdpi.com/1424-8220/24/8/2436.

Bar-Shalom, Y. (2002): Update with out-of-sequence measurements in tracking: exact solution. *IEEE Transactions on Aerospace and Electronic Systems*, 38, 3, pp. 769–777, DOI: 10.1109/TAES.2002.1039398.

Bar-Shalom, Y., Campo, L. (1986): The Effect of the Common Process Noise on the Two-Sensor Fused-Track Covariance. *IEEE Transactions on Aerospace and Electronic Systems*, AES-22, 6, pp. 803–805, DOI: 10.1109/TAES.1986.310815.

Bar-Shalom, Y., Li, X.-R. (1995): Multitarget-multisensor tracking: Principles and techniques. 3rd printing, YBS, Storrs, Conn.

Bar-Shalom, Y., Li, X.-R., Kirubarajan, T. (2001): Estimation with applications to tracking and navigation. 1st ed., Wiley-Interscience, New York. URL: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10299366.

Barrera, A., Guindel, C., Beltrán, J., García, F. (2020): BirdNet+: End-to-End 3D Object Detection in LiDAR Bird's Eye View. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)* pp. 1–6. ISBN: 978-1-7281-4150-3. DOI: 10.1109/ITSC45102.2020.9294293. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9294293.

Bayes, T. (1763): LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, F. R. S. communicated by Mr. Price, in a letter to John Canton, A. M. F. R. S. *Philosophical Transactions of the Royal Society of London*, 53, pp. 370–418, DOI: 10.1098/rstl.1763.0053.

Beard, M., Vo, B. T., Vo, B. N. (2020): A Solution for Large-Scale Multi-Object Tracking. *IEEE Transactions on Signal Processing*, 68, pp. 2754–2769, DOI: 10.1109/TSP.2020.2986136. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9063553.

Becker, J. C. (1999): Fusion of data from the object-detecting sensors of an autonomous vehicle. In: *Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (Cat. No.99TH8383)* pp. 362–367. ISBN: 0-7803-4975-X. DOI: 10.1109/ITSC.1999.821082. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=821082.

Beltrán, J., Guindel, C., Moreno, F. M., Cruzado, D., García, F., La Escalera, A. D. (2018): BirdNet: A 3D Object Detection Framework from LiDAR Information. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* pp. 3517–3523. ISBN: 978-1-7281-0324-2. DOI: 10.1109/ITSC.2018.8569311. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8569311.

Bernardin, K., Stiefelhagen, R. (2008): Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *EURASIP Journal on Image and Video Processing*, 2008, 1, p. 246309, DOI: 10.1155/2008/246309.

Blackman, S. S. (2004): Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19, 1, pp. 5–18, DOI: 10.1109/MAES.2004.1263228. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1263228.

Bogoslavskyi, I., Stachniss, C. (2017): Efficient Online Segmentation for Sparse 3D Laser Scans. *PFG@ Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 85, pp. 41–52.

Brock, A., Lim, T., Ritchie, J. M., Weston, N. (2016): Generative and Discriminative Voxel Modeling with Convolutional Neural Networks. *CoRR*, abs/1608.04236.

Burger, P., Wuensche, H. J. (2018): Fast Multi-Pass 3D Point Segmentation Based on a Structured Mesh Graph for Ground Vehicles. In: *2018 IEEE Intelligent Vehicles Symposium (IV)* pp. 2150–2156. ISBN: 978-1-5386-4453-9. DOI: 10.1109/IVS.2018.8500552. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8500552.

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O. (2019): nuScenes: A multimodal dataset for autonomous driving. DOI: 10.48550/arXiv.1903.11027.

Cao, Y., Wang, Y., Xue, Y., Zhang, H., Lao, Y. (2022): FEC: Fast Euclidean Clustering for Point Cloud Segmentation. URL: http://arxiv.org/pdf/2208.07678v1.

Chabot, F., Chaouch, M., Rabarisoa, J., Teulière, C., Chateau, T. (2017): Deep MANTA: A Coarse-to-Fine Many-Task Network for Joint 2D and 3D Vehicle Analysis from Monocular Image. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 1827–1836. ISBN: 978-1-5386-0458-8. DOI: 10.1109/CVPR.2017.198. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8099681.

Challa, S., Koks, D. (2004): Bayesian and Dempster-Shafer fusion. *Sadhana*, 29, 2, pp. 145–174, DOI: 10.1007/BF02703729.

Challa, S., Morelande, M. R., Musicki, D., Evans, R. J. (2011): Fundamentals of object tracking. 1. publ, Cambridge University Press, Cambridge. URL: http://www.loc.gov/catdir/enhancements/fy1209/2011008595-b.html.

Charles, R. Q., Su, H., Kaichun, M., Guibas, L. J. (2017a): PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 77–85. ISBN: 978-1-5386-0458-8. DOI: 10.1109/CVPR.2017.16. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8099499.

Charles, R. Q., Yi, L., Su, H., Guibas, L. J. (2017b): PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., Garnett, R. [Ed.] pp. 5099–5108.

Che, E., Jung, J., Olsen, M. J. (2019): Object Recognition, Segmentation, and Classification of Mobile Laser Scanning Point Clouds: A State of the Art Review. *Sensors (Basel, Switzerland)*, 19, 4. DOI: 10.3390/s19040810.

Chen, T., Dai, B., Liu, D., Song, J. (2014): Performance of global descriptors for velodyne-based urban object recognition. In: *2014 IEEE Intelligent Vehicles Symposium Proceedings* pp. 667–673. ISBN: 978-1-4799-3638-0. DOI: 10.1109/IVS.2014.6856425. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6856425.

Chen, X., Li, Y., Li, Y., Yu, J. (2018): PHD and CPHD Algorithms Based on a Novel Detection Probability Applied in an Active Sonar Tracking System. *Applied Sciences*, 8, 1, p. 36, DOI: 10.3390/app8010036.

Chen, X., Ma, H., Wan, J., Li, B., Xia, T. (2017): Multi-view 3D Object Detection Network for Autonomous Driving. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 6526–6534. ISBN: 978-1-5386-0458-8. DOI: 10.1109/CVPR.2017.691. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8100174.

Chen, Y., Liu, S., Shen, X., Jia, J. (2019): Fast Point R-CNN. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* pp. 9774–9783. ISBN: 978-1-7281-4804-5. DOI: 10.1109/ICCV.2019.00987. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9008235.

Chiu, H.-k., Prioletti, A., Li, J., Bohg, J. (2020): Probabilistic 3D Multi-Object Tracking for Autonomous Driving. DOI: 10.48550/arXiv.2001.05673.

Choi, S., Park, J., Byun, J., Yu, W. (2014): Robust ground plane detection from 3D point clouds. In: *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)* pp. 1076–1081. ISBN: 978-8-9932-1506-9. DOI: 10.1109/ICCAS.2014.6987936. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6987936.

Chu, P. M., Cho, S., Sim, S., Kwak, K. H., Cho, K. (2017): A Fast Ground Segmentation Method for 3D Point Cloud. *J. Inf. Process. Syst.*, 13, pp. 491–499.

Clark, D., Panta, K., Vo, B.-n. (2006): The GM-PHD Filter Multiple Target Tracker. In: *2006 9th International Conference on Information Fusion* IEEE, pp. 1–8. ISBN: 1-4244-0953-5. DOI: 10.1109/ICIF.2006.301809.

Cohen, S., Guibas, L. (1997): Partial Matching of Planar Polylines Under Similarity Transformations. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*. DOI: 10.1145/314161.314445.

Daimler Truck (2023): Daimler Truck Products. URL: https://www.daimlertruck.com/en/products. visited on 11/03/2023.

Dalal, N., Triggs, B. (2005): Histograms of oriented gradients for human detection. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1 pp. 886–893. ISBN: 0-7695-2372-2. DOI: 10.1109/CVPR.2005.177. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1467360.

Darms, M., Winner, H. (2005): A modular system architecture for sensor data processing of ADAS applications. In: *IEEE Proceedings. Intelligent Vehicles Symposium, 2005* pp. 729–734. ISBN: 0-7803-8961-1. DOI: 10.1109/IVS.2005.1505190. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1505190.

Darms, M. S., Rybski, P. E., Baker, C., Urmson, C. (2009): Obstacle Detection and Tracking for the Urban Challenge. *IEEE Transactions on Intelligent Transportation Systems*, 10, 3, pp. 475–485, DOI: 10.1109/TITS.2009.2018319.

Dempster, A. P. (1967): Upper and Lower Probabilities Induced by a Multivalued Mapping. *The Annals of Mathematical Statistics*, 38, 2, pp. 325–339, DOI: 10.1214/aoms/1177698950.

Dietmayer, K. (2016): Predicting of Machine Perception for Automated Driving. In: *Autonomous Driving*. Maurer, M., Gerdes, J. C., Lenz, B., Winner, H. [Ed.] Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 407–424. ISBN: 978-3-662-48845-4. DOI: 10.1007/978-3-662-48847-8_20.

Douillard, B., Underwood, J., Kuntz, N., Vlaskine, V., Quadros, A., Morton, P., Frenkel, A. (2011): On the segmentation of 3D LIDAR point clouds. In: *2011 IEEE International Conference on Robotics and Automation* pp. 2798–2805. ISBN: 978-1-61284-380-3. DOI: 10.1109/ICRA.2011.5979818. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5979818.

Elfes, A. (1989): Using occupancy grids for mobile robot perception and navigation. *Computer*, 22, 6, pp. 46–57, DOI: 10.1109/2.30720. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=30720.

European Commission (2020): Sustainable and Smart Mobility Strategy – putting European transport on track for the future. Brussels. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52020DC0789. visited on 06/02/2022.

European Commission and Directorate-General for Mobility and Transport (2020): Next steps towards 'Vision Zero' – EU road safety policy framework 2021-2030. DOI: doi/10.2832/391271.

European Parliament and Council (2019): Regulation (EU) 2019/2144 of the European Parliament and of the Council of 27 November 2019 on type-approval requirements for motor vehicles and their trailers, and systems, components and separate technical units intended for such vehicles, as regards their general safety and the protection of vehicle occupants and vulnerable road users, amending Regulation (EU) 2018/858 of the European Parliament and of the Council and repealing Regulations (EC) No 78/2009, (EC) No 79/2009 and (EC) No 661/2009 of the European Parliament and of the Council and Commission Regulations (EC) No 631/2009, (EU) No 406/2010, (EU) No 672/2010, (EU) No 1003/2010, (EU) No 1005/2010, (EU) No 1008/2010, (EU) No 1009/2010, (EU) No 19/2011, (EU) No 109/2011, (EU) No 458/2011, (EU) No 65/2012, (EU) No 130/2012, (EU) No 347/2012, (EU) No 351/2012, (EU) No 1230/2012 and (EU) 2015/166. URL: https://eur-lex.europa.eu/eli/reg/2019/2144/oj.

Everingham, M., van Gool, L., Williams, Christopher K. I., Winn, J., Zisserman, A. (2010): The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88, 2, pp. 303–338, DOI: 10.1007/s11263-009-0275-4.

Felzenszwalb, P., McAllester, D., Ramanan, D. (2008): A discriminatively trained, multiscale, deformable part model. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition* pp. 1–8. ISBN: 978-1-4244-

2243-2. DOI: 10.1109/CVPR.2008.4587597. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4587597.

Fernandes, D., Silva, A., Névoa, R., Simões, C., Gonzalez, D. G., Guevara, M., Novais, P., Monteiro, J. L., Melo-Pinto, P. (2021): Point-cloud based 3D object detection and classification methods for self-driving applications: A survey and taxonomy. *Inf. Fusion*, 68, pp. 161–191.

Floudas, N., Polychronopoulos, A., Aycard, O., Burlet, J., Ahrholdt, M. (2007): High Level Sensor Data Fusion Approaches For Object Recognition In Road Environment. In: *2007 IEEE Intelligent Vehicles Symposium* IEEE, pp. 136–141. ISBN: 1-4244-1067-3. DOI: 10.1109/IVS.2007.4290104.

Franken, D., Schmidt, M., Ulmke, M. (2009): "Spooky Action at a Distance" in the Cardinalized Probability Hypothesis Density Filter. *IEEE Transactions on Aerospace and Electronic Systems*, 45, 4, pp. 1657–1664, DOI: 10.1109/TAES.2009.5310327.

Gamba, J. (2020): Radar Signal Processing for Autonomous Driving. 1st ed. 2020, Springer, Singapore. *Springer eBooks Engineering*.

Geiger, A., Lenz, P., Urtasun, R. (2012): Are we ready for autonomous driving? The KITTI vision benchmark suite. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition* pp. 3354–3361. ISBN: 978-1-4673-1227-1. DOI: 10.1109/CVPR.2012.6248074. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6248074.

Genovese, A. F. (2001): The interacting multiple model algorithm for accurate state estimation of maneuvering targets. *Johns Hopkins APL Technical Digest (Applied Physics Laboratory)*, 22, pp. 614–623.

Girshick, R. (2015): Fast R-CNN. In: *2015 IEEE International Conference on Computer Vision (ICCV)* pp. 1440–1448. ISBN: 978-1-4673-8390-5. DOI: 10.1109/ICCV.2015.169. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7410526.

Girshick, R., Donahue, J., Darrell, T., Malik, J. (2014): Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition* pp. 580–587. ISBN: 978-1-4799-5118-5. DOI: 10.1109/CVPR.2014.81. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6909475.

Graham, R. L. (1972): An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. *Inf. Process. Lett.*, 1, pp. 132–133.

Granström, K. (2012): Extended target tracking using PHD filters. Dissertation at Department of Electrical Engineering. Linköping University, Linköping, Sweden.

Granström, K., Baum, M., Reuter, S. (2017): Extended Object Tracking: Introduction, Overview, and Applications. *Journal of Advances in Information Fusion*, 12.

Granström, K., Lundquist, C., Orguner, U. (2011): Tracking rectangular and elliptical extended targets using laser measurements. In: *14th International Conference on Information Fusion* pp. 1–8. ISBN: 978-0-9824438-3-5. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5977698.

Granström, K., Orguner, U. (2012): On the reduction of Gaussian inverse Wishart mixtures. In: *2012 15th International Conference on Information Fusion* pp. 2162–2169. ISBN: 978-0-9824438-4-2. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6290566.

Granström, K., Reuter, S., Meissner, D., Scheel, A. (2014): A multiple model PHD approach to tracking of cars under an assumed rectangular shape. In: *17th International Conference on Information Fusion (FUSION)* pp. 1–8.

Grewe, R., Hohm, A., Hegemann, S., Lueke, S., Winner, H. (2012): Towards a generic and efficient environment model for ADAS. In: *2012 IEEE Intelligent Vehicles Symposium* pp. 316–321. ISBN: 978-1-4673-2117-4. DOI: 10.1109/IVS.2012.6232146. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6232146.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., Bennamoun, M. (2020): Deep Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP, 99, p. 1, DOI: 10.1109/TPAMI.2020.3005434. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9127813.

Haas, R. E., Bhattacharjee, S., Möller, D. P. F. (2020): Advanced Driver Assistance Systems. In: *Smart Technologies*. Akhilesh, K. B. [Ed.] Singapore: Springer Singapore Pte. Limited. ISBN: 978-981-13-7138-7. DOI: 10.1007/978-981-13-7139-4_27.

Hamzah, R. A., Ibrahim, H. (2016): Literature Survey on Stereo Vision Disparity Map Algorithms. *Journal of Sensors*, 2016, pp. 1–23, DOI: 10.1155/2016/8742920.

Hasch, J., Topak, E., Schnabel, R., Zwick, T., Weigel, R., Waldschmidt, C. (2012): Millimeter-Wave Technology for Automotive Radar Sensors in the 77 GHz Frequency Band. *IEEE Transactions on Microwave Theory and Techniques*, 60, 3, pp. 845–860, DOI: 10.1109/TMTT.2011.2178427. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6127923.

He, T., Soatto, S. (2019): Mono3D++: Monocular 3D Vehicle Detection with Two-Scale 3D Hypotheses and Task Priors. URL: http://arxiv.org/pdf/1901.03446v1.

Hendeby, G., Karlsson, R. (2014): Gaussian mixture PHD filtering with variable probability of detection. In: *17th International Conference on Information Fusion (FUSION)* pp. 1–7.

Herbert, M., Caillas, C., Krotkov, E., Kweon, I. S., Kanade, T. (1989): Terrain mapping for a roving planetary explorer. In: *Proceedings, 1989 International Conference on Robotics and Automation* pp. 997–1002. ISBN: 0-8186-1938-4. DOI: 10.1109/ROBOT.1989.100111. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=100111.

Herpel, T., Lauer, C., German, R., Salzberger, J. (2008): Multi-sensor data fusion in automotive applications. In: *2008 3rd International Conference on Sensing Technology* pp. 206–211. ISBN: 978-1-4244-2177-0. DOI: 10.1109/ICSENST.2008.4757100. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4757100.

Herrmann, V. (2019): Objekterkennung und Klassifizierung im Urbanen Umfeld auf Basis von LiDAR-Sensoren. Masterarbeit at University of Stuttgart. Institut für Verbrennungsmotoren und Kraftfahrwesen, Stuttgart.

Himmelsbach, M., Hundelshausen, F. v., Wuensche, H. J. (2010): Fast segmentation of 3D point clouds for ground vehicles. In: *2010 IEEE Intelligent Vehicles Symposium* pp. 560–565. ISBN: 978-1-4244-7867-5. DOI: 10.1109/IVS.2010.5548059. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5548059.

Himmelsbach, M., Luettel, T., Wuensche, H. J. (2009): Real-time object classification in 3D point clouds using point feature histograms. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems* pp. 994–1000. ISBN: 978-1-4244-3804-4. DOI: 10.1109/IROS.2009.5354493. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5354493.

Hoffman, J. R., Mahler, R. P. S. (2004): Multitarget miss distance via optimal assignment. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, 34, 3, pp. 327–336, DOI: 10.1109/TSMCA.2004.824848. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1288344.

Holz, C. (2023): Entwicklung eines Verfahrens zur Multi Objekt Verfolgung auf Basis von maschinellem Lernen (Multi Object Tracking using Machine Learning). Masterarbeit at Duale Hochschule Baden-Würtemberg. Stuttgart.

Houenou, A., Bonnifait, P., Cherfaoui, V., Yao, W. (2013): Vehicle trajectory prediction based on motion model and maneuver recognition. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* pp. 4363–4369. ISBN: 978-1-4673-6357-0. DOI: 10.1109/IROS.2013.6696982. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6696982.

Houssineau, J., Laneuville, D. (2010): PHD filter with diffuse spatial prior on the birth process with applications to GM-PHD filter. In: *2010 13th International Conference on Information Fusion* pp. 1–8. ISBN: 978-0-9824438-1-1. DOI: 10.1109/ICIF.2010.5711951. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5711951.

Jocher, G. et al. (2022): ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation. DOI: 10.5281/zenodo.3908559.

Julier, S. J. (2002): The scaled unscented transformation. In: *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*. Vol. 6 pp. 4555–4559. ISBN: 0-7803-7298-0. DOI: 10.1109/ACC.2002.1025369. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1025369.

Julier, S. J., Uhlmann, J. K. (1997): New extension of the Kalman filter to nonlinear systems. In: *Signal Processing, Sensor Fusion, and Target Recognition VI*. Kadar, I. [Ed.]. Vol. 3068 International Society for Optics and Photonics. SPIE, pp. 182 –193. DOI: 10.1117/12.280797.

Kalman, R. E. (1960): A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82, 1, pp. 35–45, DOI: 10.1115/1.3662552.

Kampker, A., Sefati, M., Rachman, A. S. A., Kreisköther, K. D., Campoy, P. (2018): Towards Multi-Object Detection and Tracking in Urban Scenario under Uncertainties. In: *Proceedings of the 4th International Conference on Vehicle Technology and Intelligent Transport Systems - VEHITS* pp. 156–167. DOI: 10.5220/0006706101560167.

Kanezaki, A., Matsushita, Y., Nishida, Y. (2018): RotationNet: Joint Object Categorization and Pose Estimation Using Multiviews from Unsupervised Viewpoints. In: *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*

Kim, S.-h., Hwang, Y. (2021): A Survey on Deep Learning Based Methods and Datasets for Monocular 3D Object Detection. *Electronics*, 10, 4. DOI: 10.3390/electronics10040517. URL: https://www.mdpi.com/2079-9292/10/4/517.

Klasing, K., Wollherr, D., Buss, M. (2008): A clustering method for efficient segmentation of 3D laser data. In: *2008 IEEE International Conference on Robotics and Automation* pp. 4043–4048. ISBN: 978-1-4244-1647-9. DOI: 10.1109/ROBOT.2008.4543832. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4543832.

Klasing, K., Wollherr, D., Buss, M. (2009): Realtime segmentation of range data using continuous nearest neighbors. In: *2009 IEEE International Conference on Robotics and Automation* pp. 2431–2436. ISBN: 978-1-4244-2789-5. DOI: 10.1109/ROBOT.2009.5152498. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5152498.

Koks, D., Challa, S. (2003): An Introduction to Bayesian and Dempster-Shafer Data Fusion. URL: https://robotics.caltech.edu/~jerma/research_papers/BayesChapmanKolmogorov.pdf.

Konstantinova, P., Udvarev, A., Semerdjiev, T. (2003): A study of a target tracking algorithm using global nearest neighbor approach. In: *Proceedings of the 4th international conference conference on Computer systems and technologies e-Learning - CompSysTech '03*. Rachev, B., Smrikarov, A. [Ed.] New York, New York, USA: ACM Press, pp. 290–295. ISBN: 9549641333. DOI: 10.1145/973620.973668.

Kraemer, S., Stiller, C., Bouzouraa, M. E. (2018): LiDAR-Based Object Tracking and Shape Estimation Using Polylines and Free-Space Information. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* pp. 4515–4522. ISBN: 978-1-5386-8095-7. DOI: 10.1109/IROS.2018.8593385. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8593385.

Kuang, H., Wang, B., An, J., Zhang, M., Zhang, Z. (2020): Voxel-FPN: Multi-Scale Voxel Feature Aggregation for 3D Object Detection from LIDAR Point Clouds. *Sensors*, 20, 3. DOI: 10.3390/s20030704. URL: https://www.mdpi.com/1424-8220/20/3/704.

Kuhn, H. W. (1955): The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2, 1-2, pp. 83–97, DOI: 10.1002/nav.3800020109.

Kämpchen, N. (2007): Feature-Level Fusion of Laser Scanner and Video Data for Advanced Driver Assistance Systems. Dissertation at University Ulm. Ulm.

Labayrade, R., Royere, C., Aubert, D. (2005): A collision mitigation system using laser scanner and stereovision fusion and its assessment. In: *IEEE Proceedings. Intelligent Vehicles Symposium, 2005* pp. 441–446. ISBN: 0-7803-8961-1. DOI: 10.1109/IVS.2005.1505143.

Lakemeyer, G., (Ed.) (2003): Exploring artificial intelligence in the new millenium. Morgan Kaufmann, Amsterdam.

Lalonde, J.-F., Vandapel, N., Huber, D. F., Hebert, M. (2006): Natural terrain classification using three-dimensional ladar data for ground robot mobility. *J. Field Robotics*, 23, pp. 839–861.

Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O. (2019): PointPillars: Fast Encoders for Object Detection From Point Clouds. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 12689–12697. ISBN: 978-1-7281-3294-5. DOI: 10.1109/CVPR.2019.01298. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8954311.

Le, T., Duan, Y. (2018): PointGrid: A Deep Network for 3D Shape Understanding. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* IEEE, pp. 9204–9214. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00959.

Li, B. (2017): 3D fully convolutional network for vehicle detection in point cloud. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* pp. 1513–1518. ISBN: 978-1-5386-2683-2. DOI: 10.1109/IROS.2017.8205955. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8205955.

Li, B., Zhang, T., Xia, T., eds. (2016): Vehicle Detection from 3D Lidar Using Fully Convolutional Network Robotics: Science and Systems Foundation. ISBN: 9780992374723. DOI: 10.15607/RSS.2016.XII.

Li, B., Ouyang, W., Sheng, L., Zeng, X., Wang, X. (2019): GS3D: An Efficient 3D Object Detection Framework for Autonomous Driving. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 1019–1028. ISBN: 978-1-7281-3294-5. DOI: 10.1109/CVPR.2019.00111. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8954005.

Li, E., Wang, S., Li, C., Li, D., Wu, X., Hao, Q. (2020): SUSTech POINTS: A Portable 3D Point Cloud Interactive Annotation Platform System. In: *2020 IEEE Intelligent Vehicles Symposium (IV)* pp. 1108–1115. ISBN: 978-1-7281-6674-2. DOI: 10.1109/IV47402.2020.9304562. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9304562.

Li, Y., Ma, L., Zhong, Z., Liu, F., Chapman, M. A., Cao, D., Li, J. (2021): Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 32, 8, pp. 3412–3432, DOI: 10.1109/TNNLS.2020.3015992.

Lin, Z., Hashimoto, M., Takigawa, K., Takahashi, K. (2018): Vehicle and Pedestrian Recognition Using Multilayer Lidar based on Support Vector Machine. In: *2018 25th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)* pp. 1–6. ISBN: 978-1-5386-7545-8. DOI: 10.1109/M2VIP.2018.8600877. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8600877.

Lindenmaier, L., Aradi, S., Becsi, T., Toro, O., Gaspar, P. (2022): GM-PHD Filter Based Sensor Data Fusion for Automotive Frontal Perception System. *IEEE Transactions on Vehicular Technology*, 71, 7, pp. 7215–7229, DOI: 10.1109/TVT.2022.3171040.

Liu, H., Zhang, H., Mertz, C. (2019): DeepDA: LSTM-based Deep Data Association Network for Multi-Targets Tracking in Clutter. In: *2019 22th International Conference on Information Fusion (FUSION)* IEEE, pp. 1–8. ISBN: 978-0-9964527-8-6. DOI: 10.23919/FUSION43075.2019.9011217.

Liu, Y., Wang, P., Gao, Y., Wang, J., Fu, R. (2015): Data association combined with the probability hypothesis density filter for multi-target tracking. In: *2015 International Conference on Wireless Communications & Signal Processing (WCSP)* pp. 1–6. ISBN: 978-1-4673-7686-0. DOI: 10.1109/WCSP.2015.7341181. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7341181.

Liu, Y., Wang, L., Liu, M. (2021): YOLOStereo3D: A Step Back to 2D for Efficient Stereo 3D Detection. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)* pp. 13018–13024. ISBN: 978-1-7281-9078-5. DOI: 10.1109/ICRA48506.2021.9561423. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9561423.

Lu, Z., Hu, W., Kirubarajan, T. (2017): Labeled Random Finite Sets With Moment Approximation. *IEEE Transactions on Signal Processing*, 65, 13, pp. 3384–3398, DOI: 10.1109/TSP.2017.2688960. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7889050.

Luiten, J., Osep, A., Dendorfer, P., Torr, P., Geiger, A., Leal-Taixé, L., Leibe, B. (2021): HOTA: A Higher Order Metric for Evaluating Multi-object Tracking. *International Journal of Computer Vision*, 129, 2, pp. 548–578, DOI: 10.1007/s11263-020-01375-2.

Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., Kim, T.-K. (2021): Multiple object tracking: A literature review. *Artificial Intelligence*, 293, p. 103448, DOI: 10.1016/j.artint.2020.103448.

Mahler, R. P. S. (2003): Multitarget Bayes filtering via first-order multitarget moments. *IEEE Transactions on Aerospace and Electronic Systems*, 39, 4, pp. 1152–1178, DOI: 10.1109/TAES.2003.1261119. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1261119.

Mahler, R. P. S. (2004): "Statistics 101" for multisensor, multitarget data fusion. *IEEE Aerospace and Electronic Systems Magazine*, 19, 1, pp. 53–64, DOI: 10.1109/MAES.2004.1263231.

Mahler, R. P. S. (2007a): PHD filters of higher order in target number. *IEEE Transactions on Aerospace and Electronic Systems*, 43, 4, pp. 1523–1543, DOI: 10.1109/TAES.2007.4441756.

Mahler, R. P. S. (2007b): Statistical multisource-multitarget information fusion. Artech House, Boston. *Artech House information warfare library*. URL: https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=225185.

Mahlisch, M., Hering, R., Ritter, W., Dietmayer, K. (2006): Heterogeneous Fusion of Video, LIDAR and ESP Data for Automotive ACC Vehicle Tracking. In: *2006 IEEE International Conference on Multisensor Fusion*

and Integration for Intelligent Systems IEEE, pp. 139–144. ISBN: 1-4244-0567-X. DOI: 10.1109/MFI.2006.265593.

Marti, E., Miguel, M. A. d., Garcia, F., Perez, J. (2019): A Review of Sensor Technologies for Perception in Automated Driving. *IEEE Intelligent Transportation Systems Magazine*, 11, 4, pp. 94–108, DOI: 10.1109/MITS.2019.2907630. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8846569.

Maturana, D., Scherer, S. (2015): VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* IEEE, pp. 922–928. ISBN: 978-1-4799-9994-1. DOI: 10.1109/IROS.2015.7353481.

Matzka, S., Altendorfer, R. (2008): A comparison of track-to-track fusion algorithms for automotive sensor fusion. In: *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems* IEEE, pp. 189–194. ISBN: 978-1-4244-2143-5. DOI: 10.1109/MFI.2008.4648063.

MEKRA Lang GmbH & Co. KG (2023): Mekra Vision Solutions Trucks. URL: https://www.mekra.de/en/sichtloesungen/lkw. visited on 04/16/2023.

Milan, A., Rezatofighi, S. H., Dick, A., Reid, I., Schindler, K. (2016): Online Multi-Target Tracking Using Recurrent Neural Networks. URL: http://arxiv.org/pdf/1604.03635v2.

Mobus, R., Kolbe, U. (2004): Multi-target multi-object tracking, sensor fusion of radar and infrared. In: *IEEE Intelligent Vehicles Symposium, 2004* pp. 732–737. ISBN: 0-7803-8310-9. DOI: 10.1109/IVS.2004.1336475.

Moosmann, F., Fraichard, T. (2010): Motion estimation from range images in dynamic outdoor scenes. In: *2010 IEEE International Conference on Robotics and Automation* IEEE, pp. 142–147. ISBN: 978-1-4244-5038-1. DOI: 10.1109/ROBOT.2010.5509381.

Mousavian, A., Anguelov, D., Flynn, J., Košecká, J. (2017): 3D Bounding Box Estimation Using Deep Learning and Geometry. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 5632–5640. ISBN: 978-1-5386-0458-8. DOI: 10.1109/CVPR.2017.597. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8100080.

Munkres, J. (1957): Algorithms for the Assignment and Transportation Problems. *Journal of the Society for Industrial and Applied Mathematics*, 5, 1, pp. 32–38. URL: http://www.jstor.org/stable/2098689. visited on 06/20/2022.

Musicki, D., Evans, R. (2004): Joint integrated probabilistic data association: JIPDA. *IEEE Transactions on Aerospace and Electronic Systems*, 40, 3, pp. 1093–1099, DOI: 10.1109/TAES.2004.1337482. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1337482.

Musicki, D., Evans, R., Stankovic, S. (1994): Integrated probabilistic data association. *IEEE Transactions on Automatic Control*, 39, 6, pp. 1237–1241, DOI: 10.1109/9.293185. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=293185.

Naujoks, B., Wuensche, H. J. (2018): An Orientation Corrected Bounding Box Fit Based on the Convex Hull under Real Time Constraints. In: *2018 IEEE Intelligent Vehicles Symposium (IV)* pp. 1–6. ISBN: 978-1-5386-4453-9. DOI: 10.1109/IVS.2018.8500692. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8500692.

Nguyen, A., Le Bac (2013): 3D point cloud segmentation: A survey. In: *2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)* pp. 225–230. ISBN: 978-1-4799-1199-8. DOI: 10.1109/RAM.2013.6758588. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6758588.

Nuss, D., Reuter, S., Thom, M., Yuan, T., Krehl, G., Maile, M., Gern, A., Dietmayer, K. (2018): A random finite set approach for dynamic occupancy grid maps with real-time application. *The International Journal of Robotics Research*, 37, 8, pp. 841–866, DOI: 10.1177/0278364918775523.

On-Road Automated Driving (ORAD) committee (2021): Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. DOI: 10.4271/J3016_202104.

Pang, S., Morris, D., Radha, H. (2020): CLOCs: Camera-LiDAR Object Candidates Fusion for 3D Object Detection. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* IEEE, pp. 10386–10393. ISBN: 978-1-7281-6212-6. DOI: 10.1109/IROS45743.2020.9341791.

Pang, S., Radha, H. (2021): Multi-Object Tracking Using Poisson Multi-Bernoulli Mixture Filtering For Autonomous Vehicles. In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* IEEE, pp. 7963–7967. ISBN: 978-1-7281-7605-5. DOI: 10.1109/ICASSP39728.2021.9415072.

Panta, K., Vo, B., Singh, S. (2005): Improved Probability Hypothesis Density (PHD) Filter for Multitarget Tracking. In: *2005 3rd International Conference on Intelligent Sensing and Information Processing* pp. 213–218. ISBN: 0-7803-9588-3. DOI: 10.1109/ICISIP.2005.1619438. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1619438.

Panta, K., Vo, B. N., Clark, D. E. (2006): An Efficient Track Management Scheme for the Gaussian-Mixture Probability Hypothesis Density Tracker. In: *2006 Fourth International Conference on Intelligent Sensing and Information Processing* pp. 230–235. ISBN: 1-4244-0612-9. DOI: 10.1109/ICISIP.2006.4286102. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4286102.

Park, Y., Dang, L. M., Lee, S., Han, D., Moon, H. (2021): Multiple Object Tracking in Deep Learning Approaches: A Survey. *Electronics*, 10, 19, p. 2406, DOI: 10.3390/electronics10192406.

Patole, S. M., Torlak, M., Wang, D., Ali, M. (2017): Automotive radars: A review of signal processing techniques. *IEEE Signal Processing Magazine*, 34, 2, pp. 22–35, DOI: 10.1109/MSP.2016.2628914. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7870764.

Petersen, E., Scholze, C., Böhnke, R. (2018): Notbremsassistenzsysteme im Lkw – Eine Analyse niedersächsischer Autobahnunfälle des Jahres 2017 und Einfluss aktueller Systeme. *Zeitschrift für Verkehrssicherheit*, 64, 5, pp. 336–344.

Pfeiffer, D., Franke, U. (2011): Modeling Dynamic 3D Environments by Means of The Stixel World. *IEEE Intelligent Transportation Systems Magazine*, 3, 3, pp. 24–36, DOI: 10.1109/MITS.2011.942207. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5992863.

Pietzsch, S., Aycard, O., Burlet, J., Vu, T. D., Hackbarth, T., Appenrodt, N., Dickmann, J., Radig, B. (2008): Results of a precrash application based on Laserscanner and short range radars. In: *2008 IEEE Intelligent Vehicles Symposium* pp. 367–372. ISBN: 978-1-4244-2569-3. DOI: 10.1109/IVS.2008.4621264. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4621264.

Pinchon, N., Cassignol, O., Nicolas, A., Bernardin, F., Leduc, P., Tarel, J.-P., Brémond, R., Bercier, E., Brunet, J. (2019): All-Weather Vision for Automotive Safety: Which Spectral Band?. In: *Advanced Microsystems for Automotive Applications 2018*. Dubbert, J., Müller, B., Meyer, G. [Ed.] Cham: Springer International Publishing, pp. 3–15. ISBN: 978-3-319-99762-9.

Punke, M., Menzel, S., Werthessen, B., Stache, N., Höpfl, M. (2016): Automotive Camera (Hardware). In: *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*. Winner, H., Hakuli, S., Lotz, F., Singer, C. [Ed.] Cham: Springer International Publishing, pp. 431–460. ISBN: 978-3-319-12352-3. DOI: 10.1007/978-3-319-12352-3_20.

Qian, R., Lai, X., Li, X. (2021): 3D Object Detection for Autonomous Driving: A Survey. URL: http://arxiv.org/pdf/2106.10823v1.

Qin, Z., Wang, J., Lu, Y. (2021): MonoGRNet: A General Framework for Monocular 3D Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP, 99, p. 1, DOI: 10.1109/TPAMI.2021.3074363. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9409679.

Qu, S., Chen, G., Ye, C., Lu, F., Wang, F., Xu, Z., Gel, Y. (2018): An Efficient L-Shape Fitting Method for Vehicle Pose Detection with 2D LiDAR. In: *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)* pp. 1159–1164. ISBN: 978-1-7281-0378-5. DOI: 10.1109/ROBIO.2018.8665265. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8665265.

Quehl, J., Yan, S., Wirges, S., Pauls, J.-H., Lauer, M. (2019): Estimating Object Shape and Movement Using Local Occupancy Grid Maps. *IFAC-PapersOnLine*, 52, 8, pp. 87–92, DOI: 10.1016/j.ifacol.2019.08.053.

Raj, T., Hashim, F. H., Huddin, A. B., Ibrahim, M. F., Hussain, A. (2020): A Survey on LiDAR Scanning Mechanisms. *Electronics*, 9, 5. DOI: 10.3390/electronics9050741. URL: https://www.mdpi.com/2079-9292/9/5/741.

Rakai, L., Song, H., Sun, S., Zhang, W., Yang, Y. (2022): Data association in multiple object tracking: A survey of recent techniques. *Expert Systems with Applications*, 192, p. 116300, DOI: 10.1016/j.eswa.2021.116300.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016): You Only Look Once: Unified, Real-Time Object Detection. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 779–788. ISBN: 978-1-4673-8852-8. DOI: 10.1109/CVPR.2016.91. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7780460.

Reuter, S. (2014): Multi-Object Tracking Using Random Finite Sets. Dissertation at University Ulm. Ulm.

Rezaei, M., Terauchi, M., Klette, R. (2015): Robust Vehicle Detection and Distance Estimation Under Challenging Lighting Conditions. *IEEE Transactions on Intelligent Transportation Systems*, 16, 5, pp. 2723–2743, DOI: 10.1109/TITS.2015.2421482. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7101268.

Rezatofighi, H., Nguyen, T. T. D., Vo, B.-n., Vo, B.-T., Savarese, S., Reid, I. (2020): How Trustworthy are Performance Evaluations for Basic Vision Tasks? DOI: 10.48550/arXiv.2008.03533.

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S. (2019): Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* DOI: 10.1109/CVPR.2019.00075.

Ristic, B. (2013): Particle Filters for Random Set Models. Springer New York, New York, NY. URL: http://gbv.eblib.com/patron/FullRecord.aspx?p=1205307.

Roriz, R., Cabral, J., Gomes, T. (2021): Automotive LiDAR Technology: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, PP, 99, pp. 1–16, DOI: 10.1109/TITS.2021.3086804. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9455394.

Rosique, F., Navarro, P. J., Fernández, C., Padilla, A. (2019): A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research. *Sensors*, 19, 3. DOI: 10.3390/s19030648. URL: https://www.mdpi.com/1424-8220/19/3/648.

Roth, M., Jargot, D., Gavrila, D. M. (2019): Deep End-to-end 3D Person Detection from Camera and Lidar. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)* IEEE, pp. 521–527. ISBN: 978-1-5386-7024-8. DOI: 10.1109/ITSC.2019.8917366.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C. (2018): MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* IEEE, pp. 4510–4520. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00474.

Scheider, A. (2021): Identifikation von Systemmodellen zur dreidimensionalen Zustandsschätzung eines Peilschiffs mit Propellerantrieb unter Verwendung eines Multi-Sensorsystems. Vol. Heft Nr. 868. München. *DGK: C (Dissertationen)*. URL: https://publikationen.badw.de/en/047235679.

Schlangen, I., Delande, E. D., Houssineau, J., Clark, D. E. (2018): A Second-Order PHD Filter With Mean and Variance in Target Number. *IEEE Transactions on Signal Processing*, 66, 1, pp. 48–63, DOI: 10.1109/TSP.2017.2757905. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8052530.

Schreier, M. (2015): Bayesian Environment Representation, Prediction, and Criticality Assessment for Driver Assistance Systems. Dissertation at Technical University Darmstadt. Darmstadt.

Schreier, M. (2018): Environment Representations for Automated On-Road Vehicles. *at - Automatisierungstechnik*, 66, pp. 107–118, DOI: 10.1515/auto-2017-0104.

Schubert, R., Richter, E., Wanielik, G. (2008): Comparison and evaluation of advanced motion models for vehicle tracking. In: *2008 11th International Conference on Information Fusion* pp. 1–6. ISBN: 978-3-00-024883-2. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4632283.

Schueler, K., Weiherer, T., Bouzouraa, E., Hofmann, U. (2012): 360 Degree multi sensor fusion for static and dynamic obstacles. In: *2012 IEEE Intelligent Vehicles Symposium* IEEE, pp. 692–697. ISBN: 978-1-4673-2118-1. DOI: 10.1109/IVS.2012.6232253.

Schuhmacher, D., Vo, B. T., Vo, B. N. (2008): A Consistent Metric for Performance Evaluation of Multi-Object Filters. *IEEE Transactions on Signal Processing*, 56, 8, pp. 3447–3457, DOI: 10.1109/TSP.2008.920469. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4567674.

Schwalbe, G., Schels, M. (2020): Concept Enforcement and Modularization as Methods for the ISO 26262 Safety Argumentation of Neural Networks. URL: https://hal.archives-ouvertes.fr/hal-02442796.

Schütz, M., Appenrodt, N., Dickmann, J., Dietmayer, K. (2014): Occupancy grid map-based extended object tracking. In: *2014 IEEE Intelligent Vehicles Symposium Proceedings* pp. 1205–1210. ISBN: 978-1-4799-3638-0. DOI: 10.1109/IVS.2014.6856504. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6856504.

Shabani, M., Gholami, A., Davari, N. (2015): Asynchronous direct Kalman filtering approach for underwater integrated navigation system. *Nonlinear Dynamics*, 80, 1-2, pp. 71–85, DOI: 10.1007/s11071-014-1852-9.

Shabat, Y. B., Lindenbaum, M., Fischer, A. (2018): 3DmFV: Three-Dimensional Point Cloud Classification in Real-Time Using Convolutional Neural Networks. *IEEE Robotics and Automation Letters*, 3, 4, pp. 3145–3152, DOI: 10.1109/LRA.2018.2850061. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8394990.

Shafer, G. (1976): A mathematical theory of evidence. Princeton Univ. Press, Princeton, NJ.

Shalom, Y. B., Daum, F., Huang, J. (2009): The probabilistic data association filter. *IEEE Control Systems Magazine*, 29, 6, pp. 82–100, DOI: 10.1109/MCS.2009.934469. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5338565.

Shi, K., Shi, Z., Yang, C., He, S., Chen, J., Chen, A. (2022): Road-Map Aided GM-PHD Filter for Multivehicle Tracking With Automotive Radar. *IEEE Transactions on Industrial Informatics*, 18, 1, pp. 97–108, DOI: 10.1109/TII.2021.3073032.

Shi, S., Wang, X., Li, H. (2019): PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 770–779. ISBN: 978-1-7281-3294-5. DOI: 10.1109/CVPR.2019.00086. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8954080.

Simon, M., Amende, K., Kraus, A., Honer, J., Sämann, T., Kaulbersch, H., Milz, S., Gross, H. M. (2019a): Complexer-YOLO: Real-Time 3D Object Detection and Tracking on Semantic Point Clouds. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* pp. 1190–1199. ISBN: 978-1-7281-2507-7. DOI: 10.1109/CVPRW.2019.00158. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9025351.

Simon, M., Milz, S., Amende, K., Gross, H.-M. (2019b): Complex-YOLO: An Euler-Region-Proposal for Real-Time 3D Object Detection on Point Clouds. In: *Computer Vision – ECCV 2018 Workshops*. Leal-Taixé, L., Roth, S. [Ed.]. *SpringerLink Bücher*. Cham: Springer International Publishing. ISBN: 978-3-030-11009-3. DOI: 10.1007/978-3-030-11009-3_11.

Simonelli, A., Rota Bulo, S., Porzi, L., Lopez Antequera, M., Kontschieder, P. (2020): Disentangling Monocular 3D Object Detection: From Single to Multi-Class Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1, DOI: 10.1109/TPAMI.2020.3025077.

Smets, P. (1990): Constructing the Pignistic Probability Function in a Context of Uncertainty. In: *Uncertainty in Artificial Intelligence*. Vol. 10. *Machine Intelligence and Pattern Recognition*. Elsevier, pp. 29–39. ISBN: 9780444887382. DOI: 10.1016/B978-0-444-88738-2.50010-5.

Stüker, D. (2003): Heterogene Sensordatenfusion zur robusten Objektverfolgung im automobilen Straßenverkehr. Dissertation at Carl von Ossietzky-Universität Oldenburg. Oldenburg.

Su, H., Maji, S., Kalogerakis, E., Miller, E. L. (2015): Multi-view Convolutional Neural Networks for 3D Shape Recognition. In: *2015 IEEE International Conference on Computer Vision (ICCV)* pp. 945–953. ISBN: 978-1-4673-8390-5. DOI: 10.1109/ICCV.2015.114. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7410471.

Takizawa, H., Yamada, K., Ito, T. (2004): Vehicles detection using sensor fusion. In: *IEEE Intelligent Vehicles Symposium, 2004* pp. 238–243. ISBN: 0-7803-8310-9. DOI: 10.1109/IVS.2004.1336388. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1336388.

Teichman, A., Levinson, J., Thrun, S. (2011): Towards 3D object recognition via classification of arbitrary object tracks. In: *2011 IEEE International Conference on Robotics and Automation* pp. 4034–4041. ISBN: 978-1-61284-380-3. DOI: 10.1109/ICRA.2011.5979636. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5979636.

Thrun, S., Burgard, W., Fox, D. (2005): Probabilistic robotics. MIT Press, Cambridge, Mass. and London. *Intelligent robotics and autonomous agents*.

Thrun, S. et al. (2007): Stanley: The Robot That Won the DARPA Grand Challenge. In: *The 2005 DARPA Grand Challenge: The Great Robot Race*. Buehler, M., Iagnemma, K., Singh, S. [Ed.] Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–43. ISBN: 978-3-540-73429-1. DOI: 10.1007/978-3-540-73429-1_1.

Törő, O., Bécsi, T., Gáspár, P. (2021): PHD Filter for Object Tracking in Road Traffic Applications Considering Varying Detectability. *Sensors*, 21, 2. DOI: 10.3390/s21020472.

UNECE (2014): Regulation No 46 of the Economic Commission for Europe of the United Nations (UNECE) — Uniform provisions concerning the approval of devices for indirect vision and of motor vehicles with regard to the installation of these devices.

Unger, T. (2011): Konstellationen bei Auffahrunfällen. *Berichte der ADAC Unfallforschung*.

Vargas, J., Alsweiss, S., Toker, O., Razdan, R., Santos, J. (2021): An Overview of Autonomous Vehicles Sensors and Their Vulnerability to Weather Conditions. *Sensors*, 21, 16. DOI: 10.3390/s21165397.

Vasic, M., Martinoli, A. (2015): A Collaborative Sensor Fusion Algorithm for Multi-object Tracking Using a Gaussian Mixture Probability Hypothesis Density Filter. In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems* IEEE, pp. 491–498. ISBN: 978-1-4673-6596-3. DOI: 10.1109/ITSC.2015.87.

Veltkamp, R. C. (2001): Shape matching: similarity measures and algorithms. In: *Proceedings International Conference on Shape Modeling and Applications* pp. 188–197. ISBN: 0-7695-0853-7. DOI: 10.1109/SMA.2001.923389. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=923389.

Viola, P., Jones, M. (2001): Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1 p. I. ISBN: 0-7695-1272-0. DOI: 10.1109/CVPR.2001.990517. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=990517.

Vo, A.-V., Truong-Hong, L., Laefer, D. F., Bertolotto, M. (2015): Octree-based region growing for point cloud segmentation. *Isprs Journal of Photogrammetry and Remote Sensing*, 104, pp. 88–100.

Vo, B. N., Ma, W. K. (2006): The Gaussian Mixture Probability Hypothesis Density Filter. *IEEE Transactions on Signal Processing*, 54, 11, pp. 4091–4104, DOI: 10.1109/TSP.2006.881190. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1710358.

Vo, B.-T. (2023): RFS Tracking Toolbox. URL: https://ba-tuong.vo-au.com/codes.html. visited on 01/30/2023.

Vo, B. T., Vo, B. N. (2012): The para-normal Bayes multi-target filter and the spooky effect. In: *2012 15th International Conference on Information Fusion* pp. 173–180.

Vo, B. T., Vo, B. N. (2013): Labeled Random Finite Sets and Multi-Object Conjugate Priors. *IEEE Transactions on Signal Processing*, 61, 13, pp. 3460–3475, DOI: 10.1109/TSP.2013.2259822. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6507656.

Vo, B. T., Vo, B. N., Cantoni, A. (2009): The Cardinality Balanced Multi-Target Multi-Bernoulli Filter and Its Implementations. *IEEE Transactions on Signal Processing*, 57, 2, pp. 409–423, DOI: 10.1109/TSP.2008.2007924. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4663921.

Wang, C.-H., Chen, H.-W., Fu, L.-C. (2021): VPFNet: Voxel-Pixel Fusion Network for Multi-class 3D Object Detection. DOI: 10.48550/arXiv.2111.00966.

Wang, D., Watkins, C., Xie, H. (2020): MEMS Mirrors for LiDAR: A review. *Micromachines*, 11, 5. DOI: 10.3390/mi11050456.

Wang, Y., Chao, W. L., Garg, D., Hariharan, B., Campbell, M., Weinberger, K. Q. (2019): Pseudo-LiDAR From Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 8437–8445. ISBN: 978-1-7281-3294-5. DOI: 10.1109/CVPR.2019.00864. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8954293.

Weiherer, T., Bouzouraa, E., Hofmann, U. (2012): A generic map based environment representation for driver assistance systems applied to detect convoy tracks. In: *2012 15th International IEEE Conference on Intelligent Transportation Systems* pp. 691–696. ISBN: 978-1-4673-3062-6. DOI: 10.1109/ITSC.2012.6338658. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6338658.

Winner, H. (2016): Automotive RADAR. In: *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*. Winner, H., Hakuli, S., Lotz, F., Singer, C. [Ed.] Cham: Springer International Publishing, pp. 325–403. ISBN: 978-3-319-12352-3. DOI: 10.1007/978-3-319-12352-3_17.

Wu, H., Han, W., Wen, C., Li, X., Wang, C. (2022): 3D Multi-Object Tracking in Point Clouds Based on Prediction Confidence-Guided Data Association. *IEEE Transactions on Intelligent Transportation Systems*, 23, 6, pp. 5668–5677, DOI: 10.1109/TITS.2021.3055616. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9352500.

Wu, Y., Wang, Y., Zhang, S., Ogai, H. (2021): Deep 3D Object Detection Networks Using LiDAR Data: A Review. *IEEE Sensors Journal*, 21, 2, pp. 1152–1171, DOI: 10.1109/JSEN.2020.3020626.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J. (62015): 3D ShapeNets: A deep representation for volumetric shapes. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* IEEE, pp. 1912–1920. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7298801.

Xie, G., Gao, H., Qian, L., Huang, B., Li, K., Wang, J. (2018): Vehicle Trajectory Prediction by Integrating Physics- and Maneuver-Based Approaches Using Interactive Multiple Models. *IEEE Transactions on Industrial Electronics*, 65, 7, pp. 5999–6008, DOI: 10.1109/TIE.2017.2782236. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8186191.

Yan, Y., Mao, Y., Li, B. (2018): SECOND: Sparsely Embedded Convolutional Detection. *Sensors*, 18, 10. DOI: 10.3390/s18103337. URL: https://www.mdpi.com/1424-8220/18/10/3337.

Yang, B., Luo, W., Urtasun, R. (2018): PIXOR: Real-time 3D Object Detection from Point Clouds. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* pp. 7652–7660. ISBN: 978-1-5386-6421-6. DOI: 10.1109/CVPR.2018.00798. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8578896.

Yang, Z., Sun, Y., Liu, S., Jia, J. (2020): 3DSSD: Point-Based 3D Single Stage Object Detector. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 11037–11045. ISBN: 978-1-7281-7169-2. DOI: 10.1109/CVPR42600.2020.01105. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9156597.

Yi, Z., Khing, H., Seng, C., Wei, Z. (2000): Multi-ultrasonic sensor fusion for mobile robots. In pp. 387–391. ISBN: 0-7803-6363-9. DOI: 10.1109/IVS.2000.898374.

Yoshioka, M., Suganuma, N., Yoneda, K., Aldibaja, M. (2017): Real-time object classification for autonomous vehicle using LIDAR. In: *2017 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)* pp. 210–211. ISBN: 978-1-5090-6665-0. DOI: 10.1109/ICIIBMS.2017.8279696. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8279696.

Yu, Y., Gao, Z., Zhu, B., Zhao, J. (2014): Recognition and Classification of Vehicle Target Using the Vehicle-Mounted Velodyne LIDAR. In: *SAE Technical Paper Series*. *SAE Technical Paper Series*. SAE International400 Commonwealth Drive, Warrendale, PA, United States. DOI: 10.4271/2014-01-0322.

Yu, Z., Zhang, Q., Yu, K., Zheng, N. (2021): A State-Domain Robust Chi-Square Test Method for GNSS/INS Integrated Navigation. *Journal of Sensors*, 2021, pp. 1–8, DOI: 10.1155/2021/1745383.

Zeng, Y., Hu, Y., Liu, S., Ye, J., Han, Y., Li, X., Sun, N. (2018): RT3D: Real-Time 3-D Vehicle Detection in LiDAR Point Cloud for Autonomous Driving. *IEEE Robotics and Automation Letters*, 3, 4, pp. 3434–3440, DOI: 10.1109/LRA.2018.2852843. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8403277.

Zermas, D., Izzat, I., Papanikolopoulos, N. (2017): Fast segmentation of 3D point clouds: A paradigm on LiDAR data for autonomous vehicle applications. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)* pp. 5067–5073. ISBN: 978-1-5090-4634-8. DOI: 10.1109/ICRA.2017.7989591. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7989591.

Zhang, X., Xu, W., Dong, C., Dolan, J. M. (2017): Efficient L-shape fitting for vehicle detection using laser scanners. In: *2017 IEEE Intelligent Vehicles Symposium (IV)* pp. 54–59. ISBN: 978-1-5090-4805-2. DOI: 10.1109/IVS.2017.7995698. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7995698.

Zhang, Z. (2021): Object detection in front of a Truck using a wide-angle front mirror replacement camera. Masterarbeit at University of Stuttgart. Institut für Fahrzeugtechnik Stuttgart, Stuttgart.

Zhao, Y., Zhang, X., Huang, X. (2021): A Technical Survey and Evaluation of Traditional Point Cloud Clustering Methods for LiDAR Panoptic Segmentation. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops* pp. 2464–2473.

Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., Ren, D. (2020): Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34, 07, pp. 12993–13000, DOI: 10.1609/aaai.v34i07.6999.

Zhi, S., Liu, Y., Li, X., Guo, Y. (2017): LightNet: A Lightweight 3D Convolutional Neural Network for Real-Time 3D Object Recognition. DOI: 10.2312/3dor.20171046.

Zhou, J., Tan, X., Shao, Z., Ma, L. (2019): FVNet: 3D Front-View Proposal Generation for Real-Time Object Detection from Point Clouds. In: *2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)* pp. 1–8. ISBN: 978-1-7281-4853-3. DOI: 10.1109/CISP-BMEI48845.2019.8965844. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8965844.

Zhou, Y., Tuzel, O. (2018): VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* pp. 4490–4499. ISBN: 978-1-5386-6421-6. DOI: 10.1109/CVPR.2018.00472. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8578570.

Zhu, H., Deng, J., Zhang, Y., Ji, J., Mao, Q., Li, H., Zhang, Y. (2022): VPFNet: Improving 3D Object Detection with Virtual Point based LiDAR and Stereo Data Fusion. *IEEE Transactions on Multimedia*, pp. 1–14, DOI: 10.1109/TMM.2022.3189778.

Zou, Z., Shi, Z., Guo, Y., Ye, J. (2019): Object Detection in 20 Years: A Survey. URL: http://arxiv.org/pdf/1905.05055v2.