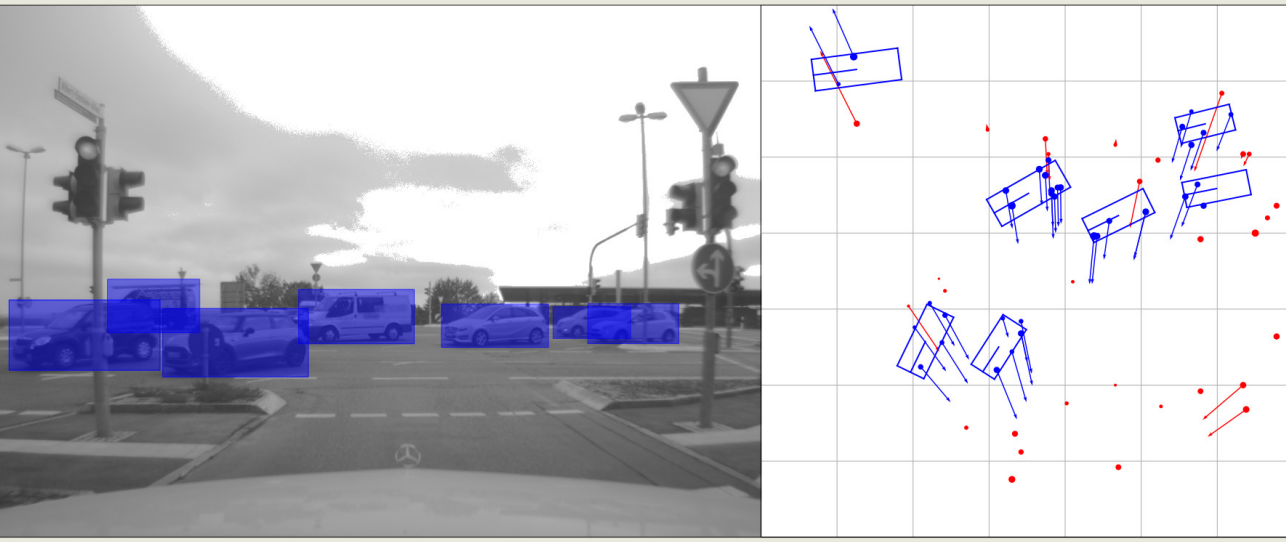
universität ulm



Object Detection in High-Resolution Radar Data

Andreas Danzer

Band 43

Schriftenreihe des Instituts für Mess-, Regel- und Mikrotechnik

Andreas Michael Danzer

Object Detection in High-Resolution Radar Data

Schriftenreihe des
Instituts für Mess-, Regel- und Mikrotechnik
Universität Ulm

Herausgeber:
Prof. Dr.-Ing. Klaus Dietmayer

Band 43

Andreas Michael Danzer

Object Detection in High-Resolution Radar Data

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar.

Dissertation, Universität Ulm,
Fakultät für Ingenieurwissenschaften, Informatik und Psychologie, 2022

Impressum

Universität Ulm
Institut für Mess-, Regel- und Mikrotechnik
Prof. Dr.-Ing. Klaus Dietmayer
Albert-Einstein-Allee 41
89081 Ulm
http://www.uni-ulm.de/mrm

Eine Übersicht über alle Bände der Schriftenreihe finden Sie unter
http://www.uni-ulm.de/mrmschriften.

ISBN 978-3-941543-68-3 (E-Book)

universität
uulm

Object Detection in High-Resolution Radar Data

DISSERTATION

zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS

(Dr.-Ing.)

der Fakultät für Ingenieurwissenschaften, Informatik
und Psychologie der Universität Ulm

von

Andreas Michael Danzer
aus Heidenheim an der Brenz

Gutachter:	Prof. Dr.-Ing. Klaus Dietmayer
	Prof. Dr.-Ing. Christian Waldschmidt
Amtierende Dekanin:	Prof. Dr. Anke Huckauf

Ulm, 13.07.2022

Abstract

Modern vehicles are equipped with many semi-automated and even highly automated driving functions, which enormously increase safety and comfort while traveling. A basic prerequisite for these features is an accurate and reliable perception of the vehicle's environment, which is modeled by suitable algorithms on the basis of various sensor measurements. One appropriate sensor for this purpose is a radar sensor, which is able to directly measure the position, the orientation as well as the radial velocity of objects in the vehicle's surroundings. Contemporary automotive radar sensors provide high-resolution data, such that multiple radar reflections are generated for a single object. Subsequently, the object detection task is to recognize all existing objects in the measured radar target list which means to determine their localization and class. This thesis proposes a novel technique for object detection in high-resolution radar data, which is then further used to realize environment perception at an object level. In summary, this covers the complete processing pipeline from a radar target list to the object recognition and multi-object tracking.

The radar object detection system designed in this thesis is based on a Deep Learning technique, where the entire processing chain basically consisting of three main components. The first module divides the radar target list into several small regions and determines whether there is an object located in it, and if given, then its associated class. The second component segments all radar targets belonging to the actual object of interest. Finally, the third module determines the parameters for a 2D bounding box modeling the respective object and outputs all predicted hypotheses after the object extraction. The training of all neural networks exclusively utilizes real radar data from typical traffic situations in rural and urban areas. The evaluation validates the radar object detector based on three different datasets, and scores the intermediate results of each module as well as the final object detection result. Two of the radar datasets are publicly available ensuring that all results are reproducible and verifiable. The third dataset is self-generated and includes high-resolution radar data from real sensors. The evaluation reveals that the radar object detector provides convincing results on all datasets, where the best performance is obtained with the self-generated dataset. Further, a successful implementation and a parallel operation of three radar object detection systems on three radar sensors installed in a real system for autonomous driving demonstrates that this technology is real-time capable and appropriate for real-world applications in the automotive field.

In the case of multiple sensors with an overlapping field of view, the respective radar object detectors provide different hypotheses for the same object, which may be contradictory. In order to obtain a consistent environment model, it is necessary to fuse and track all generated object detections over time. A multi-object tracking algorithm allows to combine detections from independent sources into one overall result. A tracking algorithm consists of two main stages: the prediction of existing objects according to a motion model and the correction of all object states based on measurements. This thesis presents a measurement model to process all hypotheses provided by the developed radar object detection system using an established multi-object tracking algorithm. The tracker processes all radar object detections and generates a robust environment model. Therefore, the algorithm optimizes the perception of objects such that the resulting tracks correspond accurately to reality. The proposed radar object detector and the subsequent multi-object tracking perform precisely and reliably in real-world applications, which makes this advanced technique an essential component for the environment perception in a real system for automated driving.

Kurzfassung

Moderne Fahrzeuge verfügen über viele teil- und sogar hochautomatisierte Fahrfunktionen, welche die Sicherheit und den Komfort während der Fahrt enorm erhöhen. Als Grundvoraussetzung erfordern solche Systeme eine genaue und zuverlässige Wahrnehmung der Umgebung um das Fahrzeug, welche durch dafür vorgesehene Algorithmen auf Basis von verschiedenen Sensormessungen modelliert wird. Ein für diesen Zweck geeigneter Sensor stellt das Radar dar, welches in der Lage ist Position, Orientierung und Radialgeschwindigkeit von Objekten in der Fahrzeugumgebung direkt zu messen. Heutige Automobilradarsensoren liefern bereits hochauflösende Daten, sodass für ein Objekt mehrere Radarreflektionen erzeugt werden. Anschließend ist es die Aufgabe der Objektdetektion in der gemessenen *Radartargetliste* alle vorhandenen Objekte zu erkennen, das heißt deren Lokalisierung sowie deren zugehörigen Klasse zu bestimmen. Diese Arbeit stellt eine neuartige Technik für die Objektdetektion in hochauflösenden Radardaten vor, welche anschließend weiterverarbeitet wird um eine Wahrnehmung der Umgebung auf Objektebene zu realisieren. Zusammenfassend handelt es sich um ein vollständiges System von der Radartargetliste über die Objekterkennung bis hin zur zeitlichen Verfolgung all dieser detektierten Objekten.

Das in dieser Arbeit entworfene Radar-Objektdetektionssystem basiert auf der Methode des Deep Learning, wobei der Detektor aus drei Hauptkomponenten besteht. Das erste Modul teilt die Radartargetliste in mehrere kleine Regionen auf und bestimmt, ob sich darin ein Objekt befindet und gegeben des Falls auch dessen Klassen. Die zweite Komponente segmentiert alle Radarreflektionen, die zum Zielobjekt gehören. Das dritte Modul bestimmt schließlich die Parameter für die zugehörige 2D-Bounding-Box, welche das Objekt modelliert. Dieser Prozess wird für alle Teiltargetlisten durchgeführt und letztendlich fasst die Objektextraktion alle prädizierten Hypothesen zu einem gesamtheitlichen Detektionsergebnis zusammen. Das Training der neuronalen Netzwerke verwendet ausschließlich reale Radardaten, welche aus typischen Verkehrssituationen in ländlichen und städtischen Gebieten stammen. Die Auswertung evaluiert den Radar-Objektdetektor auf drei verschiedenen Datensätze und bewertet die Zwischenergebnisse der einzelnen Module sowie das eigentliche Detektionsergebnis. Zwei der Radardatensätze sind öffentlich zugänglich, sodass alle Ergebnisse reproduzierbar und nachvollziehbar sind. Der dritte Datensatz ist selbst erzeugt und enthält hochauflösende Radardaten von echten Radarsensoren. Die Evaluation zeigt, dass der Objektdetektor auf allen Datensätzen überzeugende Ergeb-

nisse liefert, wobei die Detektionsresultate auf dem selbst erzeugten Datensatz am besten abschneiden. Außerdem beweist eine erfolgreiche Implementierung sowie ein paralleler Betrieb von drei Radar-Objektdetektionssystemen auf drei Sensoren, die in einem realen System für autonomes Fahren installiert sind, dass diese fortschrittliche Technologie echtzeitfähig ist und sich für den Einsatz im realen Straßenverkehr eignet.

Beim Einsatz von mehreren Sensoren mit einem sich überlappenden Sichtbereich, liefern die zugehörigen Radar-Objektdetektoren für das gleiche Objekt verschiedene Hypothesen, die sich durchaus widersprechen können. Um ein konsistentes Umgebungsmodell zu erhalten, müssen alle erzeugen Objektdetekionen fusioniert und über die Zeit verfolgt werden. Ein Algorithmus für *Multi-Objekt-Tracking* ermöglicht es, Messungen aus unabhängigen Quellen zu einem ganzheitlichen Ergebnis zu kombinieren. Solch ein Trackingalgorithmus besteht aus zwei wesentlichen Schritten, der Prädiktion von bestehenden Objekten gemäß einem Bewegungsmodell und der darauffolgenden Korrektur von aktuellen Objektzuständen mit Messungen. Diese Arbeit stellt ein geeignetes Messmodell vor, um alle Hypothesen, die das entwickelte Radar-Objektdetektionssystem liefert, mit dem eingesetzten Multi-Objekt-Trackingalgorithmus zu verarbeiten. Der Multi-Objekt-Tracker erzeugt unter Nutzung der gefundenen Radar-Objektdetektionen ein robustes Umgebungsmodell. Dabei optimiert der Algorithmus die Wahrnehmung von Objekten, sodass die resultierenden getrackten Objekte genau denen in der Realität entsprechen. Der in dieser Arbeit vorgeschlagene Radar-Objektdetektor und das anschließende Multi-Objekt-Tracking funktioniert im realen Betrieb sowohl schnell und präzise, als auch zuverlässig, sodass diese Technik ein unerlässlicher Bestandteil für die Umgebungswahrnehmung in einer realen Anwendung für das automatisierte Fahren darstellt.

Contents

Chapter 1

Introduction

Object detection is the task of recognizing the state, including localization and classification, for all objects in the surrounding based on uncertain sensor measurements. This thesis focuses on the detection of objects using noisy measurements provided by a high-resolution automotive radar sensor. Due to the increased resolution of such a radar sensor, multiple reflections per object are generated. This thesis proposes a novel radar object detection system which determines hypotheses for objects from different classes solely based on radar target lists using a deep learning technique.

For many Advanced Driver Assistance Systems (ADAS) and automated driving functions, a highly accurate perception of the vehicle's environment is a crucial prerequisite. In order to detect traffic participants in the surrounding of the vehicle, modern vehicles are equipped with different sensors, such as radio detection and ranging (radar), light detection and ranging (lidar) and camera sensors. Further, a multi-object tracking algorithm processes data generated by these sensors and often preprocessed by object recognition algorithms to model each traffic participant as a dynamic object. A tracking system provides object state estimations comprising, for example position, orientation, velocity and extent, based on the noisy input data. Each sensor is characterized by particular advantages and deficiencies. Camera sensors provide semantic information and show remarkably results in the classification task of objects. Lidar sensors have a high distance and angular resolution, and are able to yield precise information about the geometry of objects. However, both sensor are sensitive to weather conditions and occlusion by other objects. In contrast, radar sensors are more robust against all weather conditions and enable the observation of occluded objects through multipath propagation. One of the main advantage is, a radar sensor is able to directly measure the object velocity, more precisely, the object's radial velocity based on the Doppler effect. However, a tracking algorithm achieves best results by a sophisticated fusion of information from different sensors. Nevertheless, this thesis focuses on a multi-object tracking approach exclusively using radar data to demonstrate the potential of high-resolution automotive radar sensors.

Modern automotive radar sensors provide high-resolution measurement data. As a consequence, radar sensors are able to generate multiple measurements for each object, hence, the measurement data yield information about its extent. A classical tracking algorithm follows the assumption that exactly one measurement is generated per object. This assumption is violated when using high-resolution sensors. There are two common approaches to address this challenge. One approach encompass tracking methods which are able to process multiple measurements per object to solve an extended object problem. An advantage is that these techniques process each measurement individually, and consequently the entire measurement information is applied. However, these tracking filters often use very complicated measurement models to find a relation between sensor measurements and object states. Another approach to deal with multiple measurement per object is to preprocess the measurements first, and subsequently apply a classical tracking algorithm using a more simple measurement model. For this purpose, a preprocessing method combines all measurements to a representation which satisfies the assumption of one measurement per object, for example a single object detection. Therefore, a detector generates object hypotheses using the sensor's measurements, that can be a bounding box which models an object and represents position, orientation and extent of the object.

A high-resolution automotive radar sensor delivers a list of radar targets per measurement cycle. Such a target list contains various information about a measured reflection, for example spatial coordinates, radial velocity and reflectivity of the measured target. The goal of the radar object detection system proposed in this thesis is to estimate a classified bounding box comprising the entire object based on all radar targets belonging to the object of interest. For this purpose, there are three major challenges. Firstly, the classification of an entire list with multiple radar targets to decide whether an object is located there or not. Secondly, the segmentation of all radar targets which belong to the object of interest by classifying each radar target. Thirdly, the estimation of a bounding box comprising the entire object based on the segmented radar targets. In summary, the proposed radar object detector performs classification, segmentation and bounding box estimation to recognize different objects solely using the radar data from an automotive sensor.

There are several existing approaches to process radar data using machine learning methods. Most neuronal networks expect a defined input format, for example images with a fixed size. For that reason, the radar data is first transformed into a certain regular format, for example into a grid map representation, before it will be fed into a neuronal network. Since the radar data is naturally represented as radar target list, it is recommended to directly process this list as input for the neuronal network. In literature, the PointNets are proposed which facilitate to directly process target lists and are therefore particularly well suited for this radar data format. In a further work, the Frustum PointNets extend this approach which finally results in the first object detector based on PointNets. Therefore, camera information is fused with a

lidar point cloud, in other words, a 2D camera object detection is combined with a 3D instance segmentation and a 3D bounding box estimation using lidar point clouds. In contrast, this thesis presents an radar object detector to perform a 2D object detection exclusively in radar data based on the concept of PointNets. The proposed method classifies an object and segments all radar targets belonging to it, as well as applies a bounding box regression based on the segmented radar targets.

The developed radar object detection system provides one hypothesis for each object, that means, one classified 2D bounding box represents one object. Hence, this fulfills the assumption of a single measurement for each object, and consequently, a classical multi-object tracking algorithm is applicable. A multi-object tracking algorithm consists of two key elements, the prediction and the update. The prediction processes the multi-object state, that is the number of tracks and their individual states by predicting it to the next measurement time step using a multi-object motion model. The update deals with measurements provided by a sensor using a multi-object measurement model which corrects the predicted multi-object state. The results of a multi-object update depend on two factors: firstly, the sensor which generates measurements, and secondly, the measurement model assigning raw or preprocessed measurements to the existing objects. Since a sensor with long range and high resolution provides much more precise information about objects, it is possible to model the environment of the vehicle with much more detail. But, this requires a more complicated measurement model or an elaborate preprocessing to handle the large amount of sensor data. However, the update and consequently the overall performance is improved if the measurement model is tuned or a more sophisticated preprocessing is performed. This thesis presents a full processing chain for high-resolution radar data from the raw sensor measurements up to the incorporation of processed object detections in the update step of a multi-object tracking algorithm.

The scientific contribution of this thesis mainly covers three topics that must be fulfilled. Firstly, the radar object detection system is able to detect objects in radar data exclusively. Neither the radar data from several measurement cycles, nor from multiple radar sensors, nor any other kind of information from other sources is fused. Secondly, the radar object detector directly process the radar target list as received from the sensor, without a preliminary transformation into another data format. Thirdly, the radar object detection system is real-time capable allowing this algorithm to be deployed in real-world applications for the environment perception.

The thesis is structured as follows. Initially, Chapter 2 introduces the fundamentals of radar and deep learning, as well as the basics and the state of the art for object detection and multi-object tracking. Afterwards, Chapter 3 proposes the novel radar object detection system which predicts object hypotheses exclusively in radar data, and describes how the object detections are processed in a multi-object tracking algorithm. Then, Chapter 4 discusses radar datasets which are used to evaluate the radar object detector. Subsequently, Chapter 5 presents a detailed evaluation of the radar object detection system. Finally, Chapter 6 briefly summarizes the contents of this thesis and gives a conclusion, as well as providing suggestions for future work.

Chapter 2

Fundamentals and State of the Art

This chapter presents the basic principles that are essential for comprehending the contents of the thesis. In addition to the fundamentals, this chapter also describes the state of the art in the corresponding sections. This means that for each topic, the fundamentals as well as the state of the art are described in the same section. Subsequently, this thesis distinguishes itself from the existing methods and briefly describes how it is solved in the thesis. Section 2.1 introduces the fundamental knowledge about a high-resolution automotive radar sensor, as such a sensor is used for this thesis. Furthermore, this section describes the basic principles of deep learning to understand its application for object detection. Section 2.2 discusses the state of the art in object detection using radar data. Therefore, several existing approaches are presented and distinguished from those proposed in this thesis. Finally, Section 2.3 provides the basics for multi-object tracking using radar data.

2.1 Basic Principles

This section presents the basics of radio detection and ranging (radar) and the use of deep learning in a rather general way. Section 2.1.1 introduces the fundamentals of an automotive radar sensor. This includes the measured variables of a radar sensor, which are used as input data for the radar object detection system proposed in this thesis. Further, a common processing chain of an automotive radar sensor, and its possible different output data, are described. Due to the fact, that the proposed radar object detector is based on a deep learning approach, Section 2.1.2 covers the fundamentals of deep learning. In the course of this, the differences between artificial intelligence, machine learning and deep learning are disclosed. Moreover, the section explains the classification and regression task, and deals with the advantage of neuronal networks. Finally, the idea of optimizing model parameters of a network during training and a common technique to achieve this goal are outlined as well.

2.1.1 Radar Basics

The importance of automotive radar sensors in modern vehicles for environment perception is constantly increasing. Admittedly, a radar sensor does not achieve same accuracy in the near range as a light detection and ranging (lidar) or camera sensor, but it has several other advantages. Firstly, high-resoultion automotive radar sensors are capable of detecting objects in a distance up to 250m [Con] and due to multipath propagation, it is also possible to recognize objects that are moving one behind the other and are partially obscured. Secondly, in comparison to lidar sensors, a radar sensor is extremely economical in terms of cost effectiveness [Con]. Thirdly, most lidar sensors only operate impeccable under good weather conditions, and cameras do not perform well in the dark. In contrast, a radar sensor operates roughly the same in the light and dark, as well as in all weather conditions, even in the rain, fog and snow [Con]. Last but not least, a radar sensor is the only one capable of directly measuring a velocity information of an object. Hence, a radar provides distance as well as velocity information [Con], where the velocity is a radial velocity. The following subsection only addresses the parts of the technology of a high-resolution automotive radar sensor, which are essential for this thesis. Extensive information about radar in general are provided in [Sko08; ST13; Wol98], and more details concerning the application of automotive radar systems is given in [WHLS15].

Measured Variables

Basically, an automotive radar sensor is able to directly measure the range and the azimuth angle in polar coordinates, as well as the radial velocity and reflection power of a target. Section 4.1 presents the radar sensor which used in this thesis. However, this automotive radar sensors does not provide the mentioned measured variables, instead the spatial information is given in Cartesian coordinates and the Radar Cross Section (RCS) is derived from the reflective power. Accordingly, the actually measured signals are processed by the sensor itself. Since it is not possible to influence this preprocessing step for the applied radar sensor, the measured variables are used in this thesis exactly in the format in which the radar sensor provides them. After the internal transformation of range and azimuth from polar coordinates to Cartesian coordinates, the position of a radar target is specified by an x and y value.

An automotive radar sensor is able to determine the radial velocity of a moving target by exploiting the Doppler effect. An electromagnetic wave undergoes a frequency shift whenever an observer and a transmitter move relative to each other. The same phenomenon occurs when a radar beam is reflected from the object of interest which is moving relative to the radar sensor. Consequently, the radial velocity v_r is the

velocity part of a target that acts towards or away from the radar sensor. Since the measured radial velocity is the relative velocity of a target, any motion of a radar sensor affects the resulting radial velocity. With the purpose of obtaining the objects' actual radial velocity, the motion of a radar sensor itself has to be extracted. This technique is called ego motion compensation. The derivation and associated formulas for the ego motion compensation are presented in [Sch19]. As a result, the parameter $\tilde{v}_r$ describes the ego motion compensated radial velocity of a radar target.

In order to describe the reflectivity of a target in terms of a quantity, the RCS is derived from the reflection power. The RCS value σ strongly correlates with the target to be measured and varies depending on the shape of the target, the material properties, the wavelength as well as the angle of incidence and angle of reflection of the beam. In general, the higher the RCS value for an object is, the better this object can be detected with a radar sensor. In summary, all measured variables that are provided by the automotive radar sensor from Section 4.1 are now introduced. Moreover, these are exactly the measured variables which are the input radar data of the radar object detection system, which is proposed in Section 3.1 of this thesis.

Signal Processing

In order to extract measurement information about a real object from a reflected electromagnetic wave, an automotive radar sensor performs a specific signal processing. Figure 2.1 presents the individual modules the radar signal processing chain of a common automotive radar sensor as presented in [WHLS15]. The processing pipeline is structured in a way that it is generally applicable and independent of a particular radar design, for example modulation type or antenna concept. Below, all components of the radar signal processing are explained, since it corresponds to the chain which is applied in the radar sensor deployed in this thesis. Comprehensive information on the fundamentals of radar signal processing can be found in [Ric14].

The functionality of a radar is based on emitting and receiving electromagnetic waves, however these are only utilized as carriers to transmit information modulated on a signal. The signal modulation is part of *signal shaping* component. In this module the signals are generated, so there is no input data as shown in Figure 2.1, but in fact it is the initial step in the radar signal processing. There are different concepts to modulate information on amplitude or frequency of a signal by mixing signals, for example modulating information on a sinus signal, or using pulse modulation. Automotive radar sensors perform frequency modulation, which varies a carrier frequency in the bandwidth of $76 - 77$GHz over time to transmit information. A technique for frequency modulation is the Frequency Shift Keying (FSK), which modifies the current frequency of the signal in multiple steps over a specified time

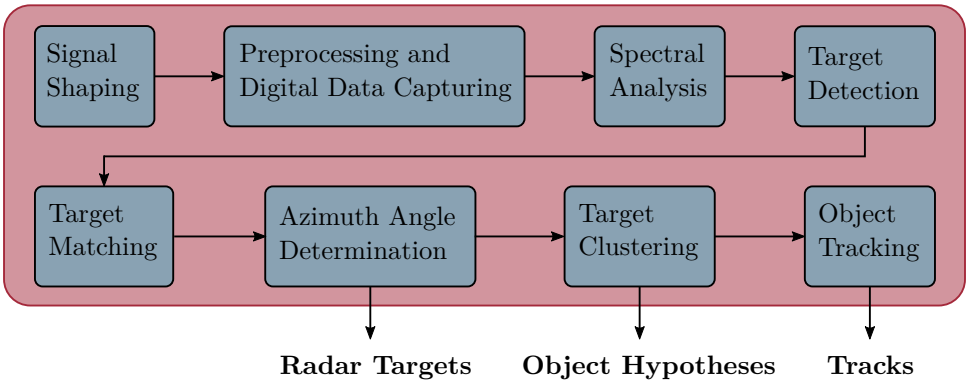


Figure 2.1: Overview of the radar signal processing: The transmitted radar is reflected by an object, and further processed to provide radar targets, object hypotheses or tracks. The image is drawn in reference to the overview of the radar signal processing in [WHLS15].

period. Further, the Multi Frequency Shift Keying (MFSK) is an extension of FSK, which uses more than two frequencies at the same time. In today's automotive radar sensors, the common techniques are the Frequency Modulated Continuous Wave (FMCW) and the Chirp Sequence (CS) modulation. In FMCW, the current frequency is changed continuously using various ramps with different slopes. The CS modulation is based on the principles of FMCW, but it sequentially transmits multiple linear frequency ramps centered, which are realized as single upchirp signals of short duration, around the carrier frequency. More details on the topic of radar signal modulation is given in [Far17], and for automotive radar sensors in [WHLS15].

Whenever the emitted signal hits an object, it is reflected and returned to the receiver unit of the radar. The *preprocessing and digital data capturing* module performs a demodulation and amplification of the received signals and converts the analog signals into digital data by sampling. Then, the *sepctral analysis* applies a Fourier Transform to turn the signal from the time domain into the frequency domain. Practically this is realized on radar sensors using the Fast Fourier Transform algorithm [CT65]. The resulting frequency spectrum contains information about range, radial velocity and azimuth angle, which are extracted by the following units.

The next step is to extract a list of *radar targets*, which comprises the detection and matching to eventually determine the explicit values for range, radial velocity and azimuth angle for each radar target. The *target detection* examines the measured data for specific characteristics that appear as peaks in the spectrum. The goal is to

find all peaks belonging to real objects. Often an adaptive threshold is applied for filtering purpose. This technique is also called Constant False Alarm Rate (CFAR) detection [Sko08]. Subsequently, the *target matching* assigns all detected peaks to an object. This includes the matching of different peaks in the same beam as well as peaks corresponding to several beams. Finally, the *azimuth angle determination* analyzes the amplitudes of peaks which are associated to the same target, but occur in different beams. Along with the knowledge of the antenna characteristics deployed in the radar sensor, this allows to identify the respective azimuth angle of a target.

Modern automotive radar sensors have a high resolution which allows to detect multiple radar targets for one object, especially at vehicles or trucks. Therefore, the *target clustering* groups all reflections that belong to the same object and treats them in the subsequent steps as one measurement, or, in other words as a hypothesis for an object. The clustering makes use of heuristic assumption to decide whether the measured quantities belong to the same object. In general, after the clustering process, each detected object is modeled as a single point target. Hence, the clustering module outouts a list that contains *object hypotheses*. Since it strongly depends on which part of an object is detected, the resulting radar targets are often rather different in radial velocity or RCS. This is because the measured relative velocity varies, for example, the wheels of a vehicle rotate faster than the actual velocity of the entire vehicle, or the limbs of a pedestrian move more rapidly than the rest of the body. Further, some parts of an object reflect better than others, for instance, the wheel housing or the license plate of a vehicle. However, the target clustering technique may turn out to be not straightforward and is therefore prone to errors.

With the focus to achieve a robust perception based on the clustered but still noisy targets, the *object tracking* unit filters this data over time. The object hypotheses constitute a snapshot of the current measurement cycle. Consequently, a tracking algorithm correlate previous object hypotheses with recent measurements. Primarily, the tracker predicts all captured objects to the actual measurement time. Subsequently, the generated object hypotheses are associated to the objects and the update corrects those using the measurement information. This algorithm produces *tracks*, which are objects described by their current dynamic states, for example, position, orientation and velocity, and additionally a history of previous states. This results in a smoothed trace which represents the motion of objects up to the actual time.

The radar processing chain from Figure 2.1 reveals that an automotive radar sensor is able to generate various outputs on different modeling levels. The radar sensor produce measurement data on a target level, which are called *radar targets*. While the results at an object level are provided as *object hypotheses* and *tracks*. The algorithms for generating radar targets, object hypotheses or track are already implemented on commercial sensors. The radar sensor presented in Section 4.1 outputs a radar target list where each target comprises four dimensions: the position

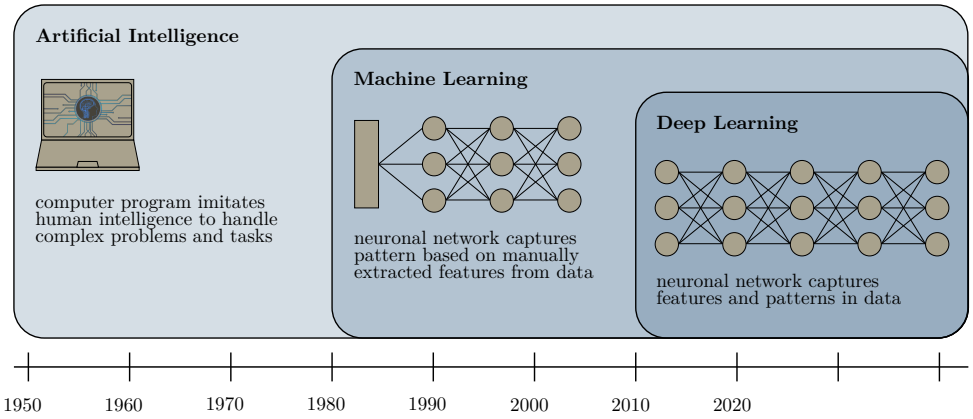


Figure 2.2: Categorization of artificial intelligence: Distinction between the general artificial intelligence, machine learning and ultimately deep learning. The image is redrawn and inspired by [Cop16].

in cartesian coordinates, the radial velocity and the RCS value. Since the algorithm that generates the radar target list is kept secret by the manufacturer, it is not possible to influence this processing chain and its output. However, algorithms for object detection and tracking based on the radar target list can be developed. Section 3.1 proposes a novel algorithm to detect objects in radar data processing an entire radar target list. This results in object hypotheses or in object detection for each radar measurement cycle. Thus, this technique replaces the *target clustering* module in Figure 2.1 above. Further, Section 3.3 presents a measurement model to process these object detections in a multi-object tracking algorithm. Eventually, the motion of dynamic objects is modeled by the obtained tracks, and this represents the environment at an object level. Hence, this thesis replaces the *object tracking* module in Figure 2.1 with a comprehensible and well-established object tracking system.

2.1.2 Deep Learning

The science of *artificial intelligence* deals with the replication of intelligent human behavior by a computer system. There are a number of definitions in literature, which may differ in some aspects, but essentially make the same statement. In [Wit19], a central aspect for an artificial intelligence system is depicted as “the attempt to develop a system that can independently handle complex problems”. This definition describes the basic idea of artificial intelligence quite well, but it is rather

general. A more detailed description of the topic closely depends on the specific research field. A categorization of artificial intelligence that is appropriate for this thesis is given in [Cop16; GBC16]. The presented division is based on subcategories which are fully enclosed by those above them to clarify their relationship. That is, *machine learning* as a part of artificial intelligence, and *deep learning* that in turn is a subset of machine learning. Figure 2.2 visualizes this categorization along with a chronological order to indicate the time when the particular topics came into focus.

In general, artificial intelligence is based on the idea that computers solve tasks in a sophisticated way and includes anything that addresses the issue of machines behaving like humans. In [Cop16], it is summarized as "artificial intelligence - human intelligence exhibited by machines". Further, in machine learning a large amount of data is analyzed by algorithms to learn from it and apply what it has been gained to make further predictions on unseen data. Traditional machine learning methods are not applicable on raw input data, but preprocessing of the data is required. This procedure is called feature extraction, which means that humans manually generate features in order to feed an algorithm, which may be a neuronal network, to produce predictions. Consequently, the computer learns an intelligent behavior based on extracted features, or as formulated in [Cop16], "machine learning - an approach to achieve artificial intelligence". Last but not least, deep learning extends the concept of machine learning to achieve better results. For this purpose, a deep neuronal network is trained using labeled data, where in contrast to machine learning algorithms, the feature extraction is part of the neuronal network. This means, a deep learning model processes the raw data and implicitly learns features for the following predictions. In summary, deep learning realizes machine learning, or in the words of [Cop16], "deep learning - a technique for implementing machine learning". This thesis applies deep learning methods to identify different objects in radar data.

Machine Learning

As already mentioned, deep learning is a particular type of machine learning. More information about these topics and important definitions of machine learning terms are provided in [Mit97]. In the following, only some basic principles are introduced. The goal of machine learning algorithms is to solve a specific task by predicting an output based on the input data. Therefore, the algorithm analyzes the data to find particular patterns in it and to use them for further predictions on unseen data. In order to automate this process, a supervised learning approach is suitable, that means, the algorithm automatically learns an appropriate model from training examples. Such a training sample comprises the input data as a vector $\underline{x} \in \mathbb{R}^n$, where each entry represents a single feature, and a label that is the desired output. Therefore, the objective during training is to find mapping from input data to labels.

Classification

The classification is a machine learning technique for predicting a class membership to specify which category an input belongs to. The objective of this task is to find a function $f : \mathbb{R}^n \to \{1, \ldots, k\}$ which maps an input to a specific category k, that is

$$y = f(\underline{x}). \tag{2.1}$$

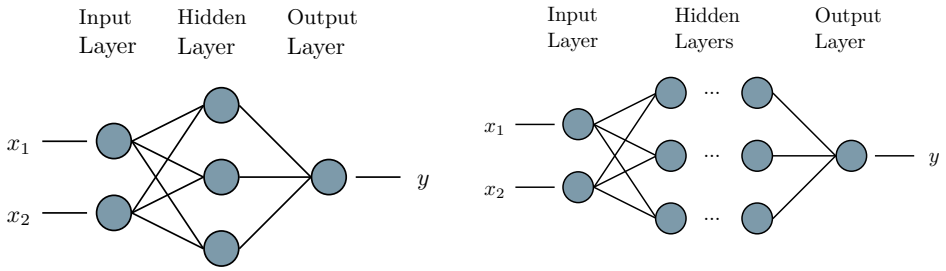
The vector $\underline{x}$ describes the input data, whereas the output y is a discrete value that represents one of the different categories or also called classes. After learning a sufficiently well performing model based on the training data, the classification algorithm actually predicts a score that is converted to a probability value and indicates how likely the assignment of input data to respective classes is. In the simple binary case, the model only distinguishes between two classes, this is called *logistic regression*. Then, the predicted output of the network is forced into the range between 0 and 1, for example by a *sigmoid function* which is also known as the *logistic function*. In the multi-class case, the model outputs values for each of the k categories. After appyling a *softmax function*, which is a generalization of the logistic function to multiple dimensions, the neuronal network output is normalized such that the sum over all class probabilities equals to one. Then, the category with maximum class probability value may be selected as the final classification result.

Regression

The regression is a machine learning method to predict continuous values based on the input data. In contrast to classification, the sought function of this task maps the input to a real number, thus $f : \mathbb{R}^n \to \mathbb{R}$. In fact, regression and classification are similar, the only difference is the output format, which is continuous for regression and discrete for classification. In the simplest case, with two-dimensional data, linear regression is performed. That means, the learning algorithms fits a straight line to represent all data points. Consequently, the model applies a linear predictor function to determine a dependent output value on the basis of independent input variables.

Deep Learning with Neuronal Networks

Modern neuronal networks allow to capture different patterns in data, either linear or non-linear. Therefore, the model of a network, that describes the mapping function, consists of multiple layers with a large amount of parameters. Figure 2.3a shows a simple neuronal network with three layers: an input layer, one hidden layer and an



(a) Network with a single hidden layer. (b) Network with multiple hidden layers.

Figure 2.3: Both neuronal networks forward an input feature vector and process them using fully connected layers to get the output result. The network on the left consists of a single hidden layer with three parameters. Whereas the architecture on the right belongs to a deep neuronal network with any number of hidden layers.

output layer. This network expects an input vector $\underline{x} = [x_1, x_2]^{\mathrm{T}}$ with two features x_1 and x_2. The hidden layer represents the function f with three parameters for mapping the input features to the single output y. The network architecture may be freely designed, so that it is even possible to cover complex functions f with a lot of parameters. In order to realize deep learning, the network is composed of many hidden layers, thus it becomes a deep neuronal network. Figure 2.3b visualizes such a deep network with the same input and output layers as in the previous example, but with several hidden layers. That architecture has much more parameters, accordingly, the model is capable to recognize even highly complex patterns. However, deep learning needs a lot of training data to learn optimal values for all parameters inside the network architecture. Furthermore, the training process is computationally quite complex and requires much computing power. Nevertheless, modern computers with high-performance Graphics Processing Units (GPU) are able to cope with that, making deep learning techniques become reality and indispensable for many tasks.

Optimization

After designing a deep learning network, the objective is to find optimal parameters for the model. Therefore, a training process is necessary, that is an iterative procedure to optimize the parameters based on training data. In order to measure the prediction quality of the deep learning algorithm, a function, called *cost function* or *loss function*, is applied. The loss function measures how well the predictions of a model match the associated labels. The goal during training is to iteratively minimize the loss function and to identify the global minimum that constitutes the optimal solution based on

training samples. A common technique to solve the optimization problem is gradient descent [Cau47]. This method determines a minimum by progressively moving closer to the lowest point using the derivation of the considered loss function. However, it is not guaranteed that gradient descent finds the global minimum. It depends on many factors, for example, the kind of loss function, the deep learning algorithm and the quality of data. It may even happen that gradient descent gets stuck in a local minimum. A local minimum is the lowermost point in a particular neighborhood, but not the absolute minimum point of the entire loss function. Hence, it is possible that several local minima exist. In deep learning the objective is to find a value for the loss function, which is very low. Nevertheless, it does not necessarily have to be the global minimum, because already a local minimum may provide sufficiently well predictions to solve the defined problem. For performing gradient descent, literature offers some existing algorithms, where [Rud16] summarizes some established ones.

2.2 Object Detection

The term *object recognition* generally refers to the task of identifying objects in data, for example radar data. Object recognition can be categorized into related but different tasks, these include object classification, object localization and object detection. These terms are now explained in relation to radar data, that is, a radar target list. During the *object classification*, the class of a single object in radar data is assigned. This implies that the input is a radar target list in which exactly one object is present. The task is to predict a class for this object. In contrast, the *object localization* does not classify objects, but identifies the location of all objects in the radar data. The input is a radar target list with one or more objects. And the task is to locate all present objects and mark them with a bounding box. As a result, the output is a list with one or several bounding boxes representing the location of all objects. The task of *object detection* combines the challenges of classification and localization. This means the goal is to localize as well as classify all objects by indicating the existing objects with a bounding box and assigning a class to each of them. Once again, the input radar target list contains one or multiple objects in it. Hence, the detection output is a list with one or more bounding boxes, as well as an associated class label for each bounding box. This thesis focuses on object detection exclusively in radar data generated by a high-resolution automotive radar sensor.

As mentioned in Section 2.1.1, a high-resolution automotive radar sensor represents the environment using radar target lists. Common automotive sensors measure range for the distance and azimuth for the direction to provide a 2D spatial information for each radar target. However, if the elevation angle is measured as well, the radar sensor is able to generate 3D spatial information. After processing, the radar sensor

provides its captured data as a radar target list containing a variable number of unordered targets. The radar target list in this thesis comprise the 2D position, the radial velocity and the RCS value for each radar target. Since, the data does not show an ordered and regular format, the technique of this thesis allows to process an unordered radar target list by applying a deep learning method for point sets.

There are already a couple of concepts for object recognition which process data generated by an automotive radar sensor. Most methods in literature classify single radar targets or clusters comprising multiple targets of an object, meaning that these approaches only perform object classification. But some methods focus on object localization by estimation bounding boxes for specifics objects. However, few methods are presented that perform object detection using only radar data. This thesis presents a technique for object detection using data from a single radar sensor.

The methods for object recognition presented in the next section differ in their strategy and can be categorized into classical methods, learning algorithms with manually extracted features and deep learning approaches. A classical way is to design engineered features and analyze them using algorithms established or developed for this purpose. Another approach is to generate features manually and forward them to a machine learning algorithm. Consequently, it is possible to learn a model based on features extracted from sufficient training data, which solves the respective object recognition task. Since it is sometimes difficult to design appropriate features, deep learning networks are designed which allow automatic learning of features solely based on training data. These self-learned features are applied directly within the network, and ultimately, a model that solves the object recognition task is learned.

2.2.1 Classical Approaches

In [RKK+15], an approach to estimate the vehicle orientation based on radar data is presented. Therefore, the Orientated Bounding Box (OBB) algorithm [Tou83] is enhanced for radar targets and adapted with a quality function. Since an object generates multiple radar reflections, a preprocessing step clusters all input targets using the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [EKSX96]. For each group representing a potential object, a vehicle model is fitted in form of a rectangle. Then, this method determines the convex hulls for every cluster and sets up a basic rectangle, which is rotated to evaluate several orientations. In order to handle clutter measurements, which are measured radar targets that do not belong to the object of interest, any target is left out multiple times to construct different convex hulls resulting in various basic rectangles. Then, again the algorithm rotates each of them to take all sides into account. Finally, the quality function rates all rectangles in consideration of different factors to find a

solution. The evaluation in [RKDW16] compares the enhanced OBB [RKK+15] with other methods. Firstly, as the standard OBB is very sensitive to clutter radar measurements, it is extended with a quality function to compensate them in the bounding box fitting algorithm. Secondly, using the *L*-fit algorithm [JCYK08], the vehicle is modeled with two perpendicular lines. Thirdly, since a radar sensors also captures reflections under the vehicle, a brute-force approach is applied to consider all these measurements. Therefore, the Random Sample Consensus (RANSAC) algorithm [FB81] determines a strong distinctive contour of the vehicle to produce a basic rectangle. Subsequently, a quality function rates different rectangles which are obtained by some variations of the basic contour. This approach produces the best possible estimate results, but is computationally very expensive. The evaluation examines the error of orientation for all algorithms, the dependency of estimation on aspect angle and distance to the object of interest, as well as the extraction of its dimension. Furthermore, it is investigated how the results for the orientation estimation change when data generated from two radar sensors are used. It turns out that the results of all investigated algorithms are more accurate regarding the orientation error metric when targets of both sensors are merged to create a denser radar target list than the result for data which is generated by a single radar sensor.

Before applying an algorithm for bounding box fitting, it is necessary to group all radar targets which belong to the objects of interest. Clustering methods are performed during this preprocessing, for example, the algorithm DBSCAN or RANSAC. Since these standard clustering algorithms are limited in their ability to consider the specific characteristics of radar data, [SRKW16] proposes an adaptive clustering method called KNN-DBSCAN algorithm. The clustering procedure is based on a k-nearest-neighbours distance examination, the basic DBSCAN algorithm and a combination of overlapping clusters to find the best possible cluster comprising all radar targets originated from a vehicle. After grouping the radar data, the method estimates an object contour by fitting an orientated bounding box of minimum area including all radar targets within a cluster. The proposed evaluation examines different parameter sets for the clustering algorithm and compares the resulting estimated bounding boxes. It turns out that in situations where limited measurements for the object of interest are present, the adaptive clustering approach finds clusters at all. Generally, it is essential to identify a cluster before the contour estimate is possible.

In [SSSW17], an alternative method for contour estimation to the OBB approach is presented. Similar to [RKK+15], the goal is to estimate an orientated bounding box using radar data. The method performs a Generalized Hough Transform [Bal81] analysis and template fitting to match the bounding box for a vehicle. The orientation and position is determined in multiple iterations, where a quality function investigates each parameter set to find the most probable solution for the template. The analysis shows that in most situations this approach achieves similar results as the OBB method. In challenging situations, for example when few radar reflections for a

vehicle or clutter measurements are given, the OBB algorithm can not fit a bounding box. Whereas the template matching algorithm [SSSW17] performs better in some of those cases. However at the expense of a much higher computational complexity.

2.2.2 Learning Methods

In the classical methods for object recognition in radar data, the entire algorithm, which comprises all calculation rules, is hand-designed. In contrast, learning algorithms are capable of automatically finding a function that maps input data to an output. As already discussed in Section 2.1.2, machine learning and deep learning differ from each other. In machine learning, the input of a model are crafted features that are generated in a previous stage. Afterwards the network predicts an output according to this feature vector. In contrast, deep learning algorithms are able to learn such features by themselves. As a consequence, the raw input data is forwarded to the model, where the neuronal network primarily extracts features and predicts the output based on these self-learned features. This may offer major advantages, since the automated analysis of training data usually results in better or even not considered features, meaning the subsequent predictions are significantly improved.

Machine Learning

In [HR10], a recognition method for pedestrians and vehicles using a 24GHz automotive radar sensor is proposed. The object recognition system deploys different signal features for the classification of measured radar targets. Therefore, three features are extracted: the received amplitude, the range profile and the Doppler spectrum. The amplitude depends mainly on the RCS and differs clearly for pedestrians and vehicles. The range profile describes the size and is used to distinguish between extended and point shaped objects. A vehicle has a static range profile as its extent is constant at any time. Whereas the range profile of a pedestrian changes over time, since the arms and legs are in motion which define the range extension. The Doppler spectrum of a vehicle is constant as long as it is moving at the same speed. A pedestrian has a changing Doppler spectrum, because the torso, arms and legs show different velocities over time. Since a pedestrian has a varying range profile and Doppler spectrum, the point shaped range profile or the maximum spread in the Doppler spectrum may only be measured at certain times. For this reason, a stochastic approach is applied by observing multiple consecutive measurements and a feature vector is built by selecting the maximum values per feature considered over time. The approach is evaluated using simulated radar signals for pedestrians and vehicles.

Subsequently, a Support Vector Machine (SVM) [CV95] is applied to classify the radar targets into pedestrian and vehicles based on the extracted feature vectors.

The object recognition system in [HR11] consist of two stages to classify objects in real-world data generated by an automotive radar sensor. The classification process distinguishes between pedestrians and vehicles using suitable feature vectors. Similar to the work in [HR10], the recognition model is based on the radar signal characteristics of both object classes, the range profile and the Doppler spectrum. In the case of a pedestrian, the range profile is point shaped, but the Doppler spectrum is extended caused by different measured velocities for the trunk, arms and legs. Hence, in a first step, the range profile and the Doppler spectrum are extracted from the radar echo signal as features. An SVM classifier processes these input features to differentiate between pedestrians and vehicles. In order to confirm the classification result, the classifier output is forwarded into a tracking system. In this second step, a Joint Probabilistic Data Association (JPDA) filter, which is able to handle scenarios with multiple objects, verifies the potential tracks using consecutive measurements. Subsequently, additional features based on the tracking results are extracted. Since a Kalman filter is the basis of a JPDA tracker, the extraction module provides the process noise matrix and the Kalman gain as further features, which are different for pedestrians and vehicles. The range and velocity of a pedestrian varies in their radar measurements, resulting in a large process noise and Kalman gain. Whereas a vehicle generates similar radar reflections over time, as a consequence, the process noise as well as the Kalman gain are small. Then, the extracted features are fed back into the classifier to finally make a decision regarding the class of the object.

The system for object recognition proposed in [HR12] processes several radar features with the main focus on distinguishing between pedestrians and lateral moving vehicles. The objective is to classify measurements of a 24GHz automotive radar sensor into four classes, that is longitudinally and laterally moving vehicles, as well as pedestrians and other objects. In contrast to the object classification process in [HR11], the tracking module and the feature extraction based on it, are omitted, and only applies an SVM classifier. Therefore, a feature vector based on range and velocity profile is extracted. Since a radar sensor measures different velocities for certain parts of a moving pedestrian, such as the torso or arms and legs, the resulting velocity profile is extended. While the extent of a pedestrian is small, the range profile is point shaped. In the case of vehicles, it is important to distinguish between a longitudinal and lateral movement, because the range profile and velocity profile is different. In both situations, the range profile of a vehicle is extended, while the extension for a lateral moving vehicles is larger. Considering the velocity profile, a longitudinal moving vehicle shows a point shaped velocity profile, because the radial velocity at all reflection points is equal to the actual velocity. Due to the lateral movement of a vehicle regarding the radar sensor, the radial velocities vary on the object, consequently, the resulting velocity profile is extended. On the basis of these

profiles, various stochastic features are calculated, for example the radial velocity, as well as extension, variance and deviation of the radial velocity and the range, plus RCS values. Since numerous of pedestrians are misclassified as lateral moving vehicles, some more features are calculated to improve especially those classification results. These additional features are based on the longitudinal and lateral velocity component of an object, which can be calculated using the measured radial velocity and the corresponding azimuth angle of the radar target. The object classification system is evaluated on a labeled dataset containing real-world radar measurements.

All object classification systems proposed in [HR10; HR11; HR12] process measurements from a 24GHz radar sensor. Today, most automotive radar sensors use the 77GHz technology [WHLS15], which is more powerful. Such radar sensors enable measuring objects in near and far ranges with a significantly increased range resolution and a better separation of objects which are in close proximity to each other even at great distances. Further, the Doppler resolution of a 77GHz radar sensor is more accurate [WHLS15], which improves the measurement of the radial velocity. In contrast, the radar sensors with 24GHz technology are cheaper than those with the 77GHz technology [WHLS15]. As already mentioned, thesis performs object detection in radar targets which are generated by a 77GHz automotive radar sensor.

In [DHS+14], an object detector for parked vehicles based on radar data is presented. The detection system consists of four steps: occupancy grid generation, candidates selection, features extraction and classification. All previously described object classification methods directly process the information in radar measurements to create feature profiles using the range and radial velocity. Another option is preprocessing the radar measurements into a different representation and then extracting features for an object classifier. This approach accumulates the radar data over time by computing an occupancy grid map. This grid map considers the spatial information of the radar measurement to model the environment using cells with the same size. The candidates selection has to distinguish between two kinds of parked vehicles: cross-parked and parallel-parked vehicles. In order to detect potential objects, the grid map is transformed into a binary representation and a connected component analysis [PLC00] is applied to get clusters with objects of interest. Since a parked vehicle normally occupies several cells, the algorithm removes objects with less cells. The selection method normalizes each sample which comprises the occupied cell of an object using an estimated direction. Finally, the candidates for cross-parked and parallel-parked vehicles are extracted after analyzing the cutout map associated to an object. The feature extraction generates a feature vector for every candidate, which describes size and shape, occupancy information and local orientation. For classification, two random forest classifiers [Bre01; Ho95] are trained, one for cross-parked vehicles and one for parallel-parked vehicles. Next, each classifier divides the candidates into two classes, *vehicle* or *non-vehicle*. Finally, the results of both classifiers are combined using classification scores to extract one detection per object.

In order to group all radar targets which belong to the same object, [SHDW18b] proposes an approach to learn parameters for the DBSCAN algorithm. This means the search for the DBSCAN hyperparameters is based on machine learning, but the actual clustering then uses the classic DBSCAN algorithm. However, this is not a method to directly generate object detections, but only to assign radar targets to an object. The results in [SHDW18b] shows that radar target clustering using DBSCAN with learned parameters outperforms the standard DBSCAN algorithm. Nevertheless, the evaluation in [SHDW18b] also shows that there are still many situations in which the DBSCAN with learned parameters performs quite poorly.

In [NQ19], an approach is presented to propose regions of interest based on radar targets. For this purpose, the preprocessing transforms the radar targets into camera-view coordinates. Then, the regions of interest are determined using the concept of anchor boxes from [RHGS17]. In order to determine the scaling factor of the anchor boxes with respect to the distance, [NQ19] applies a scaling formula with learned parameters. Summarzing, the obtained proposals are not object detections, but only areas where potential objects are located. The idea of [NQ19] is that the region proposals are used in two-stage object detection networks, for example in Fast-RCNN [Gir15], to reduce the overall processing time since this method is fast.

Deep Learning

Most deep learning approaches on radar target lists, transform the input radar data into another representation with a regular format before applying deep neuronal networks. The advantage is that if the radar targets are converted into a format with regular structure, most of the familiar approaches from computer vision are applicable. Some examples for such transformations are voxelization, projection or rendering to feed the data into a two-dimensional or three-dimensional Convolutional Neural Network (CNN) [LB98], or perform feature extraction to use fully connected networks. Modern lidar sensors provide besides the position measurement also an intensity value of the reflection that is convertible into an intensity image. This image has a fixed format and can be forwarded into a classical CNN. A common method to achieve a regular structure for radar data is the transformation into grid maps. For this purpose, the environment is approximated by a rectangular grid with squared cells of a specific size. Each cell contains certain values to represent the information of radar reflections by accumulating the data over time. Since common automotive radar sensors generate only a few reflections per measurement cycle, the accumulation of multiple measurements, overcome the problem of sparse radar data. In [WRK+15], two methods to generate a grid map based on radar data are introduced. The amplitude grid map uses the RCS information to calculate cell values resulting in a grid map which indicates the reflection characteristics of objects.

The cell values of the occupancy grid map represent the probability of whether a cell is occupied or free using the spatial coordinates of radar reflections. Since the amplitude grid map looks blurred but directly identifies the RCS of radar targets, it is more appropriate in an environment with few objects to recognize the different reflection properties. Whereas the occupancy-based approach results in a grid with sharp contours and is well suited for an environment with multiple objects. That grid map can be applied for position estimation of objects and freespace estimation, or even classification purposes based on the object contours. Summarized, both grid mapping approaches adequately model the environment and are applied for self-localization tasks. The algorithm only processes the measured radial velocity to remove radar targets belonging to moving objects, however, not for the generation of radar grid maps. This information should not be neglected, especially for the classification or detection of objects, as the radial velocity is a very strong feature.

In [LHDW15], an object detector for arbitrarily rotated parked vehicles on the basis of radar grid maps is presented. Therefore, the radar data is preprocessed by creating an occupancy grid map using the spatial information being the range and azimuth angle, and an amplitude grid map using the RCS of radar measurements. However, both grid mapping approaches do not consider the radial velocity, although it is valuable information. The object proposal extracts a list of objects using a connected component analysis on the occupancy grid map. Then two snippets per object are generated on the basis of occupancy and amplitude grid map. Since the map snippets have a fixed size similar to an image, the classification can utilize a Deep Neuronal Network (DNN). The classifier, which is a CNN, uses these map snippets as input to distinguish between the two classes *vehicle* and *non-vehicle*. The dataset for training of the network consists of labeled radar grid map snippets with processed radar data. Since the dataset is very small, data augmentation is applied by rotating each object sample and adding random noise to get more training data.

The results of [LHDW15] already confirm that deep learning methods are suitable for classification purpose of vehicles in grid maps based on radar data. The study in [LHDW16] demonstrates how to distinguish between multiple static objects in radar grids. Again, proposals for objects are extracted using a occupancy grid map. The classifier is a DNN which divides the map samples into different static classes, such as *building*, *car*, *curbstone*, *fence*, and more, as well as the class *other*. For the evaluation, an one-vs.-all classifier for each class and some combination of multiple classes are trained. In contrast to the classification system in [LHDW15], the size of the radar map snippets are slightly larger, with 8m $\times$ 8m instead of 6m $\times$ 6m, and the dataset consists of more examples. The results show that deep learning using radar grid maps is an appropriate method for classification of static objects. Since the division of the static class is too specific in some parts, the classifier does not provide a good performance for all classes. In comparison to that, the classification of vehicles, that are map snippets with the label *car*, achieves the best results. However,

it should be taken into account that the dataset is imbalanced and contains much more samples of the *car* class, which causes this classification result to be expected.

The object classification method in [LHDW17] applies a hybrid approach consisting of two different classifiers for static objects. In a preprocessing step, an occupancy and amplitude grid map is built using accumulated radar data. Similar to previous methods, the measured radial velocity is only used to filter out dynamic objects, but not as a input for generating the radar grid maps. A difference to [LHDW15] and [LHDW16] is that the radar map snippets are chosen significantly larger with a size of 15m $\times$ 15m to ensure that it contains enough information about the object of interest. The method uses a random forest classifier [Bre01; Ho95] and a deep CNN separately to classify the samples, and combines the respective results afterwards. For the random forest classifier, the feature extracting generates a couple of features based on clusters resulting from a connected component analysis, and also various features extracted from the occupancy and the amplitude grid map. The CNN processes an entire map snippet to predict a class probability for the object located in the radar grid. Both classifiers are trained to divide the object proposals into seven categories, which for example include the classes *car*, *building* and *other*. Compared to [LHDW16], that dataset consists of slightly more labeled radar samples, but is imbalanced even worse, because there are significantly more examples of the class *car* and *other*. The evaluation of the single classifiers shows that for the *car* category, the random forest classifier performs better than the CNN. Further, the ensemble classifier which combines the output of both classifiers, achieves improved results.

The goal of [SWHD17] is to classify multiple dynamic objects using features extracted from radar measurements. Therefore, two different methods are compared, a random forest classifier and a Long Short-Term Memory (LSTM) network [HS97]. Although LSTM belongs to deep learning techniques and is able to learn features by itself, the proposed method extracts features manually. In contrast, random forest requires generated features. It is assumed that then the comparison of both approaches is possible. For feature generation, the radar targets are clustered using an adapted version of the DBSCAN algorithm to consider the spatial information as well as the measured radial velocities. Subsequently, a feature vector comprising 34 features which are based on the range, ego motion compensated radial velocity, azimuth angle and RCS, is extracted for every cluster. The random forest classifier directly process the feature vector of a particular time. Since a LSTM network is capable of learning long-term dependencies, the input has to be a time ordered sequence, which in this case is a group of multiple consecutive feature vectors. Moreover, the advantage of this neuronal network is, that it is able to identify correlations between input feature vectors. Furthermore, both classifiers are trained to distinguish between different dynamic classes, namely *car*, *pedestrian* and *pedestrian group*, *bike*, *truck* and *garbage*. The LSTM network performs slightly better in the prediction of almost all classes than the random forest approach. Both classifiers achieve the best classification

results for categories *car*, *pedestrian group* and *garbage*. It is expected that with still more training data the prediction results of the LSTM classifier will improve further.

The classification approach presented in [LLH+17] is different from the previous methods, because it performs a classification of static objects cell-wise instead of object-wise. The objective is to divide cells of a radar grid map into the classes *car* and *non-car*. Before building an occupancy grid map, radar reflections that belong to dynamic objects are removed by using the captured radial velocity. The classifier is a CNN with multiple layers and processes radar grids with a specific size to predict a class probability value for each grid cell. For the extraction of objects on the basis of the resulting semantic representation, the method applies several thresholds to decide which cells belong to a vehicle. In order to analyze the network architecture, the evaluation compares results for a varying number of convolutional layers in the network. The size of the input grids depends on the number of layers, thus, the input radar grids have to be imperatively smaller when fewer layers are present in the network. The evaluation reveals that the larger a radar grid is, and consequently more convolutional layers are required, the better the final classification results are.

There are many approaches in the literature that combine radar targets with data from other sensors to perform object detection. In [MK19b], an approach is suggested to use radar targets and a camera image for 3D object detection. Therefore, the radar targets are transformed into a birds eye view image. In this process, [MK19b] discards the measured radial velocity. Next, region proposal are generated based on the transformed radar data in combination with the camera image. Finally, the proposals are used to predict 3D bounding boxes which model a detected object. A similar approach is used by [NGW+19] to improve object detection. For this purpose, [NGW+19] also fuses the radar data with a camera image. In a preprocessing step, the radar targets are projected into the image plane. However, [NGW+19] discards the radial velocity of radar targets during this process as well. In order to make the input radar target list more dense, [NGW+19] even combines radar targets from three radar sensors. For the object detection, the transformed radar data and the camera image are given into a neuronal network based on RetinaNet [LGG+17] to predict 2D bounding boxes with corresponding classification scores for each object. In [YVB20], radar targets are fused with an RGB camera images. Therefore, the radar data is transformed into camera coordinates to generate a radar feature map. Again, only the spatial information of the radar targets is utilized. Then, the features are extracted in a network which combines the transformed radar data and the camera image. In order to achieve a robust object detection, [YVB20] adapts a Faster R-CNN [RHGS17] by the radar camera fusion network. In summary, a disadvantage of all these described methods is that the radial velocity information provided by a radar sensor is not used for object detection, although this has a major impact. A review with more information on radar based fusion approaches is given in [TZQ21].

In contrast to the above approaches, [YGCU20] process the measured radial velocity for object detection. However, [YGCU20] also focuses on the fusion of radar data and fuses the radar targets with a lidar point cloud. In order to combine the radar targets with the lidar point cloud, both data formats are converted into a voxel representation. For the detection of dynamic objects, the transformed radar and lidar information is forwarded to a neuronal network which is based on PnPNet [LYZ+20].

Indeed, some other approaches in literature apply deep learning solely based on radar targets. In [PDKG20], a method is proposed to perform object classification based on a 3D radar cube. Therefore, the radar targets are transformed into the radar cube representation, where a CNN predicts the class for each radar target. In order to obtain object proposals, [PDKG20] clusters classified radar targets using the DBSCAN algorithm. In [DEBK20], 2D car detection is performed on radar targets. For this purpose, the radar target list is transformed into a grid-based structure. Then, a CNN based on YOLOv3 [Red18] is applied to predict the final 2D bounding boxes. For comparison of the grid-based object detection approach with a point cloud-based object detection technique, [DEBK20] reimplemented the radar object detector from [DGBD19]. The work in [DGBD19] is the preliminary research for this thesis. The evaluation in [DEBK20] reveals, that the radar object detection system of [DGBD19] performs much better since fewer features are lost using this technique.

Although there are several promising results using deep learning on radar grid maps, the transformation into a grid map representation has its drawbacks. Firstly, even with a high-resolution radar sensor, the measured target list is still sparse. After the transformation of radar targets, most of the resulting grid map cells are empty. Hence, the transformed data representation is unnecessarily expanded compared to the original radar target list. Secondly, depending on the cell resolution, mapping the radar data into grid cells may produce quantization artifacts resulting in a loss of information. As a consequence, natural patterns of the original radar data may be lost, which may be useful for the neuronal network. Thirdly, radar sensors naturally generate a radar target list, thus, a set of targets without a specific order. The transformation of radar targets into a regular format causes that properties of an unordered target set are no longer fulfilled, for example, the invariance to permutations of targets. For all these reasons, it is preferable that a deep learning network directly processes the entire radar target list without any transformations.

2.2.3 PointNets

Some automotive sensors generate point clouds to represent environment information. A lidar sensor provides a dense point cloud which is very similar to the raw measurements. In contrast, a radar sensor applies some preprocessing steps to provide a

radar target list. As already described, one possibility is to convert the data into a regular format before using deep learning methods, such as grid maps for radar data. However, the data transformation may result in an unnecessarily voluminous representation. Considering radar grid maps, most of the grids are empty because the grid map is too large for the relatively few measurements. A further point is that a radar grid map quantizes the sensor data which may cause natural correlations in the raw measurements to be lost or even result in spurious coherence. Accordingly, the transformation may increase the complexity to find patterns in the data. Since a target list is simple and structured, it is desirable to directly forward the targets into a deep neuronal network and learn relevant features from the radar target list.

Deep Learning on Point Clouds

In [QSMG17], the PointNet, a new deep learning approach addressing the challenge to directly process point data, is proposed. The unified net architecture of PointNet allows deep end-to-end learning of features based on unordered and scattered point clouds without any conversion into another format. Such a point cloud is represented as a set which consists of multiple points. The dimensions are not limited to the 3D position, but points can comprise additional features such as color or intensity of the reflection. The PointNet framework is applicable for different 3D object recognition tasks, for example object classification, object part segmentation and semantic segmentation. In order to classify entire objects, the trained neuronal network predicts a classification score based on a point cloud which belongs to the object of interest. In contrast, for semantic segmentation, the model outputs a score for each individual point. Since PointNet deals with a complete point cloud, the neuronal network has to overcome several challenges which arise due to point data.

In the mathematical sense, a point cloud is an unordered set of points in an Euclidean space $\mathbb{R}^n$. Such a point set has to satisfy three main characteristics, which has direct consequences, for example, on an object detection algorithm. Firstly, points in the set are unordered. Unlike the pixels in an image or cells in a grid map, the points do not have a specific order. This implies that a point cloud which contains the same points but in different order, represents mathematically the same point set. Therefore, when forwarding a point cloud in a neuronal network, the model needs to be invariant to all permutations of the input set. Secondly, neighboring points interact among one another. As all points in a space with distance metric, nearby points are not independent of each other. Consequently, a neuronal network has to capture local structures from neighboring points as well as the resulting interaction between these points or even entire local structures. Thirdly, a point set is invariant under geometric transformations. Probably the most commonly occurring transformation in the context of an object detection algorithm is rotating

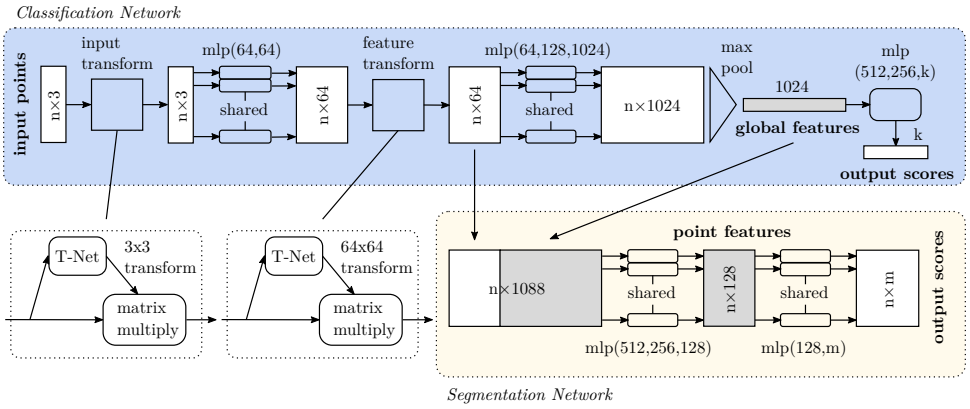


Figure 2.4: Network architecture of PointNet: All modules and deep learning layers in the classification and segmentation network to process a point cloud. The image is redrawn and inspired by [QSMG17].

and translating of points. This means, a rotation or translation of points belonging to an object must not change the output of an object detector regarding classification, point segmentation or extent of the object, but only its position and its orientation.

In Figure 2.4, the architecture of PointNet is visualized. PointNet consists of two large components, one network for classification of an object which is represented by the entire point cloud and one network for the segmentation of individual object points. The classification network expects a point cloud comprising 3D coordinates. In order to address the challenge of a point set having to be invariant under geometric transformations, the input transform aligns the point cloud using a Transformer PointNet (T-Net) and a matrix multiplication of the input data with the predicted transformation parameters. For alignment of the feature space, a second transformer network, the feature transform, is applied, by multiplying the transformation parameters from another T-Net with aggregated information from Multi-Layer Perceptron (MLP) to generate local features. The pointwise MLP in combination with the max-pooling layer ensures that PointNet is invariant regarding permutations of an input point cloud. This is achieved because the max-pooling operation is a symmetric function and always extracts same global features regardless of the point cloud's order. The classification probability for the whole point cloud is predicted by forwarding the global features into another MLP. The segmentation network extents the classification module to assign a class to each individual point. Therefore, the network extracts point features by concatenating local and global features, and then forwards them into pointwise MLP. Since the objective of semantic

segmentation is to assign a class to each point, a further pointwise MLP is applied to predict the classification probabilities for the respective points. This PointNet architecture allows object classification as well as semantic segmentation using deep learning directly on point sets without any explicit transformation on the input data.

The evaluation of PointNet is based on a detailed theoretical analysis and several experimental results [QSMG17]. The quantitative and qualitative results on object classification and semantic segmentation are convincing. One important outcome is that the PointNet model is resilient against data corruption in the input point cloud which may be caused by missing points or noisy points. When visualizing the point cloud which affects the extracted global features, it turns out that the network learns a global shape signature based on a critical point set from the whole point cloud. This means, missing non-critical points or noise points that do not affect the critical points, do not change results of the classification or segmentation network. It should be emphasized that a lidar point cloud is very dense, thus the critical point set is sufficiently large and especially the number of non-critical points is huge. For this reason, it is extremely unlikely that so many critical points will be corrupted, making the predictions of the network worthless. Since radar target lists are quite sparse compared to lidar data, it is interesting to explore the robustness of PointNet for such data. In this thesis, an object detector for radar data based on PointNet is developed, which is more challenging due to significantly fewer reflections per object.

The raw lidar or radar target list is a point set which is sampled from a distance metric space, thus a mathematical space with a defined distance metric. PointNet enables the extraction of a global signature from an unordered point set, but the neuronal network only learns either global or single-point features. Due to the structure of the architecture, PointNet is limited to capture detailed pattern in local areas and interactions among neighboring points which are caused by the distance metric in a point cloud. However, often such local structures are very useful for more reliable predictions of a neuronal network, especially regarding the generalization capability of a model for difficult or unknown scenes. The research of CNNs for object recognition shows that a deep hierarchical structure using a convolutional architecture facilitates to exploit local structures and to extract enhanced features based on such fine-grained patterns. A CNN processes regular data in form of a grid, such as a camera image or radar grid map, and applies multiple convolutions using different kernels in a hierarchical manner to capture detailed structures in the input data. The PointNet++ [QYSG17] addresses the challenge of deep hierarchical feature learning on unordered and unstructured point clouds. Therefore, the PointNet architecture is extended with a hierarchical structure to abstract differently sized regions to produce local features which are then further processed to generate high level features. Moreover, a new neuronal network called PointNet++, which is built around the hierarchical PointNet architecture to extract deep geometric features, is proposed. Therefore, PointNet++ groups the point cloud in multiple local regions

using various scales to finally combine features that are extracted at different scales.

The architecture for hierarchical feature learning on point clouds using Single Scale Grouping (SSG) is described in [QYSG17]. Therefore, the input points are grouped to extract hierarchical point set features in local regions which are continually becoming larger for every hierarchical level. The network comprises multiple set abstraction layers to process the point set of a local region using PointNet and produces aggregated information. The set abstraction layer consists of three sublayers: the sampling layer, the grouping layer and the PointNet layer. The sampling layer selects multiple points which represent the centroid of different local regions. Next, the grouping layer gathers neighboring points of that centroid and assigns these points to a corresponding local region. Finally, the PointNet layer applies a PointNet to generate a feature vector based on the local context in the respective region, which represents a new point set but with fewer points in it. By repeating the set abstraction process, the network progressively extracts features which gather more and more information from local regions, with the region growing in size with each step. This is a significant difference to the standard PointNet architecture that generates global features from the entire point cloud using a max-pooling operation.

PointNet architecture for hierarchical feature learning only uses SSG layers to capture local patterns in the point cloud of a group, where all groups have the same scale. In order to identify deep point features, PointNet++ architecture extends the abstraction level by extracting and combining features from various groups with different scales. For this reason, some density adaptive PointNet layers are added to the hierarchical network structure. In [QYSG17], two types for such layers are presented: the Multi-Scale Grouping (MSG) and Multi-Resolution Grouping (MRG). The MSG layer splits the input point set into multiple groups by using different scales. Then, it applies PointNet on the data to extract features for each scale, which are finally combined to a multi-scale feature vector. The MRG extracts features from multiple levels using different resolutions and concatenates them to a feature vector. The various levels summarize information from multiple set abstraction layers. This means when a feature vector consist of two levels, the first part comprises the processed information from a subregion of the lower level, and the other part extracts features directly from the raw point cloud in the local region. Both density adaptive layers capture local structures in the point cloud and aggregate information on the distribution of input points. The MSG layer is computationally more expensive than the MRG layer because it applies individual PointNets for every region to extract local features, which may be numerous groups, particularly for large scales regions.

The PointNet++ is a neuronal network which enables hierarchical deep learning directly on point clouds to learn deep point set features efficiently and robustly with regard to the distance metric space. Therefore, the network exploits neighboring regions to extract features which aggregate local structures and detailed patterns.

The evaluation in [QYSG17] shows that PointNet++ is applicable for classification and semantic segmentation purposes on point clouds. The results using this neuronal network are slightly better as compared to using PointNet from [QSMG17]. Overall, PointNet++ is a appropriate opportunity to imitate CNN behavior on point clouds without any data transformation, even when the computational complexity increases.

PointNets for Object Detection

In [QLW+18], the Frustum PointNets, a framework for 3D object detection based on an RGB image detector and multiple PointNets, is introduced. Although, this method allows the detection of objects using deep learning on point clouds, it still requires a camera detection which comprises an object of interest, as input. This framework extends the method for semantic segmentation on point sets based on PointNet and PointNet++ to perform object detection in several stages. Therefore, the camera detector is required for two purposes, first to find an object of interest and its class, and second to reduce the number of points for further processing. Then, several PointNets are applied for instance segmentation and estimation of a bounding box for each object. In reference to the definition of terms introduced above for object recognition, the image detector provides the classification of objects, and the PointNets perform the object localization. In summary, the Frustum PointNets is the first method for 3D object detection on unstructured point data, that includes classification and bounding boxes estimation. The evaluation in [QLW+18] provides quantitative and qualitative results. In [QLW+18], it is also demonstrated that the Frustum PointNets detector achieves comparable and partially even better performance than established object detection methods. Overall, these results confirm that PointNets are fundamentally well suited for the task of 2D and 3D object detection on lidar point clouds, but still requires the extra information from camera images.

The complete pipeline of Frustum PointNets is illustrated in Figure 2.5. This 3D object detector is structured into three main modules, the *Frustum Proposal*, the *3D Instance Segmentation* and the *Amodal 3D Box Estimation*. Firstly, a CNN predicts 2D bounding boxes for objects in an RGB image. Each image detection is used in combination with the depth image to cut out the associated region in the 3D point cloud. Therefore, the 2D region proposal of an object is transformed to a frustum and all 3D points inside this area are extracted, which makes up the search space for further processing. Besides the point cloud that contains points of a single object and its local environment, the classification information of this object is forwarded to the next module. Secondly, a segmentation PointNet determines which points in the frustum actually belong to the 3D representation of the object. A classification PointNet assigns a probability value for every lidar point to determine whether it belongs to the object of interest or not. This information is used for the masking

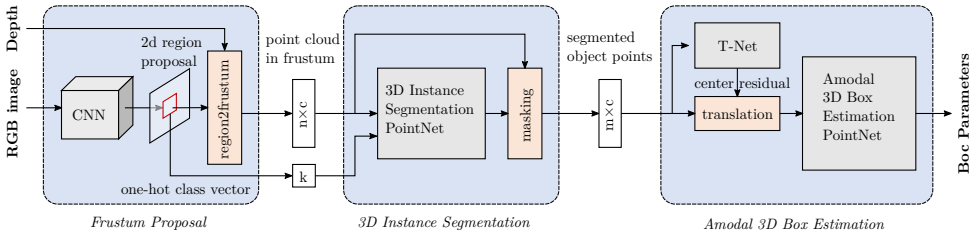


Figure 2.5: Object detection in point clouds: In the process chain of Frustum PointNets, information from a camera image and a lidar point cloud is extracted and combined to find an object detection in a specific region. The image is redrawn and inspired by [QLW+18].

process to segment all points which are actually part of the object instance. Thirdly, two different PointNets are applied to estimate a 3D bounding box modeling the object. Thus, a preceding mini-network predicts the true center of the object, which is utilized to transform the segmented point cloud into object coordinates. Then, a regression PointNet predicts the parameters of an amodal bounding box to represent the object by its position, orientation and extent. By repeating the whole procedure for all 2D object proposals in the RGB image, several classified bounding boxes are identified in the entire 3D lidar point cloud, where each corresponds to a real object.

The Frustum PointNets for object detection is realized with both, the PointNet and the PointNet++, as a base. Figure 2.6 shows the respective network architectures, where the architecture based on PointNet is referred as model *v1* and the one with PointNet++ is named model *v2*. In general, the network architectures are quite similar to those of the original PointNet and PoinNet++ models. The main difference is that a class one-hot vector is attached to the extracted feature vector, such that instance segmentation and bounding box estimation benefit from the classification information provided by the image detector. An observation that stands out is the fact that compared to the classification network in [QSMG17], the input and feature transformer networks are missing in the instance segmentation PointNet. The argumentation in [QLW+18] is that the input point cloud is already normalized by rotating it in the Frustum Proposal module. Further, in addition to the 3D position, the instance segmentation also incorporates the lidar intensity as input channel. However, the PointNet model only uses this fourth dimension for segmentation, not for box estimation. Whereas the PointNet++ model process all four channels for box estimation as well. This is not explained, but is probably, due to the fact that the bounding box parameters mainly depend on the position, where the intensity has only a small influence. But it remains questionable why this is handled differently.

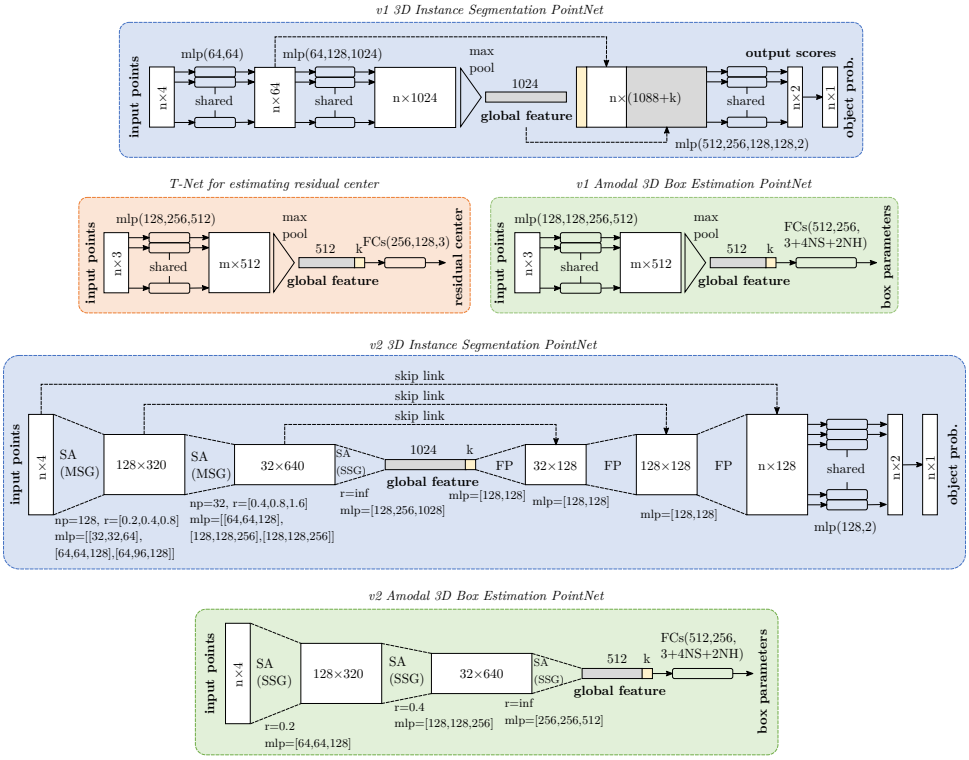


Figure 2.6: Network architecture of Frustum PointNets: Structure of the neural networks implemented in the modules using PointNet and PointNet++. The image is redrawn and inspired by [QLW+18].

The Frustum PointNets shows good results for object detection, but there are also certain limitations to this approach. The performance of the object detector depends strongly on the number of points belonging to the object. Consequently, the results are impressively accurate in a close range to the lidar sensor, when the point cloud in the frustum is dense. Especially, frustums with objects located at a large distance, result in a sparse point cloud and hence, the output of the model deteriorates. When comparing the PointNet architectures, it turns out that the overall results for instance segmentation and bounding box estimation are better when using the PointNet++ model [QLW+18]. This is expected, since PointNet++ learns more advanced features due to its hierarchical structure. One drawback of the detection system in Figure 2.5 is the dependency on the CNN object detector. When the image detector fails, there is no object proposal, and consequently, it does not exist a frustum for further 3D

object localization. Nevertheless, it is important to emphasize that the 3D Instance Segmentation and the Amodal 3D Box Estimation modules are not restricted to 2D region proposals. The last issue to be mentioned, the Frustum PointNets detector absolutely assumes that in a frustum exactly one object is located. In case of multiple instances per frustum, the segmentation of some object points is ambiguous which may have an effect on the following box estimation. Both challenges are addressed in the radar object detector proposed in this thesis. There is no dependency to a preceding CNN detector for region proposal or any data from another sensor than radar. In addition, the pipeline of this novel radar object detection system allows processing multiple instances of objects even those which are close to each other.

PointNets on Radar Data

In [SHDW18a], a neuronal network based on the PointNet++ architecture is presented to obtain semantic information for radar targets. On that account, a trained model is fed with a radar target list in order to assign a class probability to each target, whereby the network distinguishes between various categories. Every radar measurement consists of four dimensions, two spatial coordinates, the ego motion compensated radial velocity and the RCS value. Since one radar measurement cycle of a radar sensor provides relatively few reflections, the preprocessing accumulates radar data over 500ms to make the input radar target list more dense. This raises a fundamental question about how the varying radial velocity of dynamic targets with different timestamps is interpreted at the time of segmentation. It is not mentioned in [SHDW18a] how to deal with this important issue, although it is an essential aspect when processing radar data. Another option to make the input radar target list more dense is to fuse measurements from multiple radar sensor with an overlapping Field of View (FOV). While this method prevents the timestamps of the input from differing too much, it does create a dependency between these sensors. In [SLH+20], another approach in which the target list is at least partially processed without transformation is presented. [SLH+20] describes a pipeline to perform semantic segmentation on radar targets. A distinction is made between the perception of the static environment and the moving objects. For the static environment perception, the radar list is transformed into a radar grid map, which is forwarded to a CNN. For the semantic segmentation of radar targets which belong to dynamic object, a neuronal network based on PointNet++ [QYSG17] is applied.

PointNets are not only used for semantic segmentation in radar targets, but also to identify anomalies in the radar data. Hence, [GAH+21] modifies the PointNet architecture to detect noisy radar targets in the measured data. Strictly speaking, this is also a type of semantic segmentation in which the networks distinguishes between real radar targets and anomalous targets. Similarly, [CRSW21] presents an

approach that detects ghost targets in 3D radar target lists. The network used for this purpose is also based on the PointNet architecture. Summarizing, PointNets are becoming increasingly popular for different types of detection in radar target lists.

The method, which is proposed in [SFY21], performs object detection in a radar target lists using a graph based structure. Therefore, the radar targets are transformed into a graph. However, this method does not take the measured radial velocity into account when constructing the graph. The neuronal network, which is based on the Point-GNN [SR20] for lidar points, processes the radar data using the graph representation and outputs one object proposal as a bounding box for each radar target. Finally, overlapping bounding boxes are suppressed using the Non-Maximum Suppression (NMS) algorithm [RHGS17], which results in exactly a single bounding box for each object. Section 3.1.2 explains the NMS technique in more detail, since it is also a part of the radar object detection system that is proposed in this thesis.

An overview of techniques for object detection in radar target lists generated by automotive radar sensors is given in [SKA+21]. This thesis proposes a radar object detection system which directly processes a radar target list in a deep neuronal network using multiple PointNets without any transformation of the input data. In contrast to [SHDW18a], the radar data is neither accumulated from different measurement cycles, nor from multiple sensors, but rather, the object detection is performed on a sparse radar target list. Further, the radar object detector in this thesis does not use information from another sensor, as it is done in [QLW+18]. In summary, the radar object detection system proposed in Section 3.1 successfully operates on a sparse radar target list generated by a single automotive radar sensor.

2.3 Multi-Object Tracking

In many perception applications, tracking of objects is a crucial prerequisite for modeling objects in an arbitrary environment. The goal of object tracking algorithms is to estimate the state of dynamic objects over time, where the state comprises for example position, orientation and velocity. In this context, single object tracking methods focus on one specific object, while multi-object tracking system are able to track several objects in parallel. Thus, multi-object tracking is the task of estimating both, the number of dynamic objects and the state of each object simultaneously using a noisy sequence of uncertain measurements. In literature, three approaches for multi-object tracking are established, namely, the JPDA [BF88], the Multiple Hypotheses Tracking (MHT) [Rei79] and the multi-object Bayes filter [Mah07]. The last one is based on the Random Finite Set (RFS) framework [Mah07] which is useful to model multiple objects at the same time in a tracking application. This section

only introduces the essentials for object tracking, which are important for this thesis.

In multi-object tracking, the current dynamics of a single object are mathematically represented by a stochastic state vector $\underline{x}$. The state may comprise values for position, orientation, velocity, yaw rate, acceleration or even more. As an example, the vector

$$\underline{x}^{\mathrm{T}} = [x, y, \theta, v, \omega], \tag{2.2}$$

describes the state including spatial position (x, y), orientation θ, velocity v and yaw rate ω of an object. A tracking filter has to estimate the object's state $\underline{x}_k$ at time step k using the noisy measurement $\underline{z}_k$ of the same time. However, the measurement vector does not necessarily have to contain all the same state components as the state vector. In order to continue the example, if the current measurement vector is

$$\underline{z}^{\mathrm{T}} = [x, y, \theta], \tag{2.3}$$

which is measured by an arbitrary sensor, then it only contains values for the x and y position as well as the orientation θ. Nonetheless, the application of motion models during the tracking process still allows to estimate all the values of a state vector.

In general, a tracking algorithm performs two steps, a prediction step and an update step. During prediction the estimated state $\hat{\underline{x}}_k$ of each tracked object is predicted into the time step $k+1$ by applying a motion model. Next, the algorithm incorporates each obtained measurement $\underline{z}_{k+1}$ during update to correct the predicted state $\hat{\underline{x}}_{k+1|k}$ of each object. After the prediction and update step, each object is described by the estimated state $\hat{\underline{x}}_{k+1}$ which represents the dynamic state at time $k+1$ taking into account all available measurements. The basic method for single object-tracking to perform the prediction and update step, is the Kalman filter [Kal60] for linear systems, and the Extended Kalman Filter (EKF) [BF88; BP99; Jaz70; Sor85; Uhl92] and the Unscented Kalman Filter (UKF) [BWT11; JU04; JUD00; JUD95] for nonlinear systems. In multi-object tracking algorithms, these steps are much more complex, but even there, a Kalman filter, a EKF or a UKF are the underlying basis.

LMB Filter

This thesis implements the RFS tracking approach by applying an existing and approved realization of the Labeled Multi-Bernoulli (LMB) filter [Reu14; RVVD14]. The idea of the RFS filtering is to model the multi-object state as an RFS. In order to be able to track the label of an object in addition to its state and the number of objects, [VV13] introduces the labeled RFS. Further, the δ-Generalized Labeled Multi-Bernoulli (δ-GLMB) filter [Vo08; VV13] is the first analytic implementation of the multi-object Bayes filter based on labeled RFSs. Since the computational complexity

of a δ-GLMB filter exponentially increases with the number of objects, this filter is not suitable real-time critical system. However, the LMB filter [Reu14; RVVD14] is a time-efficient approximation of the δ-GLMB filter to achieve real-time capability. Although the approximation results in a loss of information, an LMB filter shows almost the same performance as a δ-GLMB filter in most typical tracking scenarios.

The theory and mathematical description of the LMB filter is explained in [Reu14; RVVD14]. This includes the derivation of the LMB RFS, as well as the filter equations including the prediction and update step. This thesis implements a Gaussian Mixture (GM) LMB filter for tracking objects in radar data. The GM LMB filter is fully described and illustrated in [Reu14], including the specification of all the equations.

The standard LMB filter expects one measurement for each track. However, since a high-resolution radar sensor generates several targets per object, this assumption is not fulfilled. In [BRG$^+$16], an approach to incorporate multiples radar targets per object using an LMB filter is introduced. This approach generates multiple object hypotheses by partitioning individual measurements into various groups, where every partition represents a different assignment of measurements to objects. Hence, this automatically results in several hypotheses for one object, meaning that this technique implicitly realizes a kind of multiple hypothesis tracking. With the intention of incorporating individual radar reflections into such an LMB filter, [SD19] presents a radar measurement model designed for such a purpose. This technique allows to determine a predictive density in a radar target list based on a particular vehicle state. More details about this method is given in [Sch19]. In general, this is a convenient method to process radar target lists in tracking algorithms, but the radar model proposed in [SD19] is limited to vehicles. This thesis uses the standard LMB filter for multi-object tracking in radar data. The radar object detection system proposed in Section 3.1 outputs exactly one object detection for each object. Thus, all assumptions for the LMB filter are fulfilled. Section 3.3 describes a suitable technique to successfully integrate the recognized radar object detections into an LMB filter.

Chapter 3

Object Detection in Radar Data

The detection of objects based on sensor data is an important task to capture the perception of the environment around a vehicle. Advanced Driver Assistance Systems (ADAS) are already able to recognize objects in front of a vehicle with the help of data from a radio detection and ranging (radar) sensor. In applications, for example, adaptive cruise control system or distance warning, it only matters whether there is an object in front of the vehicle. Such kind of tasks can be accomplished with relatively simple algorithms, since it is sufficient to check if an object in front of the vehicle generates radar reflections or not. However, the demand for autonomous driving requires more information in object detection, such as the class of an object or its specific position and heading as well as its extent. That requires high-resolution radar data, but also much more sophisticated algorithms to extract all the information from the radar target list. This thesis presents a novel radar object detection system based on a deep learning technique using high-resolution radar data. The proposed object detector allows to process pure radar data in its natural target format without any transformation. Further, the object detection is based solely on radar reflections from one measurement cycle of a single sensor. That implies that there is neither an accumulation of multiple measurement cycles or various sensors, nor a fusion of radar data with measurements from other sensors like light detection and ranging (lidar) or camera. The application of deep learning techniques requires solving several challenges. Firstly, often the input data has to be transformed into a suitable format for the neuronal network. The radar sensor employed in this thesis generates reflections as targets with multiple dimensions. Since the presented radar object detection system is able to directly process point sets by using PointNets as a basis, this issue is obsolete. Secondly, the complete system needs to be assembled with appropriate components and the architecture of a neuronal network must be defined. The proposed radar object detector consists of three main components which themselves contain neuronal networks with different architectures. Each network architecture comprises various layers which have to be chosen in such a way that the neuronal network is enabled to handle the input data

and achieves the expected result. Thirdly, a training strategy is necessary to find an optimal solution. And last but not least, the data must be prepared for training purposes which includes collecting, labeling as well as preprocessing the radar data.

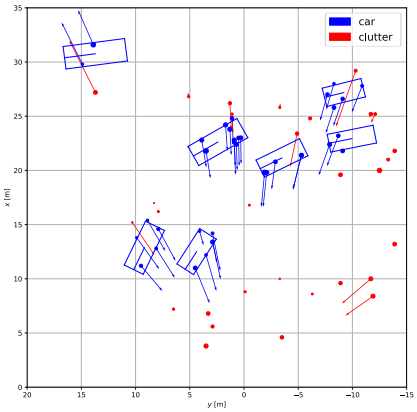
This chapter proposes suitable solutions for all mentioned issues. Section 3.1 deals with the entire radar object detection system. This comprises the problem formulation which mathematically defines input data and the object detection result as desired output. Moreover, the complete pipeline of the radar object detector is introduced and each individual component is explained in detail. Subsequently, the network architecture, as well as the training process are given. Section 3.2 describes the implementation details for the realization required to employ the application in a real system. Finally, Section 3.3 presents an appropriate measurement model for the obtained radar object detections to process them in a multi-object tracking system.

The basic approach of the radar object detector proposed in this chapter, is already pre-published in [DGBD19]. However, the method presented in [DGBD19] has been significantly further refined in this thesis. Hence, the thesis proposes a complete radar object detection system to recognize traffic participants of different classes. Furthermore, Section 5.1 presents a greatly improved and more detailed evaluation.

3.1 Radar PointNets

This thesis proposes a sophisticated radar object detection system to solve the object detection task in real high-resolution radar data. The proposed realization relies on a deep learning technique based on PointNets to process radar reflections in their natural form, namely, as radar targets. Section 3.1.1 mathematically defines the problem of object detection in radar data. Then, a solution is designed which solves the subproblems of the detection task step by step. Section 3.1.2 describes the entire radar object detection system from raw radar targets as input to object hypotheses in form of classified 2D bounding boxes as output. In the application of deep learning, neural network design is an important part. Section 3.1.3 presents the network architectures of the applied models based on PointNet and PointNet++. After designing the neuronal networks, these have to be trained, that means to identify the optimal model parameters based on labeled training data. Section 3.1.4 explains the training process by introducing a loss function as well as a technique to train the complete system. Finally, Section 3.1.5 gives a final conclusion about the proposed method. In summary, this section presents the entire radar object detection system from concept over design to realization through deep learning techniques.

(a) Camera view of the traffic scene.



(b) Reflections of the partial radar view.

Figure 3.1: 2D object detection in radar data: Exemplary radar target list with reflections belonging to cars (blue) or clutter (red), where the length of arrows visualizes the radial velocity and the size of targets displays the RCS value. The blue amodal 2D bounding boxes represents position, heading and size of objects, here cars.

3.1.1 Problem Formulation

The aim of the radar object detection system is to find all objects, which are located in the Field of View (FOV) of the radar sensor. The object detection task includes classification and localization of various objects in a 2D space based on a radar target list. This means each object is modeled as a classified and orientated 2D bounding box. Figure 3.1 visualizes a part of the radar measurements and the corresponding camera image for an exemplary real-world scenario, where multiple vehicles are located at a two-lane intersection. The employed radar sensor is mounted frontal on the ego vehicle and is aimed directly at the intersection. In this example, all present objects are represented by vehicles, but the radar object detector proposed in this thesis is able to detect multiple classes, namely, vehicles, trucks, bikes and even pedestrians.

In order to define the problem mathematically, a set of targets $\mathcal{P} = \{p_i | i = 1, \ldots, n\}$ represents the input radar target list, where $n \in \mathbb{N}$ denotes the number of radar targets. Furthermore, each target $p_i = (x, y, \tilde{v}_r, \sigma)$ has four dimensions and contains the (x, y)-coordinates, the ego motion compensated radial velocity $\tilde{v}_r$, and the Radar Cross Section (RCS) value σ. The radar data is generated by multiple high-resolution automotive radar sensors, which perform a proper data preprocessing to provide data in the target list format. The object detection system provides an output for

each measurement cycle of a single radar sensor. This means that neither radar data is accumulated over time nor the measurements from multiple sensors are combined.

As already mentioned, a subtask of object detection is the classification task. For this purpose, the radar object detector has to distinguish between multiple classes, which are defined as *car*, *truck*, *bike*, *pedestrian* and *clutter*. Mathematically the set

$$\mathcal{C}_{cls} = \{(\text{clutter}, 0), (\text{car}, 1), (\text{truck}, 2), (\text{bike}, 3), (\text{pedestrian}, 4)\}, \tag{3.1}$$

describes each class c as a pair of a specific class name and a number. Consequently, this is a multi-class problem with five different categories, where the set $\mathcal{C}_{cls}$ comprises all of them. The special class *clutter* describes the case that the considered radar targets do not belong to any real object, or expressed in other words, it represents the *non-object* case. Otherwise, all occurring objects are assigned to one of the other classes. If a real object cannot be mapped exactly to any of the given categories, the class with the next best match is selected. For example, there is no separate category for a bus, but as buses and trucks look similar in radar data, all buses are assigned to the class *truck*. The same reasoning applies to motorcycles and bicycles, both are mapped to the more general *bike* category. Naturally, it would be reasonable to keep these more finely structured classes, since certainly each of them show noticeable differences in the radar measurement. In the case of bicycles, for example, it is possible to recognize the cyclic movements of the pedals in the radar measurements. In contrast, this effect does not occur with a motorcycle because it is driven by an engine. However, in order to recognize such fine details, the radar target lists representing the environment have to be dense. But as this is not guaranteed at all times when using an automotive radar sensor, this thesis focus on the approach of combining as many classes as possible for similar appearing radar measurements.

In the case that radar targets are classified as object, the localization of these objects is the other part of the object detection system. Hence, the radar object detector performs the localization task by determining an amodal 2D bounding box. In real-world scenarios, situations may occur, where only parts of the real object are visible to the radar sensor, for example due to shadowing, obscuring or the object being only partially located in the sensor's FOV. The prediction of an amodal bounding box means that even in such situations, always a box that surrounds the entire object is estimated. The final 2D bounding box is described by several box parameters, which are the center position (x_c, y_c) and heading angle θ in the xy-plane as well as the size containing the length l and width w, which describes the extent of the object.

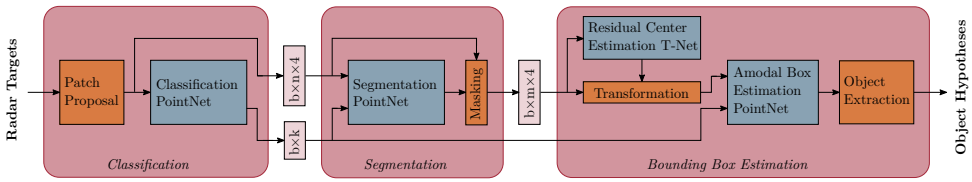


Figure 3.2: Overview of the 2D object detection system in radar data using PointNets: Firstly, the classification module divides the radar target list into patches and classifies them to find regions containing objects. Secondly, the segmentation module performs an instance segmentation by classifying each of the n radar targets per patch. Thirdly, the bounding box estimation regresses a 2D box for each object based on all the m segmented radar targets.

3.1.2 Radar Object Detection

The goal of the presented method is to perform object detection using solely radar data which is provided by a high-resolution automotive radar sensor. A basic version of this approach is already presented in [DGBD19], but the thesis describes the complete system with a number of adjustments resulting in improvements. Figure 3.2 shows an overview of the proposed 2D radar object detection system, which consist of three principal parts: the *classification* module, the *segmentation* unit and the *bounding box estimation* module. In order to generate object hypotheses from the radar target list as input, certain components are interconnected to forward extracted information to the next module. This process finally results in box parameters to model each object as a separate 2D bounding box. The following sections provide detailed information about the structure and functionality of the respective modules. For this purpose, the entire processing chain is explained in depth and is illustrated with a real-word example. Figure 3.1 visualizes the radar data for a traffic scenario. In order to describe the radar object detection system in detail, the target list is processed step by step, and all intermediate results are depicted as well as explained.

Classification

The first module of the radar object detection system is the *classification* unit. This module expects the radar target list provided by the radar sensor as input data, where each radar measurements comprises the position as (x, y) coordinates, the

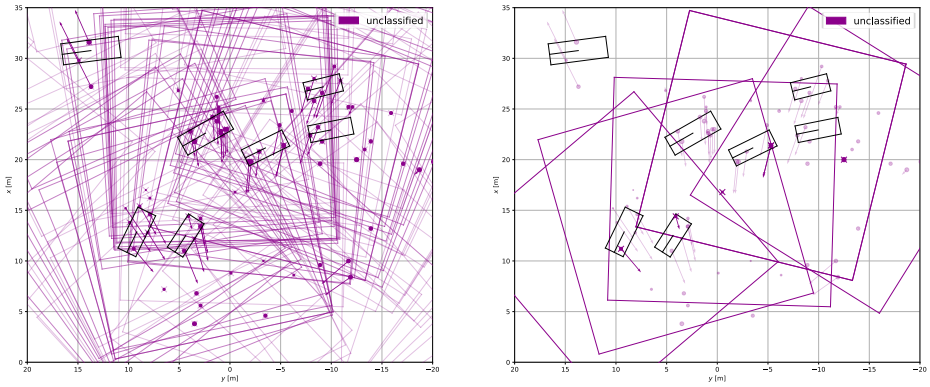
ego motion compensated radial velocity $\tilde{v}_r$ and the RCS value σ. The radar target list is directly forwarded to the *patch proposal* component. This module divides the entire radar target list into many regions of interest. Objects that are located in these areas and are to be detected are referred to below as objects of interest. It is important to clarify why the patch proposal concept is reasonable or even essential. Another possibility would be to pass the task of generating regions of interests to the neural network as well. A method of how this can be achieved in 3D point clouds is suggested in [QLHG19]. But the patch proposal technique has some convincing advantages. The fundamental motivation of this approach is that after the normalization of patches, objects that are located at different positions result in similar patches. Hence, after generating and normalizing the regions of interest, it does not matter where objects are actually positioned in the radar target list with respect to the angle. Consequently, the neuronal network does not have to learn all angular regions in the sensor's FOV, and, as a result, it requires far fewer learning examples for the training. The normalization method is explained in detail below. Another advantage is that the patch proposal approach reduces the complexity of the overall detection problem. When considering the entire radar target list, the challenge is to detect an undefined number of objects which may be located anywhere within the target list. However, the patch proposal technique reduces the task to find exactly one object per patch which is always located in the region's center or rather approximately in the center of the extracted radar target list. An evaluation concerning the complexity of computation is hard to give. In principle, solving several small tasks which are less complicated, is often faster than finding the solution of one complex problem. Since this investigation is not part of the thesis, no statement concerning this issue is made. It can be assumed that a parallel execution of the radar object detection system on a Graphics Processing Unit (GPU) accelerates the calculation significantly, because this allows a processing of all generated patches simultaneously. Section 3.2 gives more details on this topic and describes how parallelization is implemented in an application for real operation.

The patch proposal creates a rectangle with specific length and width, called patch, around every radar target, where the target under consideration specifies the center point of the region. The choice of a patch's length and width is essential for further processing steps, since a patch should enclose the full potential object. This creates a dilemma, because if the patch is too small, large objects like trucks will not fit within it. But, if the region of interest is too large, small objects such as pedestrians might get lost, because there might be other objects inside the patch that are attracting the focus. In order to comprehend this issue, it is important to understand what consequences the application of the patch proposal component has for the whole object detection processing chain. In fact, this method is designed to detect exactly one object per patch. Since each radar target defines a custom patch, there are several patches where the same object of interest is located in the patch center. Thus, the patch proposal produces so many regions of interest, that each real object

is guaranteed to be repeatedly in the center of various patches. Nevertheless, it is possible that various objects are in the same patch at the same time following the rule, the larger a region, the more objects can be located inside it. In the worst case, a few objects are very close to each other at the center of a region. Even though, the radar detection system is trained that the target object is located in the very patch center, it may happen that another object close to the actual one is detected. Thus, the wrong object attracts the focus. Due to the fact that an object usually generates more than one radar target, and consequently multiple patches, this behavior does not necessarily imply that the object is not detected at all. In theory, such a behavior is not really desired when examining the detection pipeline at a patch level, because this type of prediction is strictly speaking a false detection. However, since the radar object detections of each patch are merged into one final output, such false predictions usually have no effect on the object detection result. Another aspect to consider when choosing the patch size is, the region under examination should be large enough to ensure that the radar targets still represent the typical measurement behavior of a radar sensor. As already mentioned before, the measured radar reflections do not only originate from dynamic objects, but also from the static environment, which are referred to as *clutter* measurements. With the goal to model an object as a bounding box, it is essential to extract only the radar targets belonging to the object of interest. This may be performed manually by an algorithm specially designed for that purpose. Alternatively, a neuronal network can learn this extraction and integrate it into the pipeline. The object detection system presented in the thesis makes use of this learning approach and solves the challenge in the segmentation component. In order to enable the network learning the occurrence of real radar reflections and clutter measurements, the region under examination must realistically reflect this characteristic behavior. For that reason, the patch needs to be large enough so that sufficient radar targets are placed in it. Section 5.1.3 presents the results of determining a reasonable patch size in this thesis.

The patch proposal extracts a region of interest for each radar target. Figure 3.3 visualizes the results for all reflections on the left side. Since this illustration is confusing, the right side of the figure exemplary visualizes some patches for individual radar measurements. In the end, the patch proposal generates various regions with slightly different positions and orientations for an object comprising multiple radar reflections. Since all patches are processed in the pipeline, the detector ultimately provides multiple hypotheses for the same object. In order to obtain a single detection, the detection result requires further processing. In the subsequent description of the corresponding module, a convenient method of achieving this objective is illustrated.

Subsequently all regions of interest are normalized by rotating them into the center view. Therefore, the radar target list inside the patch is rotated by the azimuth angle of the central radar target which defines the patch. In Figure 3.4 the normalization step is visualized. This method ensures rotation invariance of the algorithm. It



(a) Complete output of the patch proposal for the radar target list in the scene.

(b) An exemplary selection of some generated patches for a better visualization.

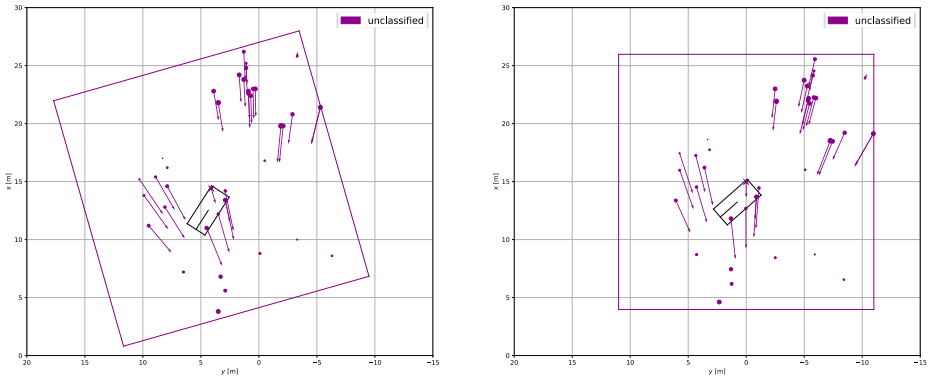
Figure 3.3: Patch proposal for a radar target list: Each radar target defines a specific patch which is visualized as colored square. The radar targets defining a patch are highlighted and marked with a cross. The ground truth is colored in black that to allow distinguishing which reflections, or rather which patches, belong to the object.

guarantees that regardless of the measured angle of the object's radar targets towards the sensor, after normalization all radar targets generated by the object of interest have an azimuth angle close to zero. This process only normalizes the radar target list of a patch regarding the aspect angle, but not with respect to the range. That is deliberately realized in such a manner, because the range information is helpful in the classification process. Due to the angular resolution of a radar sensor, objects near the sensor produce many more reflections than at a larger distance. Accordingly, this valuable information should not be neglected in the following classification.

Afterwards, the radar target list of each patch is passed to the actual classification process to determine whether the respective targets contain an object or not. The classification component is composed of a PointNet to classify the patch under consideration of all radar targets. The classification network distinguishes between the defined classes *car*, *truck*, *bike*, *pedestrian* and *clutter*. If the patch is classified as *clutter* it implies that the examined radar target list does not contain any objects of interest. The classification PointNet estimates a probability vector for each patch with values $p_{cls,c} \in (0, 1)$ per class $c \in \mathcal{C}_{cls}$. Finally, the predicted class is given by

$$\hat{c} = \underset{c \in \mathcal{C}_{cls}}{\arg\max}(p_{cls,c}), \tag{3.2}$$

where, the *argmax* function extracts the class index c associated to the maximum



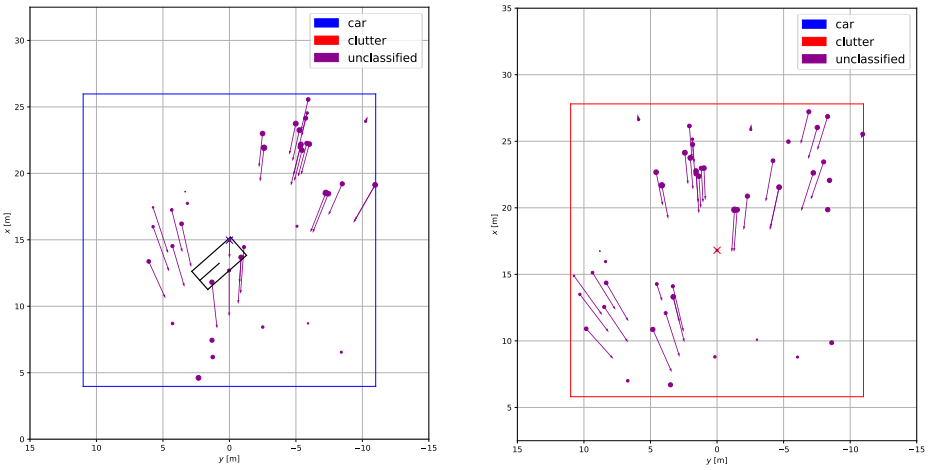
(a) Extracted patch with a car in the center. (b) Patch after rotation to the center view.

Figure 3.4: Patch normalization: Each patch, respectively the radar targets inside, is rotated to the center view for normalization purpose. As an aid the object of interest is outlined as a black bounding box.

probability $p_{cls,c}$ that represents the most likely class of the prediction. Finally, the relation of Equation (3.1) maps the integer value $\hat{c}$ to the corresponding class label.

The neuronal network is trained that the radar target which defines the region of interest, that is the radar target in the patch center, actually specifies the class of the whole patch. This implies that if the central radar target belongs to an object class, the patch should be recognized with the according category. Otherwise, the region of interest should be classified as *clutter*. As a consequence, it is possible that one or more objects are located inside the radar target list of the patch, but the classifier assigns it to the non-object class. Since each target is the center of a patch once, all objects can still be recognized. Furthermore, there are several possibilities to detect an object, because one object usually generates multiple radar reflections. Accordingly, various patches belong to the same object of interest. This procedure of patch classification is essential to ensure a unique identification in multi-object scenarios. If more than one object of different category is present in the radar target list, the assignment of a class to the patch is ambiguous. In order to avoid this aspect, only the radar target in a patch center is of importance, or more precisely, the object to which this central target belongs to is actually the object of interest.

The classification outcome is assigned to the entire patch. Figure 3.5 shows the result of the classification component exemplary for two patches extracted from the radar measurement cycle. In both regions the radar target list contains multiple objects. In the example on the left side, the central radar target is a reflection of a vehicle, consequently the radar target list in the patch is classified as *car*. The vehicle



(a) Radar target list classified as *car*.

(b) Radar target list classified as *clutter*.

Figure 3.5: Patch classification: All patches are individually classified to determine if the radar targets belong to an object or not. Two examples are given where both patches contain multiple objects, but only in the left one, the central radar target indeed belongs to the object of interest (black box) and is classified accordingly.

produces five more radar reflections in this scene. The radar sensor measures two targets on the rear left wheelhouse located at the identical position but with different radial velocities. In the optimum case, the classification assigns all six patches to the *car* category. Hence, the object detector creates exactly six hypotheses for the same vehicle. Although, the patch on the right side comprise almost the same radar target list, the patch is correctly classified as *clutter*, because the radar target in the center does not belong to any object. This example visualizes the classification principle for objects applying the proposed patch proposal technique. Particularly, it clarifies how the detection system ensures an unambiguous classification result for the case when several objects are present in the radar target list of a single patch.

Segmentation

After the classification process, the radar target list of each patch as well as the predicted class is forwarded to the segmentation module to determine the radar targets which belong to the current object of interest. This segmentation procedure is applied for all patches regardless of the results previously obtained, also for the target lists that are classified as *clutter*, which is the *non-object* class. Nevertheless, the

segmentation results are only relevant for real objects, because later, the predictions for *clutter* patches are ignored. Another classification PointNet performs the radar target segmentation. This PointNet predicts a probability score $p_{seg,o} \in (0, 1)$ for each radar target in the patch. This value indicates the probability that a radar reflection belongs to the object. Since the segmentation network already has the class information $c \in \mathcal{C}_{cls}$ of the previous module at its disposal, that the PointNet only distinguishes between the classes *object* or *non-object*. Thus, class o is an element of

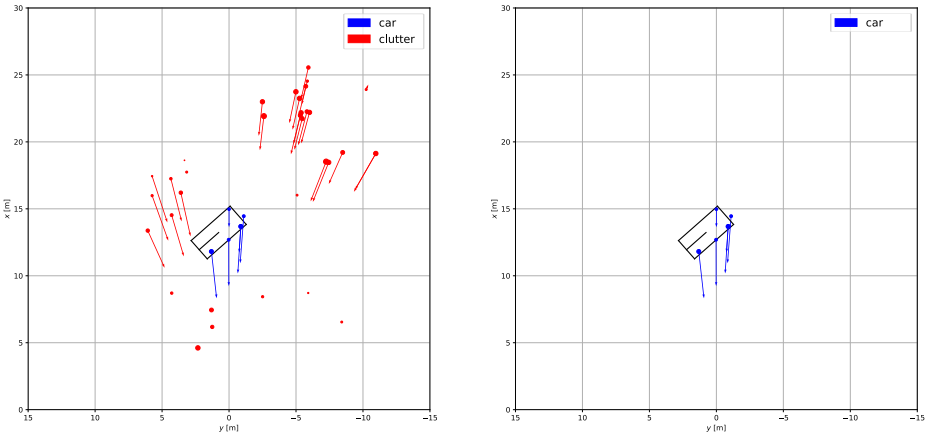
$$\mathcal{C}_{seg} = \{(\text{non-object}, 0), (\text{object}, 1)\}, \tag{3.3}$$

hence, this is a logistic regression problem. The actual decision whether a radar reflection belongs to the object of interest or not, is obtained by maximizing the probability vector from the segmentation PointNet. An *argmax* function calculates

$$\hat{o} = \arg\max_{o \in \mathcal{C}_{seg}} (p_{seg,o}), \tag{3.4}$$

for each radar target, which results in the final segmentation information. Although, the network estimates another class o for all radar targets in a patch, this prediction depends on the class information c. Therefore, the feature vector of the segmentation network is expanded by the predetermined class information $\hat{c}$, which is encoded in a k dimensional one-hot class vector. A one-hot vector is a vector that contains the value zero at all components except at the component which represents the class under consideration, where the value is one. Section 3.1.3 addresses this issue, when describing the network architecture of the radar object detector as well as the feature vectors for the respective main modules. Since the segmentation procedure of radar targets considers the individual class information c, the segmentation result of an identical targets list may vary with regard to the previously determined category. Consequently, the neuronal network learns a class dependent segmentation by utilizing a prior class information to determine the optimal segmentation result.

The masking step extracts all radar targets that are classified as *object* by the segmentation network. In other words, this procedure removes all radar reflections from the input target list of a patch that do not belong to the object of interest. In order to make sure that the algorithm is translation invariant, the masking process additionally normalizes the coordinates of masked radar targets as presented in [QLW+18]. Therefore, it transforms the target list into a local coordinate system with the centroid of the segmented radar targets as origin. Hence, for further processing, rotation and translation invariance are guaranteed. This implies that distance information of radar targets affects the patch classification and target list segmentation, but not the following bounding box estimation. However, rotation of objects respectively the azimuth information of their associated radar targets has almost no influence on any of the modules due to the previous patch normalization.



(a) Radar target list after segmentation.

(b) Radar target list after masking.

Figure 3.6: Target list segmentation and masking: Primarily, all radar measurements that belong to the object of interest, depicted as a black bounding box, are marked. Subsequently, only radar targets that are classified as reflections of that object are kept for next processing, and all others are removed from the target list.

At the end, the segmentation provides a list with all radar targets belonging to the object. Figure 3.6 visualizes the segmentation result for a vehicle. The left side shows the outcome of the PointNet that classifies all radar targets of the object. Thus, this network performs an instance segmentation for the object of interest. The right side illustrates the resulting masking process that extracts all radar targets classified as objects. For further processing only these radar targets are relevant, since the bounding box solely depends on measurements reflected from that object.

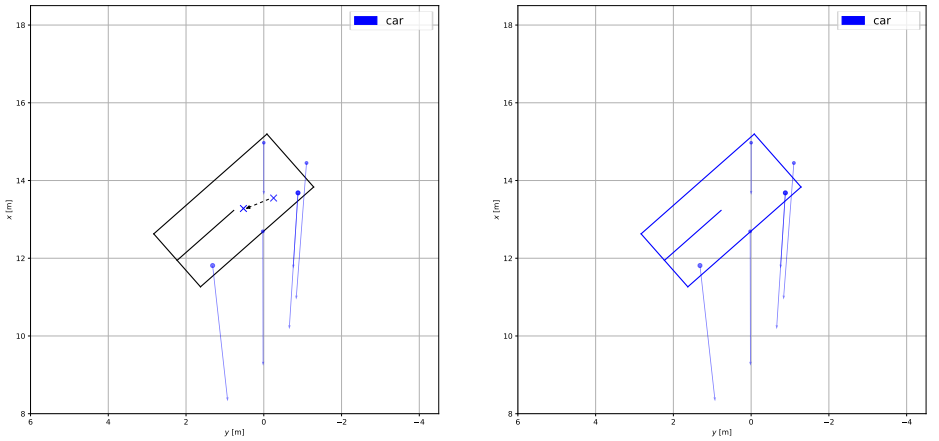
Bounding Box Estimation

The goal of the bounding box estimation is to localize all objects in the radar target list. There are a total of two tasks for the module. Firstly, for each patch, it predicts bounding box parameters based on the classification information and the results of the instance segmentation. Secondly, it chooses one of the various detection hypotheses per object, because usually several patches comprising the identical object. This means that if an object appears in multiple patches, several independent hypotheses are generated. Subsequently, a separate module selects the most probable one for this object. In order to estimate a complete 2D bounding box, three parameters are necessary, namely, the center coordinates, the heading and the

extent. For this purpose, two neuronal networks are applied, one for estimating the real center of the object, and one for predicting its orientation as well as its extent.

After the masking process, the spatial coordinates are present in relation to the centroid of extracted radar targets. As [QLW+18] states, the real object's center is often different from that of the amodal bounding box. This occurs especially, when a lidar sensor is applied, because it only receives measurements targets from the sides that are directed towards the sensor. Since often only parts of the object are visible for the lidar sensor, the centroid is quite far from the real object center. For this reason, [QLW+18] proposes a lightweight regression PointNet called Transformer PointNet (T-Net) to estimate the center coordinates with respect to the entire object. This T-Net only cares about transforming the centroid to the actual center of the object, which makes it extremely fast to process. The advantage of a radar sensor is that it allows to obtain reflections from the non-visible side, for example the opposite wheel housing of a vehicle. Since the radar target list is sparse and rarely all prominent spots of the object are measured, the centroid of masked radar targets does not match the true object center. That is why the bounding box estimation unit includes a T-Net to predict the residual center. Finally, the transformation component translates the segmented radar target list into a local object coordinate system with the predicted center as origin using the output produced by the T-Net.

The next stage is the estimation of the amodal 2D bounding box based on the transformed radar targets and the classification information of every patch. For this purpose, another regression PointNet predicts the box parameters by utilizing the residual approach described in [QLW+18]. The bounding box estimation depends on the previously determined classification result. In order to integrate the classification results, features generated in the bounding box network are extended using the predicted class. Since PointNet provides residual values for the center coordinates regarding the previously estimated object center by the T-Net, the absolute bounding box center combines the center residuals from the T-Net as well as those from bounding box network. The size and heading estimation applies a hybrid formulation that unites the typical classification and regression techniques. Therefore, predefined size templates and angle bins are deployed. This method is similar to anchor boxes that are introduced in [RHGS17]. An established option is to specify a box with average length and width values for each class. The average sizes per category is derivable from the employed radar dataset. The division of angle bins is uniform, which means for example, a graduation in 30° intervals requires twelve bins. That is exactly the way this thesis proceeds for the choice of templates. The PointNet classifies the size and angle template based on the input target list. This technique is already applied in [QLW+18] and is explained in detail there. Summarizing, it predicts a score for all size and angle categories. Here, the expanded class information has a particular impact on the classification result of the size template, but this does not mean that automatically the one defined for the corresponding class is always



(a) Predicted center of the object. (b) Predicted bounding box of the object.

Figure 3.7: Bounding box estimation: A mini-network predicts the center coordinates of the object (black) that is illustrated by shifting the blue cross. After transforming all radar targets into the local object coordinate system, another neuronal network estimates the parameters to describe the object as a bounding box, that is the adjusted center position, as well as its heading and its size.

selected. Further, PointNet estimates a residual for every category to obtain the precise values for size and heading of the bounding box. Thus, the absolute size and heading results from the most probable predefined template and the corresponding predicted residual values. This is a common approach in object detection, more information about it when using image data is given in [MAFK17; RHGS17]. As in the segmentation module, the bounding box estimation unit determines box parameters for all patches, including those classified as *clutter*. Since there is no object in these regions, estimating a bounding box actually does not make any sense. But in the following processing stage, all patches which not containing an object are discarded and only the prediction results for classified objects are processed further.

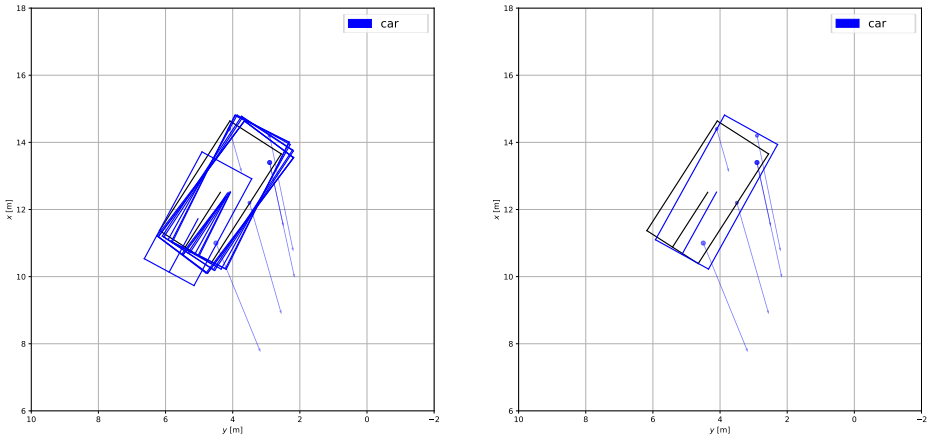
The bounding box estimation is performed for each generated patch. Figure 3.7 shows the results for a vehicle that generates six radar reflections. The left side visualizes the centroid of the target list and the predicted true center coordinates. In this example, the difference between centroid and center coordinates is considerable. The right side shows the estimated bounding box for the object of interest. The center coordinates of the bounding box are slightly adjusted compared to the previously estimated value. The heading of the object corresponds to the driving direction which is expected according to the measured radial velocity measurements. Further, the

size estimation is reasonable, as the reflections of wheel houses are quite recognizable. Since a radar sensor typically produces noisy measurements, not all radar targets are located within the bounding box, although these belong to the corresponding object.

Once the radar target list is divided into patches, the radar object detector performs the explained actions visualized in Figures 3.4 - 3.7 for all patches. In summary, the detection system predicts a classified bounding box based on the radar target list in a patch. As already mentioned, this results in several hypotheses per object since one object typically causes multiple radar reflections. However, the radar object detection system ultimately outputs only one hypothesis per object, specifically the one which matches the real object best. Hence, a last component in the detection pipeline is responsible for the matching, which is called the object extraction. Beside the prediction results, all neuronal networks, that are PointNets for classification, segmentation and bounding box estimation, implicitly supply an individual score $\mathcal{S}_i \in [0, 1]$ for each of the predictions. This value indicates how confident the neuronal network is with its prediction result and can be interpreted as probability. For every region, the bounding box estimation module combines all these scores into one comprehensive score. Hence, for one object detection the confidence score is given by

$$\mathcal{S} = \mathcal{S}_{cls} \times \mathcal{S}_{seg} \times \mathcal{S}_{bbox} \in [0, 1], \tag{3.5}$$

where $\mathcal{S}_{cls}$ is the classification score, $\mathcal{S}_{seg}$ the segmentation score and $\mathcal{S}_{bbox}$ the score of the 2D bounding box. The multiplication of all scores ensures $\mathcal{S} \in [0, 1]$, thus, the resulting score may be interpreted as the detection probability of an object. As a consequence, the confidence score may be used for filtering object hypotheses by the object extraction module. There are two selection issues in this detection system, one is optional for filtering out unlikely hypotheses, the other is mandatory for satisfying the requirement of generating exactly one hypothesis per object. Since the confidence score represents a probability, that value may be used for filtering purposes of improbable predictions. The extraction module removes all object hypotheses with a confidence score lower than a specific threshold γ_{obj}, which has to be chosen depending on the field of application. This operation is optional and recommended primarily for practical applications. However, what cannot be avoided is the filtering of multiple hypotheses belonging to the same real object. For the purpose of finding the best matching hypothesis per object, the confidence score is applied again. In this calculation, the optimum is specified by the self-assessed confidence of the different neuronal networks. In order to realize this task, the object extraction unit performs a Non-Maximum Suppression (NMS) [RHGS17]. The NMS filters all predicted radar object hypotheses regarding the score and overlap of bounding boxes. Therefore, the algorithm obtains a list including all proposals, which comprises the bounding boxes together with the corresponding confidence score for each patch. As a result the NMS returns a filtered list containing the most probable non-overlapping bounding boxes, where the likelihood of a bounding box is represented by its individual score.

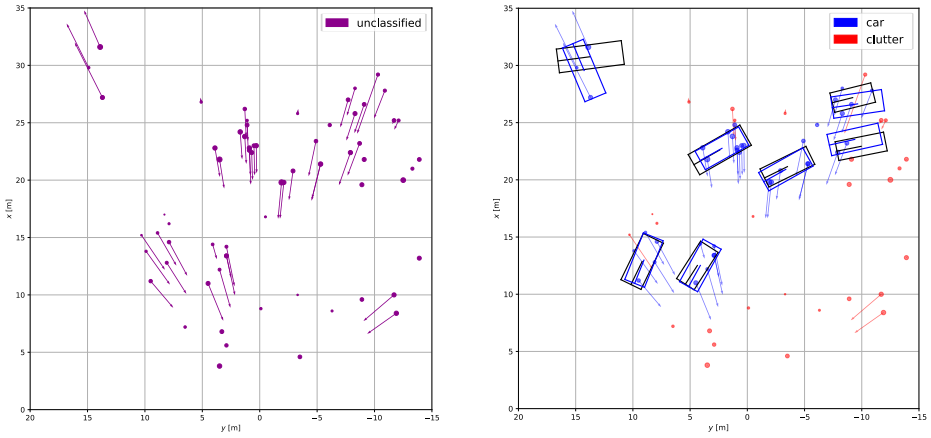


(a) Various predicted radar object hypotheses from different patches for one object.

(b) The most likely radar object detection for the object after the NMS method.

Figure 3.8: Non-Maximum Suppression: The left side illustrates all predictions for the same object of interest. Since the object is present in multiple patches, this leads to various overlapping bounding boxes. The right side depicts the selected radar object hypothesis after the NMS, that is the predicted bounding box with highest detection score. The ground truth object is drawn as black box.

The NMS is a technique that was first applied in computer vision, for example in the Faster R-CNN [RHGS17] object detection system. This postprocessing algorithm is designed for the purpose to merge all proposals of bounding boxes which belong to an identical object. The filtering of proposals depends on several factors, the object extraction in this thesis utilizes an NMS based on two criteria, the overlap of estimated bounding boxes and the predicted confidence score. In the following the NMS algorithm and its individual steps are explained by processing the preceding outputs of the radar object detector. In general, the pipeline of the radar object detection system generates various classified bounding box proposals for every object. The amount of object hypotheses depends on the number of radar targets reflected from the real object. Thus, the more radar measurements belong to that object, the more patches are generated for it. And if eventually the predictions are correct, the radar detector generates a corresponding number of proposals for this object. The NMS is an iterative algorithm comprising two steps to finally obtain the object detection result based on a list including all hypotheses. Therefore, in each iteration, the NMS selects the object hypothesis with the highest predicted confidence score and adds it to the output list. Then, the associated estimated bounding box is compared to all others. In the case that the two inspected boxes are overlapping,



(a) Radar targets as input target list. (b) Object hypotheses as detection output.

Figure 3.9: Radar object detection: The left side illustrates the input data, that is the unclassified radar targets provided by the sensor. The right side visualizes the output of the radar object detector, hence the various classified object hypotheses modeled as 2D bounding boxes and colored according to their class. In order to rate this detection result, the ground truth is depicted in black.

the hypothesis that belongs to the other bounding box is removed. The NMS in this thesis applies the Intersection over Union (IoU) metric [Jac12] to measure the overlap of two bounding boxes. Therefore, the IoU value is determined and once it exceeds a specific threshold, the respective object hypothesis is rejected. Since the IoU is later applied to evaluate the bounding box estimation of the radar object detection system, Section 5.1.2 introduces this metric. The described two iteration stages are repeated as long as all object hypotheses are processed. Figure 3.8 visualizes the output of the object extraction unit after the NMS procedure for a single object. The left side shows all proposals that belong to the object of interest. The right side displays the final radar detection of this object after performing an NMS. As a result, the bounding box with the highest score is selected, which is the most probable hypothesis for this object, whereas all the others are discarded. Compared to the true object, the final detection is quite precise, since the object is correctly classified as *car*, and the estimated bounding box is almost identical to the ground truth box.

The object extraction is the last stage in the pipeline of the radar object detection system and summarizes the intermediate outcomes of every patch to a final result. Hence, this process is the counterpart of the patch proposal, which is the first procedure in the pipeline. The patch proposal divides the complete input radar

target list into several small and independent regions of interest. The processing of theses patches runs in parallel and each patch provides a particular object proposal. Finally, the object extraction merges all this information to produce one detection output for the entire input target list. Figure 3.9 visualizes the input data, thus, all radar targets of an entire measurement cycle, and the predicted object hypotheses as output of the radar detection system. Hence, for every object the detector outputs an amodal 2D bounding box and its classification information, as well as the segmented radar targets which the bounding box estimation is based on. In summary, the object detection result for the real-world scenario, which is shown in Figure 3.1, is really satisfying. Especially those bounding box estimates for objects with sufficient measurement targets at distinctive spots, as for example on wheel houses or on the facing sides, are quite accurate. The estimated heading of the object located in the upper left corner is distorted compared to the object label. This is due to the fact that the segmentation module misclassifies one radar target, which is actually a clutter measurement but fits well with the other two reflections. Nevertheless, taking all these targets into account in bounding box estimation, this constitutes a reasonable solution. This example demonstrates the effect of a single noisy measurement on the overall result. Since the measured target list often is very sparse, individual radar targets may have a significant impact on the prediction of an object. However, these kinds of clutter reflections do not occur in each measurement cycle in such an unfortunate way, which means that in previous and following time steps the extracted hypothesis for such an object may be much better. For the purpose to overcome the issue that single outliers considerably deteriorate the perception performance, a filtering of the object proposals over time is applicable. Therefore, Section 3.3 introduces an appropriate measurement model to simultaneously process all radar object hypotheses in a multi-object tracking system. This technique improves the radar object perception, as it becomes much more resilient and even more accurate.

The following is a short analysis focusing on a few more details on the interaction of all modules in the radar object detection system. Figure 3.2 shows the three main components of the object detector, that is classification, segmentation and bounding box estimation. Each of them has a specific task and the respective interim results may directly affect the final object detection result. Firstly, the predicted classification information establishes the basis for further object detection. For one thing, this module determines the category for the object of interest. In addition, the predicted class is an important global feature for the segmentation and bounding box estimation networks. Consequently, a false classification may already cause the whole object detection to be incorrect. Next, the segmentation has a considerable impact on the quality of the estimated bounding box. The fact that only radar reflections which are assigned to the object of interest are relevant for the estimation of a bounding box underlines the significance of that stage. And ultimately, the bounding box estimation predicts the localization of an object and its extent. Since for every class a template is provided, the classification information has an effect

especially on the estimated size. In contrast, the segmentation result has an influence on the estimation of position and orientation, as well as on the residual size regarding the predefined anchor box. That brief discussion emphasizes once again that all core modules have to perform at their best for an optimal overall object detection result.

3.1.3 Network Architecture

The radar object detection systems consists of neuronal networks with different subtasks, namely, classification, segmentation, residual center estimation and bounding box estimation. The architectures of these networks are similar to the presented ones in [QLW+18]. In contrast to the detection pipeline of this thesis, the Frustum PointNets for 3D object detection requires lidar and camera data as input. Hence, a 2D Convolutional Neural Network (CNN) object detector is essential to classify the objects of interest as well as to identify their position in the image. This object detection is used for the following frustum proposal to find the respective objects in the 3D lidar point cloud. In contrast, the radar object detection system proposed in this work solely processes radar data to detect objects in a radar target list. Therefore, the object detector has a different processing chain, especially the procedure for generating regions of interest and the classification process differ significantly from Frustum PoinNets. The radar object detection system of this thesis can be designed with an underlying PointNet [QSMG17] or PointNet++ [QYSG17] architecture. Both variants are presented and their performance is later compared with each other.

Radar PointNet

First of all, the PointNet network architecture is introduced. Figures 3.10 - 3.14 visualize the architecture for all radar object detector modules. In order to classify a radar target list in a region, a classification network based on [QSMG17] is deployed. Figure 3.10 shows the corresponding network architecture. The special feature of this network is that it allows a classification of a complete radar target list without prior transformation directly on radar targets. In contrast, the Frustum PointNets, realizes the classification using a CNN camera detector on an RGB image. In order to process all information of an radar target, the input is adjusted to four dimensions compared to the classification network in [QSMG17]. Further, the classification network of [QSMG17] includes two transformer networks for normalization of input data and local features. In [QLW+18], a similar network architecture is used for instance segmentation as in the classification network of [QSMG17]. But, [QLW+18] describes that the transformer networks for normalization purposes are in principle not necessary, because the point cloud in the region of interest, referred to as frustum

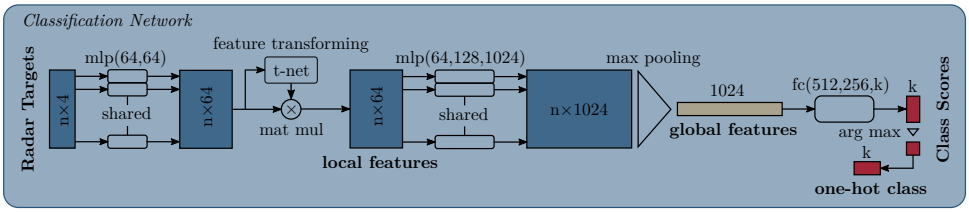


Figure 3.10: Architecture of the classification PointNet: The network processes any number of radar targets and predicts the classification information for the radar target list as one-hot class vector.

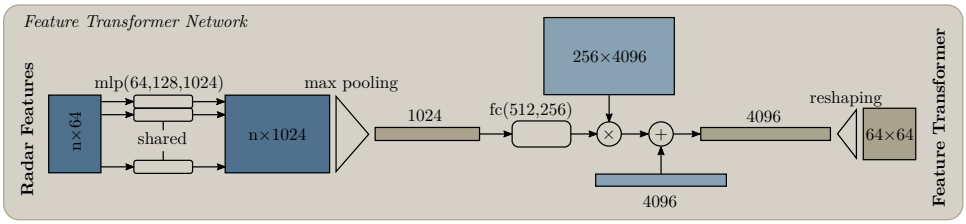


Figure 3.11: Architecture of the feature transformer T-Net: The network outputs a feature transformation matrix to generate local features based on extracted radar features from the radar targets.

there, is already normalized. This statement also holds for radar data inside the generated patches, since the above described rotation normalizes the radar target list. However, the radar object detector does not renounce the feature transformation in the classification network which aligns the feature space. Therefore, an alignment network realized as a T-Net predicts a feature transformation matrix based on extracted target features. Figure 3.11 illustrates the network architecture of this T-Net. Next, the set of features is normalized by a matrix multiplication with the predicted transformation matrix resulting in local target features. This technique allows normalizing in particular the information extracted from the radial velocities and RCS of a radar target list. Experiments in Section 5.1.3 evaluate the effect of this transformation and show the detection results in case that additionally a T-Net for input data is used. Both transformer networks are initially proposed in [QSMG17].

In order to realize the classification on a target level, the network deploys several shared Multi-Layer Perceptrons (MLP), one for each of the n input radar targets. The subsequent *max pooling* layer generates a set of global features from the radar targets for classification purpose. After applying various Fully Connected (FC) layers,

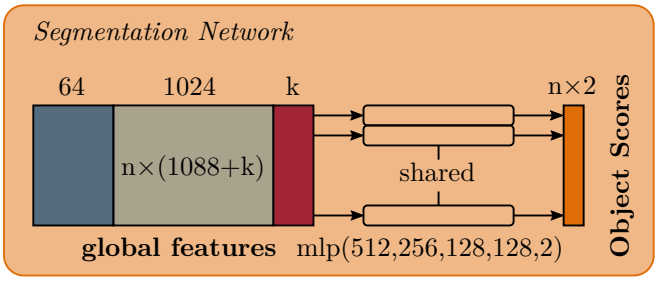


Figure 3.12: Architecture of the segmentation PointNet: This network performs radar target segmentation by predicting the classification information for each target. The object scores are determined on the basis of the local features and the global features as well as the classification information for the whole radar target list.

the network provides the classification scores in form of *logits*. With the intention to get probability values $p_{cls,c}$ for each class $c \in \mathcal{C}_{cls}$, the *softmax* function [Bis06] is applied to normalize the output vector. This transforms classification scores into a probability distribution, where all components are normalized to the range $(0, 1)$ and add up to one. Softmax, also known as the normalized exponential function, is a generalization of the binary logistic function to k dimensions, and it is defined as

$$\sigma(\underline{z})_c = \frac{\exp(z_c)}{\sum_{j=1}^{k} \exp(z_j)}, \tag{3.6}$$

for all classes $c = 0, \ldots, k-1$ and classification scores $\underline{z} = (z_0, \ldots, z_{k-1}) \in \mathbb{R}^k$. For further processing, the network encodes the classification information by applying one-hot encoding. This means, the predicted class with the highest score, respectively probability, is selected as the classification result and a one-hot vector is generated with value 1 for the most likely class and value 0 for all other components. Since the succeeding modules employ this one-hot vector as a global feature, this once again clarifies the importance of a correct classification for the following processing chain.

In order to extract features for the segmentation task, the network with architecture shown in Figure 3.12, is applied. This neuronal network initially concatenates all the local and global features as well as the predicted output of the classification network. As a result, the global features in the segmentation network consist of local features which are target specific, and several others that are the same for each input radar target. This procedure is essential because every radar target is classified individually during the segmentation, and therefore the consideration of individual features is meaningful. After various shared MLPs, the network outputs

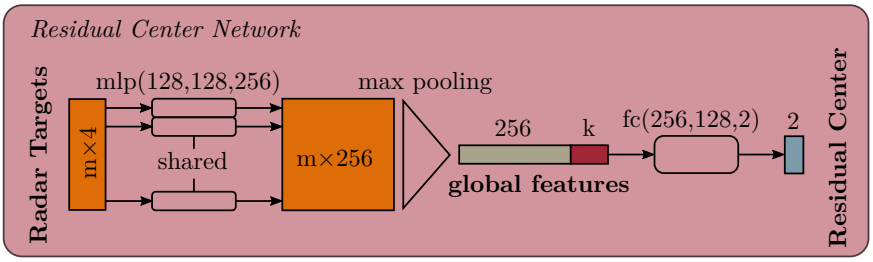


Figure 3.13: Architecture of the residual center T-Net: This network estimates the residual center based on all segmented radar targets which belong to the object of interest. The output is predicted using features extracted from the segmented radar targets and the classification information of the complete radar target list.

object scores for each radar target indicating whether a reflection belongs to the object of interest or not. As already explained, the usage of a softmax function allows converting the segmentation results into comprehensible probability values.

After masking of the segmented radar targets, the residual center networks predicts the true center of the regarded object. Figure 3.13 visualizes the network architecture of the corresponding T-Net. For the segmentation, all masked radar targets are forwarded to three shared MLPs. Subsequently, the max pooling operation generates features which are extended with the previously predicted classification information. Based on these global features, different FC layers finally estimate the residual position to the real center of the object. More precisely, this neuronal network provides the difference between the masked object targets' centroid and the object's true center position to transform all radar targets into the object coordinate system.

The architecture of the neuronal network for bounding box estimation, which is depicted in Figure 3.14, is almost the same as for the prediction of the residual center. Merely for the generation of much more global features, an additional MLP is added before the max-pooling operation is executed. The network itself extracts most of the global features and again the predicted class of the classification network is attached to these features. Further, the following FCs layers generate multiple output values, which are exactly so much that all parameters of the 2D bounding box can be calculated. Accordingly, this network does not provide the final bounding box parameters. Section 3.1.2 introduces the object extraction that implements a strategy to merge all the estimated values to the actual bounding box comprising center position, orientation and size, before extracting the ultimate object detection.

As already mentioned, a PointNet allows to process an entire radar target list without

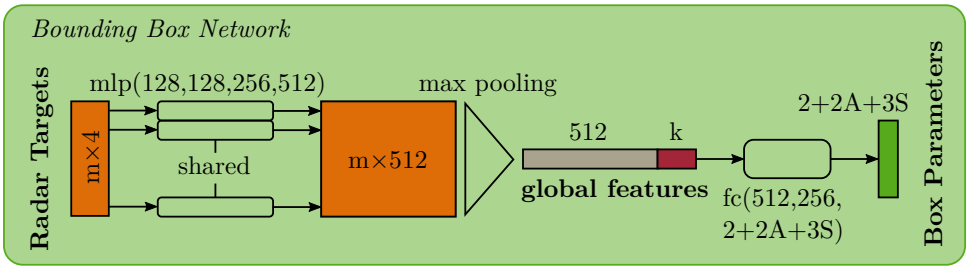


Figure 3.14: Architecture of the bounding box PointNet: The neuronal network determines all the information to calculate parameters for the final 2D bounding box. The box parameters are calculated based on features extracted from the transformed radar target and the classification information of an entire radar target list.

any transformation beforehand. The network architectures in Figures 3.10 - 3.14 clarifies that the shared MLPs and the max-pooling operation are the key element of PointNet. The particular MLP ensures that the neuronal network extracts features from every individual input radar target. Since each MLP of a layer is equal and consists of the same learned weights, the network examines all radar targets under the same conditions. The max-pooling operation guarantees that the order of input targets is irrelevant, as well as radar targets with identical information not having a major impact on the feature extraction. In other words, regardless of the order, max-pooling layer always extracts the same features, exactly those of maximum value with respect to all n radar targets. Whenever identical input radar targets occur more than once, it results in the same feature values for those radar targets. Identical radar targets mean that the targets have exactly the same value in each of the four dimensions. Since the maximum value of those global features is equal, max-pooling extracts the corresponding feature solely one time. More information and also a theoretical analysis on the max-pooling function is given in [QSMG17].

Radar PointNets++

Next up, the PointNet++ network architecture for radar object detection is described. As proposed in [QYSG17], PointNet++ realizes a hierarchical network with density adaptive PointNet layers. Figures 3.15 - 3.17 show all the elements of the PointNet++ architecture that are different to PointNet. Figure 3.15 visualizes the classification network with an underlying PointNet++. The neuronal network consists of multiple Set Abstraction (SA) Multi-Scale Grouping (MSG) layers and a final SA Single

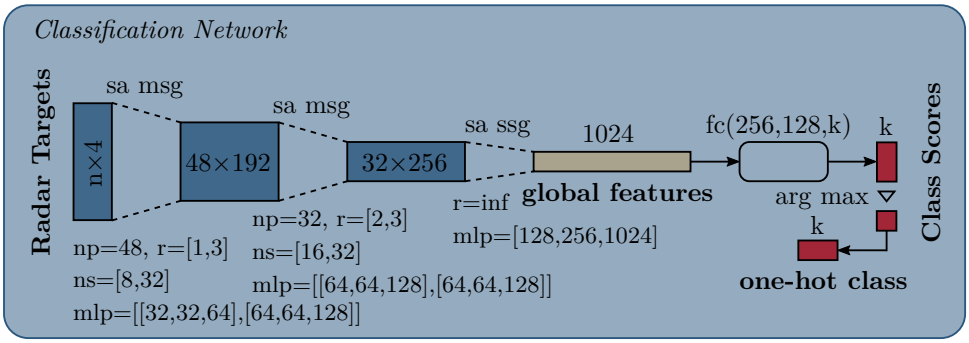


Figure 3.15: Architecture of the classification PointNet++: The neuronal network generates global features on the basis of the input radar targets. The predicted ouput is the classification information for the radar target list and is encoded in a one-hot class vector.

Scale Grouping (SSG) layer to generate global features. The MSG allows to capture multi-scale patterns by extracting features from different scales and concatenate them to a multi-scale feature. Thus, this layers basically contains several MLPs, but primarily performs some grouping operations. Subsequently, the determination of a predicted class is the same as in PointNet by forwarding the global features to several FC layers and performing one-hot encoding which results in a class vector.

For the segmentation of radar targets, the network architecture that is illustrated in Figure 3.16 is applied. In order to generate global features, the network extends features extracted from classification unit with the resulting one-hot class vector. Next, several feature propagation (FP) levels with fully connected layers aggregate information to get target features for all input targets. In contrast to a SA layer, which subsamples the input target set, the FP layer propagates features from downsampled targets to the original radar targets. In comparison to PointNet architecture, the FP layers replaces some of the MLPs, however within this layer there are again some MLPs. Lastly, the network forwards all upsampled features to shared MLPs for predicting the final object scores. The segmentation scores distinguish which target belongs to the actual object. The network for residual center estimation is same as in the PointNet version. Figure 3.13 shows the corresponding network architecture.

The architecture of the bounding box network is similar to the one for classification purpose. Figure 3.17 shows the corresponding architecture. This neuronal network only consists of SA layers with SSG instead of MSG. Nevertheless, the network achieves hierarchical point set feature learning for the radar target, but the SSG

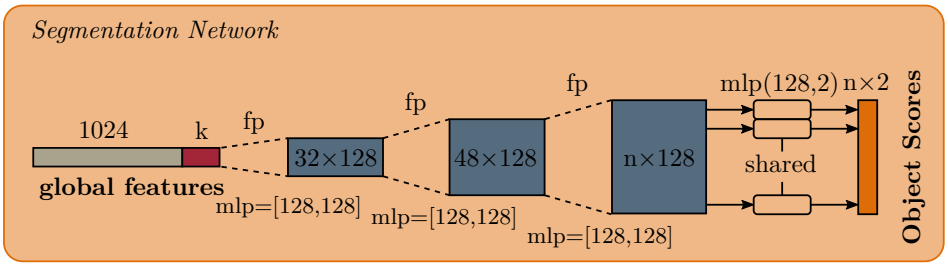


Figure 3.16: Architecture of the segmentation PointNet++: This network processes the global features and the classification information of the radar target list to determine the scores for segmentation.

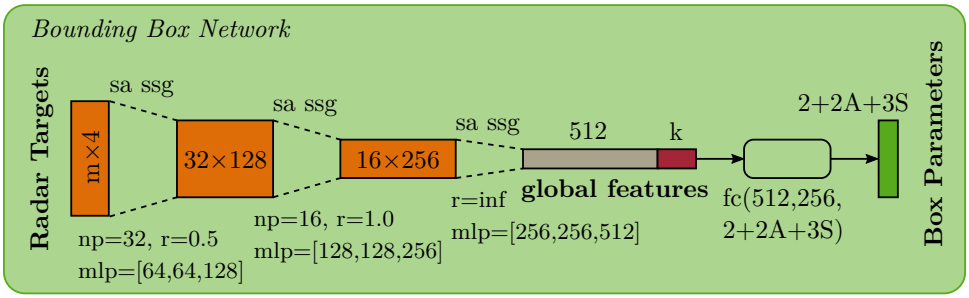


Figure 3.17: Architecture of the bounding box PointNet++: The network predicts all information for the calculation of the 2D bounding box. The final parameters are estimated using features extracted from the segmented and transformed radar targets and the classification information of the entire radar target list.

accomplishes only single-scale grouping in each SA level. After generating all the global features, the predicted classification information is attached to these and the output of the FC layers supplies all the required parameters to construct the ultimate 2D bounding box. That is the same procedure as in the PointNet version. More information on PointNet++ and all the network architectures as well as the differences between PointNet++ and PointNet is described in [QLW+18; QYSG17].

3.1.4 Training Process

The training procedure using deep learning is an optimization process, where parameters are optimized on the basis of a loss function. The training of the neuronal networks in the radar object detection system follows the strategy to simultaneously optimize all networks by applying multi-task learning [Car97]. The general idea of this technique is to use one model, that includes multiple individual tasks, and to train this model for parallel learning of multiple tasks while sharing the information for all tasks. This means for the radar object detection system that the model must learn how to detect objects in a radar target list. In order to successfully achieve this goal, several small tasks have to be solved, which are introduced in this context as classification, segmentation and the bounding box estimation module. Consequently, the idea is to share what is learned for each task with the other task to learn better for the overall goal, the complete object detection. This is a major advantage, which is the reason why in this thesis multi-task learning is applied and not each module is trained individually. For the realization of multi-task learning, a multi-task loss for the optimization is required. In [QLW+18], a suitable multi-task loss optimization is presented, which is also used in this thesis. In order to train all modules of the radar object detector, the loss requires an extension for the patch classification. For this purpose, an additional loss function for the classification part is added to the total loss as presented in [QSMG17]. Mathematically, the multi-task loss is specified as

$$\begin{aligned} L_{multi-task} &= w_{cls}L_{cls} + w_{seg}L_{seg} + w_{bbox}L_{bbox}, \\ L_{bbox} &= L_{c1-reg} + L_{c-reg} + L_{h-cls} + L_{h-reg} + \\ &\quad L_{s-cls} + L_{s-reg} + w_{corner}L_{corner}, \end{aligned} \tag{3.7}$$

where this loss combines several individual weighted losses by adding them together. Altogether, the multi-task loss consists of three major parts, the loss for patch classification, the loss for segmentation of a radar target list and the loss for bounding box estimation. During the training process, the associated weighting parameters $w \in \mathbb{R}$ allow to control the degree of optimization for the respective three main modules. In Table 3.1 all parameters of the multi-task loss from Equation (3.7) are listed and described. For optimizing the model parameters using this multi-task loss, a higher weight increases the influence of the individual loss on the total loss.

Loss	Weight	Description
L_{cls}	w_{cls}	loss and weight for classification
L_{seg}	w_{seg}	loss and weight for segmentation
L_{bbox}	w_{bbox}	loss and weight for bounding box estimation
L_{c1-reg}	–	loss for residual center regression
L_{c-reg}	–	loss for center regression of the bounding box
L_{h-cls}	–	loss for classification of the bounding box heading
L_{h-reg}	–	loss for residuum estimation of the bounding box heading
L_{s-cls}	–	loss for classification of the bounding box size
L_{s-reg}	–	loss for residuum estimation of the bounding box size
L_{corner}	w_{corner}	Loss and weight for corner estimation of the bounding box

Table 3.1: Multi-task loss parameters: Overview of all components and their corresponding weights used in Equation (3.7) for the total loss.

Since a bounding box estimation is only useful if a region of interest is classified as containing an object, the weight w_{box} is also used to control whether a bounding box estimation is performed or not. Consequently, the weight is set to zero for clutter patches, and it remains unaffected for a patch where an object is classified inside it.

The corner loss is a novel loss for regularization that is first suggested in [QLW+18] for 3D bounding boxes estimation. Since the radar object detection system predicts 2D bounding boxes, the corner loss is modified and mathematically calculated by

$$L_{corner} = \sum_{s=1}^{n_S} \sum_{a=1}^{n_A} \delta_{sa} \min \left\{ \sum_{c=1}^{4} \|P_c^{sa} - P_c\|, \sum_{c=1}^{4} \|P_c^{sa} - P_c^*\| \right\}, \tag{3.8}$$

where it ensures an optimized estimation of the 2D bounding box using the IoU metric. The corners for a predicted box are calculated using the estimated center, heading and size of the bounding box. Since these three parameters and the corners depend on each other, the corner loss allows to regularize the multi-task training for center, size and heading simultaneously. As described in [QLW+18], the corner loss is the sum of all distances between the four corners of a predicted bounding box and the ground truth box. For this purpose, the loss calculator constructs $n_S \times n_A$ anchor boxes, where n_S and n_A are the number of size templates and angle bins, respectively. In order to ensure that the constructed anchor boxes are comparable to the predictions, these boxes are translated to the estimated bounding box center. The corner of a predicted box is denoted with P_c^{sa}, where s, a, c are the indices of size and heading class, as well as a predefined order of the corners, accordingly. Moreover, P_c denotes the corners of the ground truth box, and P_c^* the ones for the flipped ground truth box. Determining the minimum of the distances between the predicted

box and the ground truth box or between the predicted box and the flipped box ground truth box, avoids a large penalty during optimization for a flipped heading estimate. The function δ_{sa} is a mask that equals one for the ground truth size or heading and zero otherwise. It selects the required distance term for the corner loss.

Another important aspect is the choice of the actual losses which used for implementation of the training. In this thesis, the training process uses the tasks for classification and segmentation use softmax [Bis06] with cross-entropy loss [Mur12]. Whereas, the regression tasks apply smooth-l_1 loss [Hub64], also called huber loss. After defining the loss function, the training process applies this loss for the optimization of all PointNets using end-to-end training. In order to find optimal model parameters, the training applies the Adam optimizer [KB15]. The Adam optimization algorithm is an extension of the classical stochastic gradient descent to iteratively adapt the weights of a neuronal network using training data. For the purpose of speeding up the learning process, this thesis uses the Adam optimizer with an initial learning rate and incremental learning rate decaying by half after a specific number of iterations.

A common technique for accelerating the whole training procedure, which is also used in this thesis, is batch processing. This method takes advantage of the parallel computing capability offered by GPUs. It involves simultaneously processing multiple batches of training data, where every batch contains exactly one training sample. The batch size depends on the amount of data, that is the cardinality of an observed target list, the network architecture and the real memory of the installed GPU. Thus, all these hyperparameters define the number of training samples to process before updating the model parameters. Furthermore, the training procedure performs batch normalization for all layers except the last classification and regression ones to speed it up even more. In doing so, batch normalization uses an initial decay rate which is gradually increased up to a fixed decay rate using momentum. Next, for regularization purposes, drop out layers are used during training in the ultimate FCs of the classification PointNets and in the last MLPs of the segmentation networks.

Another hyperparameter is the number of epochs which specifies how often a dataset has to be passed through during the training. An epoch consists of multiple batches and covers the event that every training sample is processed once. Then the model parameters are updated according to the acquired knowledge. For monitoring and assessment of the training, the current model parameters are analyzed after each epoch by measuring the performance on validation data. In case that the model performs poorly or the trained parameters tend to overfit, it is possible to react at this point and adjust some design parameters. Eventually, training is stopped when the observed metrics are no longer improving and the model is providing satisfactory results on the validation dataset. Finally, the training freezes all parameters and the optimized model for the radar object detector is ready to be evaluated on test data.

3.1.5 Discussion

The radar object detector which is proposed in the previous sections is a system to process a raw radar target list and detect objects of different classes. This method is inspired on the idea of Frustum PointNets [QLW+18], which is an approach based on PointNet [QSMG17] or PointNet++ [QYSG17] to perform object detection in lidar data combined with information from a camera image. Comparing the pipeline of the radar object detection system in Figure 3.2 with the Frustum PointNets for 3D object detection [QLW+18], as well as the network architectures in Figures 3.10 - 3.17 with the PointNet [QSMG17] and PointNet++ [QYSG17] architectures, several similarities can be observed. However, it must be emphasized that the goal of this thesis is not to find a new type of network architecture for object detection, but to propose a technique for detecting objects by directly processing a radar target list.

As already mentioned above, the motivation of this thesis consists of three main aspects. Firstly, the detection system is intended to detect objects in radar data exclusively, without combining information from any other source. This is a significant difference to the method in [QLW+18], which extracts and combines information from a camera image and lidar point clouds. Secondly, the radar object detector must process the target list directly, without the need to transform the data into another format. This is an important difference to methods existing in the literature, which initially transform the radar targets into a fixed format, such as radar grid maps, and then apply image processing algorithms to identify objects. And thirdly, the radar object detection system has to be real-time capable, so that it can be deployed on a real system for environment perception. Section 3.2.2 gives a definition of real-time capability with respect to the radar object detector. The evaluation in Section 5.1 demonstrates that the presented technique for object detection in radar data is feasible and that it is also real-time capable. In conclusion, the proposed radar object detection system combines, extends and adapts elements from established methods in a sophisticated way to fulfill all these requirements. Consequently, all the objectives are achieved and this approach is distinguishable from existing methods.

3.2 Implementation Details

The radar object detection system has to meet certain requirements. In order to realize this complex system in a real system, some practical implementation details are necessary. These are especially important when using real data or to ensure real-time capability of the algorithm. Firstly, the detection system has to cope with a varying number of radar targets which a sensor provides per measurement cycle.

Secondly, the processing of all measurement cycles must be real-time capable. With respect to this, the reduction of computational complexity is a remedy. Thirdly, in real operation mode, the detection system only outputs hypotheses with a specific confidence score, and it must be decided which ones should actually be processed further. Finally, a reliable detection in real operation is indispensable. For this reason, the detection system focuses on dynamic objects, as these can be well identified with a radar sensor due to the occurring radial velocity. The following sections explain all these challenges in more detail and propose a suitable solution for each of them.

3.2.1 Input Data

In theory, a PointNet principally allows an arbitrary number of input radar targets. In case of the radar object detection system proposed in this thesis, there are two situations where a varying number of radar targets occur and affect the processing chain. In fact, each measurement cycle may consist of a different number of radar reflections depending on the current environment. Since each radar target generates an individual patch, the number of targets defines the quantity of input patches. Furthermore, every patch may contain a different number of radar targets depending on how many reflections are measured in this region of interest. One possible solution is to perform object detection for one patch after the other. Then, the amount of input radar targets is equal to the number of required iterations and the batch size is the same for each iteration. This eliminates the problem of a varying radar targets number per patch by nature, because only one radar target list is forwarded to the network and the PointNet allows any number of targets by architecture design. However, the sequential processing of all radar patches is computationally quite expensive as the neuronal network is reinitialized before each execution. Another and much more efficient method is to run the processing of all patches in parallel by using multiple batches. In this case, the network architecture requires that all batches comprise the same number of radar targets. Since one patch corresponds to exactly one batch, the number of targets per patch varies quite a lot during the same measurement cycle. But, this case practically never occurs. Thus, the radar target list have to be adjusted into an appropriate data format to ensure that the input dimensions of all the batches are equal without modifying the target representation.

With the intention of getting a representation that allows batch processing there are two techniques: either downsampling or upsampling the input targets to the same quantity. This means that for all regions of interest the number of radar targets has to be fixed. It is necessary to remove or add targets within the patches to achieve this. Removing radar reflections has the major drawback that the information of those targets is lost and the neuronal network may extract worse features. In contrast, when adding radar targets it is generally assumed that the extracted information is

also falsified due to the fact that additional targets may generate new global features or weight existing ones differently. But when using the PointNet architecture, it is possible to upsample the input targets to any number without changing the extracted features. This is comparable in its basic idea to padding methods in computer vision, for example zero padding and reflection padding. The goal of the upsampling process in the radar object detector is to increase the input radar target list to a specific quantity to ensure that every patch consists of the same number of targets. There are two techniques to make this happen. The first approach is to add radar targets with zero information, which is a similar method to zero padding. For this purpose, so many radar targets are synthetically created such that the desired number of targets is achieved. The values for each dimension, that is spatial coordinates, radial velocity and RCS, are set to zero. Such a radar target never occurs in reality, since the radar sensor is located at position $(0, 0)$. As a result, the upsampling generates a new radar target which may affect the following feature extracting. The alternative technique is to reproduce already existing radar targets. That approach is close to reflection padding. Therefore, the method duplicates arbitrary radar targets including their values in the respective dimension until the desired number of targets is reached. Section 3.1.3 already discusses that the identical radar targets do not have an influence during extracting global features, because of the preceding max pooling layer as key element. In summary, reproduction of existing radar targets does not affect the resulting features. Therefore, this is the best technique to obtain an equal number of input radar targets for all patches, and hence, equally sized batches.

The reproduction technique is not only necessary for batch processing, but rather for realizing the masking process during the segmentation process. The bounding box estimation module only processes the m segmented object radar targets instead of all n input targets. With the purpose to avoid a reallocation of GPU memory, the implementation does not reduce the radar target list to size m. Instead, the masking randomly replicates radar targets assigned to the object of interest as long as the segmented radar target list comprises n reflections again. When the batch size is one, it is possible to skip this step and simply select the radar targets belonging to the object. But for batch processing, a fixed size of the target list is necessary. Therefore, this masking implementation is recommended. However, if batch processing is performed, the upsampling method is necessary to ensure a fixed number of radar targets during the segmentation over all batches. As already mentioned, the proposed technique to upsample radar targets does not affect any global features, because always a max pooling layer is used for the feature extraction.

3.2.2 Real-Time Capability

By definition, a technical system is real-time capable if it is able to reliably deliver certain results within a predefined period of time. Thus, in case of the proposed radar detection system, the requirement is that when radar reflections from a new measurement cycle are received, the system must have completely finished processing the previous measurements and the object detection result must be available. As a consequence, the measuring frequency of the deployed radar sensor determines the specific time interval which defines the real-time capability of the system. Section 4.1 presents the technical specifications of the radar sensor which is used in this thesis. That sensor has a measurement interval of about 70ms to 80ms, which corresponds to a frequency of $12 - 15$Hz. Accordingly, the object detection system has a maximum time of 70ms for processing the entire measurement cycle. However, it has to be considered that during this time also the decoding of input data from the sensor must be finished. For this reason, it is recommended to budget a small time buffer. Section 5.1 presents an expressive evaluation that investigates the calculation time as well as the real-time capability of the radar detection system to clarify this issue.

With the aim to achieve a real-time capable radar object detection system for a real application, two implementation tricks are applied. The first one realizes a parallel processing of all patches to accelerate the entire object detection pipeline. After the patch proposal module, all following components up to the object extraction are independent of each other and may be performed in parallel. In this implementation the calculation of dependent components, namely, the patch proposal and the object extraction, is performed on the Central Processing Unit (CPU) and all others on the GPU. Since a GPU is designed for parallel calculations, the idea is to perform all independent modules simultaneously. The realization of this uses the advantages of batch processing. Normally, only training of deep learning models applies batch processing to speed up and parallelize the training iterations. Later in the application, the inference uses this trained model for the prediction with a batch size of one. For example in a camera application, the model is able to handle multiple images in the training process, but during inference there is normally only one image available at the same time. However, the special processing chain of the radar detection system with independent patches in a measurement cycle enables batch processing during training and inference. That means that multiple batches allow processing several patches in parallel. Then, a question that arises is how to choose the batch size. In this thesis the maximum number of radar targets per measurement cycle is sparse, and comprises a maximum of 250 targets. This limitation is caused by the radar sensor employed in this, which is introduced in Section 4.1, as this sensor does not produce more targets per measurement cycle. Accordingly, the maximum number of generated patches corresponds to this value, and thus the required batch size. For better performance on the GPU a power of 2 should be chosen. As a result, the

usage of such a radar sensor recommends a batch size of 256. Consequently, the six additional batches are filled with dummy data, that is data in form of zeros or through radar target duplication. Since the radar sensor does not guarantee a fixed number of measured reflections, the missing batches are filled with dummy data anyway. In summary, this reproduction technique ensures a constant batch size with small effort over all the radar measurement cycles to allow parallel batch processing.

The second issue concerns the realization of the masking process in the segmentation component. In order to grasp this, it is necessary to briefly explain the implementation of the masking that is proposed in [QLW+18]. This method gathers object targets according the predicted segmentation mask, more precisely the object scores which predicts the segmentation PointNet. For this purpose, a loop randomly draws object targets until the desired number of radar targets is reached. This implements a method to increase targets randomly, where especially randomness is an important characteristic during the training process. However, the number of loop iterations is not deterministic, but depends on the input radar target list and the segmentation result. During training this does not matter, because computing complexity is not a critical factor. But in the application at inference, it is essential that the calculation time is deterministic and especially constant. For this reason, in application mode the masking algorithm is approximated such that the calculation time is almost fixed. On that account the masking does not randomly sample radar targets according to the predicted segmentation scores, but masks the object targets using a single multiplication operation. Therefore, the segmentation mask is transformed to a binary mask that enforces all scores greater than 0.5 to be corresponded to 1, all other to 0. Next, this mask is multiplied with the input radar targets, which causes all segmented targets to remain unchanged and get a value 0 in each dimension. Consequently, the approximation maps all non-object radar targets to a dummy target $(0, 0, 0, 0)$ which is mathematically valid. However, this radar target never occurs in practice, because the real radar sensor is exactly installed at the position represented by this target. Due to the fact that no radar reflection with these values occurs in the training dataset, the neural network will neither see this radar target during the training nor extract corresponding features. As a consequence, that target does not create any features in the bounding box estimation PointNet. Further, the max pooling layer ensures that multiple occurrence of the dummy radar target do not have any influence on the feature extraction. Hence, this is a valid approximation technique which does not affect the final radar object detection result in any way.

All of these suggested techniques constitute an option to efficiently operate the radar detection system in a real application. These methods are designed in such a way that no restrictions are imposed on their usage. Nevertheless, for general evaluation the batch size is set to one and the approximation during masking process is deactivated. Consequently, no upsampling of the input radar target list is necessary, although it is already shown that mathematically this does not affect the feature generation. With

the focus to speed up the training process, the batch size will be as large as possible, and therefore in training mode the proposed upsampling trick needs to be performed to realize the usage of multiple batches during training. Where performing batch processing and the approximation of the standard masking method are required to ensure a real-time capable system with critical time constraints in operating mode.

3.2.3 Computational Complexity

An important objective to be achieved in the scope of this thesis is that the developed radar object detection system successfully operates on a real-world system for environment perception. Since the PointNet++ version of the object detector is not real-time capable, only the architecture based on PointNet is deployed on the experimental vehicle. In order to reduce the computational complexity, two tricks are applied to approximate the required calculations in real operation. With the purpose to reduce the number of arithmetic operations, the feature transformer in the classification networks is deactivated. The comparison in Section 5.1.3 about different ways of implementing transformer networks reveals that this approximation only marginally deteriorates the overall detection results. Another technique to tremendously reduce the computational complexity is switching to a half precision floating point format, also called float16 or FP16. Instead of performing all calculation with a single precision floating point format, also named float32 or FP32 as it occupies 32 bits in the computer memory, the neuronal networks only perform 16 bits calculations. The approximation results in a noticeable acceleration of all networks, but still provides sufficient accuracy. Although Section 5.1 does not present a direct comparison of the different number formats, this statement is confirmed by successful application on a real experimental vehicle in automated operation mode.

3.2.4 Object Extraction

The output of the radar object detection system is one hypothesis for each object comprising a 2D amodal bounding box and a confidence score for each of them. Since this score can be interpreted as detection probability, it is applicable for filtering the object hypotheses. For every object detection, the final score is determined using Equation (3.5), which combines the confidences for classification, segmentation and bounding box estimation. Then, the object extraction filters all hypotheses in such a way that only detections are output whose confidence score exceeds an application specific threshold. In practical operation, this allows to control whether more reliable predictions or rather uncertain ones are to be considered for further processing. While a higher threshold results in more reliable predictions, the detector produces

less false positives (FP), but more frequently true positives (TP) may be lost. In contrast, a lower threshold value means that more TP are preserved, however, it generates more FP. Finally, it is a decision depending on the actual application field.

3.2.5 Dynamic Objects

The radar object detection system proposed in this thesis has one key characteristic, in fact, it is just able to detect dynamic objects, or more precisely, objects in motion. However, this is not caused by the developed design of the object detector itself, but rather due to the quality of input radar data. The employed sensor delivers with a maximum number of 250 radar targets relatively few radar reflections. However, automotive radar sensor are mainly good at measuring reflections from moving objects by exploiting the Doppler effect. As described in [WHLS15], for measuring multiple radar targets, the sensor requires a suitable separation capability in at least one of the dimensions distance, azimuth angle and relative velocity. In general, particular importance is attached to separability at distance and radial velocity. In the case of stationary objects, the separability via radial velocity cannot be performed. Consequently, azimuth angle and range can be used for the separability and the number of radar targets depends only on the extent of the object. However, as soon as objects move, another dimension for separability is available in the form of radial velocities. Furthermore, various Doppler effects occur on real objects, for example on tires of vehicles, on pedals of bikes or on arms and legs of pedestrians. That produces more peaks in the measured spectrum, and as a result more radar reflections are measurable per object. This argument again indicates quite distinctly what a strong feature the radial velocity represents for object detection. Altogether, these circumstances lead to the fact that without a sensor fusion and without a merging of several radar measurement cycles, a detection of stationary objects is not realistic, respectively not possible at all. Accordingly, this thesis exclusively aims at the detection of dynamic objects which are in motion. Nevertheless, it is possible to model stationary objects in the environment perception by using a multi-object tracking algorithm with a corresponding motion model, but under the condition that these objects are moving beforehand and the radar object detector recognizes them.

3.3 Radar Object Tracking

The detection of objects is an essential step for environment perception, but for a robust object recognition, it is advisable to track these object hypotheses over time. One common approach to achieve this are multi-object tracking methods, that

iteratively process measurement information obtained from objects in the surrounding. As already described in Section 2.3, an object tracking algorithm performs two basic tasks, the prediction and the update step. As a consequence, there are two options for improving the overall tracking result, either enhance the prediction technique or improve the update method. For a more accurate prediction, the challenge is to find a way to more realistically model the motion of objects between two observed time intervals. Therefore, [DGD18] proposes an extension for modeling vehicle interaction with respect to its static and dynamic environment during the prediction step of a standard Labeled Multi-Bernoulli (LMB) filter resulting in an enhanced prediction, and finally, this yields to an overall improved multi-object tracking performance.

In order to improve the update results, there are basically two approaches. Either to optimize the update calculation technique itself, or simply to provide more accurate observations as input data. This is a valid argumentation, since the further development of the sensor's technology improves the quality of measurements, but it is not necessarily granted that existing algorithms are able to handle this amount of data directly. Especially in the case of radar sensors, modern automotive sensors have a much higher resolution, and therefore provide much more information per measurement cycle. This thesis presents in Section 3.1 a novel object detection system to process high-resolution radar data with radar object hypotheses as an output, one for each detected object. Consequently, the second option is chosen and the tracking algorithm is enhanced by using improved object detections. Therefore, the predicted radar object detection results are forwarded into a multi-object tracking system, an LMB filter, as measurement input. Overall, the enhanced radar object observations are expected to provide much better multi-object tracking performance.

This section covers the topic of how the output of the presented radar object detection system can be processed in a multi-object tracking algorithm. For this purpose, a measurement model is proposed to incorporate the radar object detections, hence the bounding boxes after the object extraction, into the update step of an LMB filter.

Modeling Radar Objects

The radar object detection system proposed in Section 3.1 provides at most one radar object hypothesis for each object after the object extraction, thus as explained in Section 3.1.2 the output after the NMS. Since this is a prerequisite for the LMB filter, it can be used for multi-object tracking. Therefore, the tracking algorithm requires a measurement model to correct all predicted tracks based on these observations.

In addition to a tracking of the dynamic object state, the LMB filter is capable of estimating the static extent of an object[Reu14]. For this purpose, the state vector

from Equation (2.2) is extended by the length and the width of a track to become

$$\underline{x}^{\mathrm{T}} = [x, y, \theta, v, \omega, l, w], \tag{3.9}$$

where x and y is the center position and θ the heading, v is the velocity and ω the yawrate, as well as, the length l and width w of the track. However, the output of the radar object detection system is a set of classified 2D bounding boxes, each representing one object detection belonging to a different object. In order to represent a single measurement, the measurement vector from Equation (2.3) is adjusted to

$$\underline{z}^{\mathrm{T}} = [x, y, \theta, l, w], \tag{3.10}$$

where the vector includes the center position (x, y), the heading angle θ as well as the length l and the width w of a detected object. A comparison between state and measurement components reveals that the employed radar object detector is not able to directly measure all state components. However, to still obtain an estimate for each object state, a suitable motion model must be used. This thesis uses a Constant Turn Rate and Velocity (CTRV) model [SRW08] when processing radar object detections. Then, the LMB filter is able to reliably estimate the unknown states, because these can be derived from the measurable states. In this case, the velocity is derivable from the position measurements and the yaw rate from the heading angle measurements. An important step during the measurement update step is to calculate the residual between the actual measurement and the predicted measurement. In order to get the predicted measurement, the components for position, heading angle, length and width have to be extracted from the predicted object state. Eventually, it is possible to correct the prediction for a track which spatial distribution is represented by a Gaussian Mixture (GM). In other words, this measurement model allows to forward object detections generated from sparse radar target lists by the radar detection system proposed in this thesis into an LMB filter for tracking multiple objects.

Chapter 4

Radar Data

The last chapter deals with the entire radar object detection system itself to find objects of different classes in radar data. In contrast, this chapter covers the input of the proposed radar object detector, the radar data. Section 2.1.1 presents the signal processing of an automotive radar sensor and already describes that the introduced detection system of this thesis processes radar targets with the goal to generate object hypotheses based on multiple reflections. Therefore, radar data is required in the target format explained above to ensure an appropriate training and testing of the object detection system. For a detailed evaluation of the detector, this thesis examines detection results based on radar data at a target level from various origins.

This thesis has two objectives regarding the proposed radar object detection system. Firstly, the radar object detector needs to operate on a real system. For this purpose, the autonomous experimental vehicle of Ulm University [KNW+15], that is equipped with multiple high-resolution automotive radar sensors, is used as test platform. Secondly, to ensure that this approach is not limited on a particular platform or on radar data from a specific radar sensor, the object detection system is also evaluated on publicly accessible radar datasets. In order to address and evaluate both of these issues, appropriate radar data are required, which must be preprocessed accordingly.

Initially, Section 4.1 introduces the radar sensor which is deployed in the experimental platform, the vehicle of Ulm University. Section 4.2 discusses different datasets with radar data. However, not all of them are open to the public, and therefore only the accessible datasets are discussed in detail. One important outcome may already be anticipated, in fact that the publicly available radar datasets are not suitable to train models which are applicable on the real experimental platform. For that reason, this thesis introduces a self-generated dataset based on radar data which is delivered from the radar sensor described in Section 4.1 and collected using the experimental vehicle. Since datasets is an important topic when using deep learning, all datasets used in the evaluation are discussed in detail, especially the self-generated dataset.

(a) Experimental vehicle of Ulm University.

(b) ARS 408-21 long range radar sensor.

Figure 4.1: Radar setup: Three radar sensors facing forward are mounted on the experimental vehicle. This setup ensures an excellent sensor coverage of the vehicle's front area with an overlapping FOV at near range and a complementary FOV at the far range.

4.1 Radar Sensor

The experimental vehicle of Ulm University [KNW+15] is a Mercedes E-Class S212 equipped with different automotive sensors, for example, camera, lidar and radar sensors. Figure 4.1 shows the front view of this vehicle with all installed radar sensors, as well as a close-up view of a single radar sensor. In this thesis, three radar sensors are deployed, which are mounted on the front left, center and right of the experimental vehicle. The utilized automotive radar sensor is an *ARS 408-21 Long Range Radar Sensor 77GHz Premium* [Con] of the *Continental Engineering Services GmbH* company. The ARS 408-21 is a very robust sensor that operates in the frequency band of $76 - 77$GHz to measure distance and radial velocity of objects based on Frequency Modulated Continuous Wave (FMCW) with very fast ramps. Moreover, it uses Digital Beam Forming with a total of 24 antenna channels, where in the near field as well as in the far field it applies 2 antennas for sending and 6 antennas for receiving the signals. The radar sensor has two modes to measure reflections, one for the near range and the other for the far range. Table 4.1 gives an overview of all important sensor properties. The ARS 408-21 radar sensor provides for both modes measurements in a time period of approximately 72ms. Since the sensor is connected via Controller Area Network (CAN) interface, it is limited in the amount of data that can be sent using CAN messages in this time interval. More information about the ARS 408-21 radar sensor is provided in [Con]. In summary, the ARS 408-21 is a high-resolution automotive radar sensor that provides measurements at short time intervals and generates multiple radar targets for a single real object.

Property	Mode	Performance
distance range	near range	up to 20m at ±60°
at opening angle		up to 70m at ±45°
	far range	up to 150m at ±9°
		up to 250m at ±4°
distance resolution	near range	0.39m ± 0.10m
with accuracy	far range	1.79m ± 0.40m
azimuth angle resolution	near range	3.2° ± 0.3° at 0°
with accuracy		4.5° ± 1.0° at ±45°
at opening angle		12.3° ± 5.0° at ±60°
	far range	1.6° ± 0.1°
velocity range	both ranges	−400km/h to +200km/h
velocity resolution	near range	0.43km/h ± 0.1km/h
with accuracy	far range	0.37km/h ± 0.1km/h

Table 4.1: Radar sensor data: Overview of properties and data about the ARS 408-21 high-resolution automotive long range radar sensor.

After discussing the hardware specifications, the setting options of the radar sensor with respect to its software are explained. The ARS 408-21 sensor offers numerous options and parameters for configuring its behavior, for example, the transmitted radar power, the sensitivity or the usage of filtering methods. Furthermore, the sensor supports two different types of output, the *cluster list* and the *object list*. Firstly, the radar sensor generates clusters based on the signal reflection. A cluster consists of spatial position, radial velocity and RCS value. With reference to the signal processing chain presented in Figure 2.1, the cluster list corresponds to radar targets. Depending on the output configuration, the sensor sends a maximum number of 250 clusters or 125 clusters with additional quality information for each cluster. This extra message contains details such as standard deviation, false alarm probability, the cluster state and the ambiguity state of the radial velocity measurement. Secondly, the sensor produces a list of objects including their history. This means that multiple radar reflections are clustered to the same object and then are tracked over time. Under consideration of the signal processing chain, the sensor clusters multiple radar targets to object hypotheses and performs an object tracking. This thesis proposes an algorithm for object detection, that is the generation of object hypotheses, which are forwarded into a multi-object tracking system. Consequently, the radar sensor is configured to output the cluster list including radar targets. Since the radar object detection system performs much better when the input radar target list is dense, the sensor is configured to generate up to 250 radar targets in each measurement cycle.

An ego motion compensation is required for further processing of measured radar targets. In order to determine the ego motion, the experimental vehicle is equipped with an Automotive Dynamic Motion Analyzer (ADMA) of the company *GeneSys Elektronik GmbH*. This system combines an Inertial Measurement Unit (IMU) and a Differential Global Positioning System (DGPS) that provides measurements for acceleration and yaw rate at a rate of 50Hz. The ADMA determines the vehicle's ego motion at any time. In order to compensate the resulting ego motion part from the measured radar targets, a preprocessing software module is implemented. This ego motion compensation is done immediately after decoding the CAN messages.

4.2 Datasets

In order to use the radar object detection system proposed in Section 3.1, suitable radar data is required. The training, application and evaluation of the radar object detector is based on different datasets. Firstly, this section discusses some radar datasets in literature and presents two publicly available datasets in detail. Since none of these datasets are fully qualified, a proprietary radar dataset is developed.

The literature study in Section 3.1.2 shows that some approaches for object recognition based on radar data already exist. In order to evaluate the proposed methods, radar datasets are necessary. But unfortunately, most of them are not publicly available what makes it impossible to reconstruct the results. In [SHDW18a], semantic segmentation is performed on radar targets. The used dataset [SHS+21a; SHS+21b] contains real-world radar data collected from multiple 77GHz sensors. Another radar dataset in literature is the one utilized [PDKG20] for road user detection. The radar data is recorded in an urban environment using a radar sensor, which is of the same series as the one described in Section 4.1 above. The annotation of the collected radar data is automatically performed on the basis of a camera detection system. Presumably, the effect is that not all objects are accurately or correctly labeled. Moreover, the resulting dataset is not publicly available at the time of this thesis.

This thesis also utilizes a radar dataset that is not yet publicly accessible, but the proposed technique for object detection in radar data is also evaluated on two public datasets. Thus, it is possible for anyone to reproduce all the results presented in this thesis. Henceforth, the focus is on the datasets that are used for evaluating the proposed radar object detector, which in total are three. Section 4.2.1 presents the *Astyx HiRes2019 dataset* and Section 4.2.2 introduces the *nuScenes dataset*, which are both publicly available. Section 4.2.3 discusses the internal *Radar Dataset*, which is created as it turned out that the other two are not ideal for this thesis to develop a radar object detection system that can be deployed in a real-world application.

Dataset Preparation

In order to forward the radar data into the object detection system proposed in Section 3.1, data must be available in a specific format. Firstly, the input radar data has to be a set of targets comprising the spatial position in form of (x, y)-coordinates, the ego motion compensated radial velocity $\tilde{v}_r$ and the RCS value σ of a radar target. In case that the radar sensor described in Section 4.1 or a different sensor which also provides that data, is used, the corresponding values are simply transferred. Otherwise, a method for mapping the sensor data to the appropriate data format is necessary. Secondly, for a faster training process of the object detection model, it is advisable to divide the complete dataset into patches in advance. The reason is that the training process of respective PointNets for classification, segmentation and bounding box estimation is solely based on preprocessed patches. Accordingly, the patch proposal and object extraction module of the detection system are rather detached modules regarding the training, and do not have to be considered in this stage. Therefore, the dataset preparation performs the patch proposal on the entire dataset and then stores all patches for the subsequent training process. Consequently, the prepared dataset does not consist of radar targets from a complete measurement cycle, but rather it contains separate patches including a smaller radar target list.

The patch proposal utilizes various parameters to generate patches from the original radar target list. An important parameter is the patch size, because it may have a significant impact on the detection result of a considered patch. Another possibility to control the patch generation is the decision whether a patch is valid or not. One option is the number of radar targets that determines if a patch is applicable. This can be either the total amount of targets inside a patch or the number of reflections belonging to the object of interest itself. There are other approaches on how to handle the patch generation, but this thesis restricts them to the patch size and number of radar targets. During this process, a patch is accepted as valid, when a minimum number of targets belonging to the classified object are present in that region. The choice of patch proposal parameters for the dataset generation process is heuristic. In order to detect objects in a patch, it is a basic requirement that a patch completely encloses the object of interest. As a consequence, the patch size depends strongly on the objects existing in the dataset. However, for a comparison of the detection results on different datasets, the patch size is the same for all datasets used in this thesis. Therefore, Section 5.1 conducts a brief investigation based on a subset of the *Radar Dataset* introduced in Section 4.2.3 below. The experiment reveals that a patch size of 22 shows the best performance considering various metrics. Furthermore, the minimum number of radar targets which belong to the classified instance of a patch is 2. This means, for an object patch at least two radar target must be assigned to the object of interest to make the patch valid. In case of clutter patches, there must be one other clutter reflection present in addition to the center target. There is

another restriction for patches belonging to an objects. It may happen that an object of interest is not completely present in a patch, because it is too large or is located at the edge of the sensor's FOV. Since such objects do not necessarily contribute positively to the training process, all patches where less than 25% of the object's bounding box is contained within the patch are sorted out. In total, the patch proposal generates an individual patch for every radar target in the measured target list. Since most of them are reflections not belonging to any object, clutter patches prevail in the resulting dataset. All in all, there are sufficient clutter reflections per measurement cycle, and due to fact that many of them appear similar, some of them may be neglected. Consequently, this requires a technique for data balancing solely at the expense of clutter reflections, or more precisely, patches classified as clutter. The patch proposal processes all radar targets one by one, starting from the close ones to the more distant ones. A simple method is to count the number of generated patches per class and discard a clutter patch if it exceeds the number of object patches. The problem is that since the radar target list is systematically looped, clutter patches are methodically dropped. That is why a random factor is introduced, that controls the discarding process of clutter patches. When generating a patch of the clutter category, the proposed technique checks two conditions: Firstly, the patch remains if the total number of object patches is greater than the number of clutter patches. Secondly, a pseudo-random generator draws a number in the semi-open range $[0.0, 1.0)$. In case that the random number is greater than a value of 0.1, the clutter patch is discarded. This method ensures a balanced dataset with regard to the ratio between object and clutter patches. The random factor does not guarantee a 50/50 split, but still one that is close to it. Further, this technique does not control the ratio of respective object classes among each other, but it assures that the prepared data includes all labeled objects which are suitable for a training.

In order to identify model parameters for a neuronal network, different datasets are essential. Therefore, the complete dataset is divided into three parts, the training and validation dataset during the training process, and the testing dataset for evaluation purposes. The training process described in Section 3.1.4 optimizes the initial parameter set based on backpropagation over several epochs and evaluates them on the validation dataset. The training is stopped when the trained model performs well and the results on that dataset do not improve any more. Furthermore, during training of the neuronal network, it is a common practice to vary the model's hyperparameters. For fixing these parameters, the validation process compares different results based on the validation dataset and selects the best parameter set. The last step is to show the generalization ability of the finalized model. This is done by applying the model to a dataset containing unseen samples that are independent of the training and validation data, the testing dataset. Since data for training and validation is only available in patches so far, the evaluation can only measure the performance of the model on individual patches, but not for the entire radar object detection system. This requires another testing dataset that contains labels of

complete radar frames, instead of selected patches from different measurement cycles. For this purpose, the patch proposal prepares the radar data in such a way that extracted patches still can be associated with the individual measurement cycle. This allows to consider the radar data frame by frame, so that it is possible to generate a result with respect to a radar frame. After the object extraction using all patches of a radar frame, the detection result for the whole measurement cycle is obtained. Since the training dataset only contains samples with at least two radar targets belonging to the object of interest or clutter class, this restriction still holds when generating the testing dataset with all radar frames. In contrast, when generating the test data comprising full radar frames, the constraint that 75% of the object's bounding box must be included in the patch, is discarded. This means that all objects which generate at least two radar reflections are present in the final testing dataset. In summary, the described data preparation results in four datasets, two for the training process and two for evaluation purposes. The training and validation datasets solely contain extracted patches and are balanced in regard to object and clutter patches. Both testing datasets are based on the same data, but have different application fields. The patchwise dataset is similar to the training and the validation data, thus, it comprises selected patches to ensure data balancing. The objective of that dataset is to evaluate the classification, segmentation and bounding box estimation module of the radar object detector. Whereas the evaluation goal on the test dataset with full radar frames is to demonstrate the capability of object detection in a radar target list. Accordingly, this dataset is used to supply testing results of the complete object detection system for each radar measurement cycle.

4.2.1 Astyx HiRes2019 Dataset

The *Astyx HiRes2019 dataset* [MK19a] is a dataset with focus on high-resolution radar data. For this purpose, a proprietary radar sensor, the *Astyx 6455 HiRes* radar, is deployed. This sensor produces data on a target level, and is even capable of generating spatial information in three dimension. Hence, the resulting output is a radar target list with (x, y, z)-coordinates, radial velocity and the magnitude of the reflection that is convertible to an RCS value. In order to generate ground truth data, a semi-automatic labeling process based on measurements of a radar, lidar and camera sensor, is performed [MK19a]. This results in 3D object labels containing position, rotation, size in all dimensions, and class information with the distinction of seven classes, occlusion indicator and uncertainty information for position and extent of an object. Although the semi-automatic labeling workflow enables a rather fast labeling, the entire dataset only comprises slightly over 500 measurements frames. That is a small amount of data and is practically not suitable for the application of deep learning methods. Moreover, this radar dataset has a tremendous disadvantage: the measured radial velocities are not ego motion compensated and to complicate

matters no ego motion is provided. For the usability of radar data that is actually an exclusion criterion. Nevertheless, as there are almost no public radar datasets available, this thesis examines that dataset anyway. This includes a training and evaluation of a model for object detection. However, the results are not considered to be representative compared to the performance based on the internal radar data.

The high-resolution radar sensor that is employed to record the raw data for the Astyx dataset is a completely different one compared to the sensor introduced in this thesis. Hence, the measured radar target list does not match the expected data format which is defined for the object detection system. As a consequence, the target list of radar frames from the Astyx dataset needs to be converted accordingly. The problem formulation in Section 3.1.1 requires that the input is a set of four-dimensional radar targets comprising 2D spatial coordinates, radial velocity and RCS value. For this reason, the spatial information is projected onto the flat world by cutting the z-component of the position. Despite the measured amplitudes of radar reflections being different compared to the RCS values, the amplitude represents a similar property and may remain unchanged. As already stated, the velocity data is a problem as no ego motion compensation is performed. Due to the fact that no ego motion information exists, the compensation cannot be done retroactively. Consequently, the radial velocity values remain unchanged as well. This is a significant issue, because without ego motion compensation of the measured radial velocities, the object detection depends on the ego motion. However, knowing that this may have a drastic effect on the performance of the radar object detector, the original radial velocities are retained and an evaluation is performed nonetheless.

The Astyx dataset comprises labeled data of relatively few radar frames, but it is advantageous that the Astyx 6455 HiRes sensor is able to measure a lot of radar targets in a single measurement cycle. Table 4.2 shows the full statistics of this dataset including the number of labeled instances and the patch distribution over all object classes. Although the total number of radar frames is small, the patch proposal generates a large amount of patches, because a measurement cycle contains many individual radar targets. The average number of radar targets as well as the minimum and maximum targets per frame confirm this fact. Furthermore, the average number of reflections in a patch, and also the radar targets reflected from an object, is quite high. Basically, these are very favorable circumstances for object detection, but even the large number of radar targets in a measurement does not compensate the lack of ego motion information. The evaluation of the radar object detector on the Astyx dataset in Section 5.1.3 reaffirms this anticipated statement.

In order to get an impression of the data, some selected examples of radar frames are discussed hereinafter. Figure 4.2 visualizes a few parts of radar target lists from six different measurement cycles that are recorded in an urban environment. For the purpose of demonstrating the importance of missing ego motion information, the

Criterion	Class	Training	Validation	Testing (Frames)
dataset distribution	frames	70%	15%	15%
		386	80	80
	patches	182659	33521	33625 (144530)
labeled instances	object	2100	486	493
	car	1967	454	451
	truck	106	32	22
	bike	1	0	12
	pedestrian	26	0	8
patch distribution	object	57.8%	62.8%	63.1% (14.7%)
		105564	21056	21231
	clutter	42.2%	37.2%	36.9% (85.3%)
		77095	12465	12394 (123299)
	car	46.8%	53.7%	52.8% (12.3%)
		85498	17996	17747
	truck	10.7%	9.1%	9.5% (2.2%)
		19605	3060	3209
	bike	0.0%	0.0%	0.6% (0.1%)
		15	0	207
	pedestrian	0.2%	0.0%	0.2% (0.0%)
		446	0	68
average targets	frame	2316	1842	1807
	patch	514	297	398 (399)
	object	146	88	105
	car	78	66	72
	truck	447	215	298
	bike	–	–	19
	pedestrian	27	–	13
minimum and maximum targets	frame	320/9432	557/2594	299/8501
	patch	2/3328	2/1243	2/2172 (2/2176)
	object	2/1272	2/605	2/633
	car	2/310	2/185	2/259
	truck	6/1272	16/605	5/633
	bike	–/–	–/–	4/24
	pedestrian	3/52	–/–	2/23

Table 4.2: The Astyx dataset: All statistics on the radar dataset considering frames and patches, as well as instance information. For training and validation a balanced dataset is available, but for testing there is patch data and a dataset that contains complete frames.

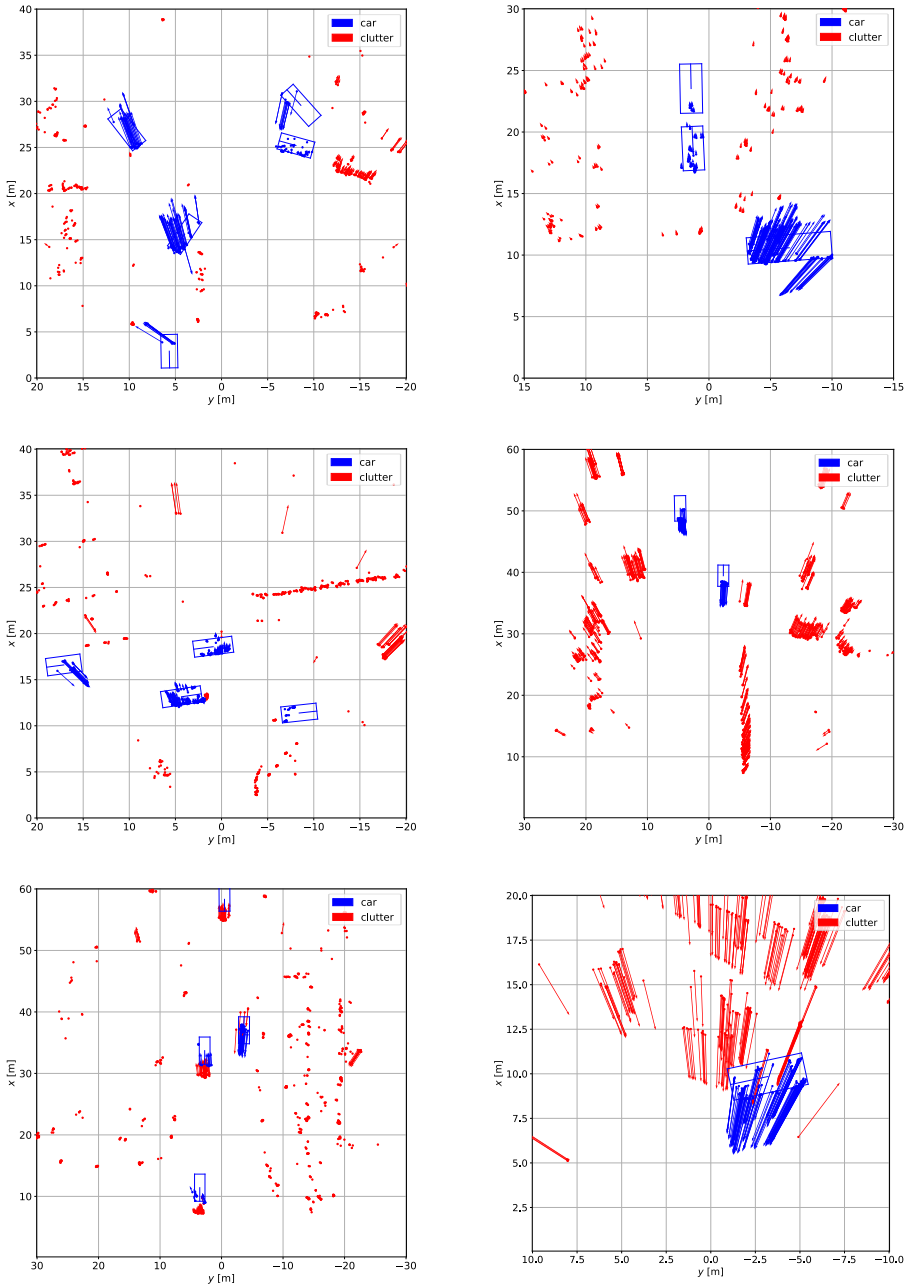


Figure 4.2: Impressions from the Astyx dataset: Exemplary radar target lists taken from some urban scenarios. On the left side, the ego vehicle is standing still, whereas on the right side, it is in motion.

ego vehicle is stationary in three examples and in motion for the other three. Even though the plots only depict excerpts of the target list, the examples illustrate the abundance of radar targets per measurement cycle. A closer look at the shown data reveals that the radial velocities of some radar targets are not meaningful. In general, the direction of radial velocities is such that the value is positive for objects moving away from the sensor and negative for approaching objects. Since the Astyx dataset does not perform an ego motion compensation, this is only recognizable for dynamic objects when the ego vehicle is stationary. In Figure 4.2, the first and third examples on the left side reveal that this property is violated. But even if the ego vehicle is moving, the corrupted radial velocity data is still visible when looking at stationary radar targets. Considering the fact that when the ego vehicle moves forward, the radial velocity values of stationary radar reflections must all point towards the sensor. However, the second sample on the right side in Figure 4.2 shows that almost all radar targets point away from the sensor, even though the vehicle actually moves in a forward direction. A closer examination of the radial velocity values discloses that over the entire Astyx dataset, the minimum and maximum radial velocity values are $v_{r,\min} = -5.12\text{m/s}$ and $v_{r,\max} = 5.20\text{m/s}$, respectively. This is equivalent to a radial velocity that does not even reach 20km/h, which absolutely does not make sense when reviewing all scenarios in the dataset. An assumption that may explain these values is, that some kind of arithmetic overflow exists in the radial velocity values. This means that as soon as a certain value is exceeded, for instance $v_r = 5.5\text{m/s}$, the next value, for example $v_r = 5.6\text{m/s}$, is mapped into the negative numerical range, thus $v_r = -5.6\text{m/s}$ in this case. This assumption has not been proven within the scope of this thesis, but remains unresolved as a hypothesis. Since this thesis does not focus on solving the problem of corrupted radial velocity values, these deficiencies must be accepted when processing radar data from the Astyx dataset.

In summary, the Astyx dataset only contains a limited number of radar frames, but each measurement cycle comprises a large amount of radar targets. Unfortunately, the radial velocity in this dataset is not ego motion compensated, and in addition the radial velocities do not behave as typically expected from radar reflections. In principle, it is possible to forward the radar data into the proposed object detector, but no substantial results are expected and moreover, these must be treated with caution. However, this thesis evaluates the radar object detection system on the Astyx data, just for the sake of completeness. In other words, the evaluation in Section 5.1.3 presents all obtained results, but it is not interpreted in any detail.

4.2.2 NuScenes Dataset

The *nuScenes dataset* [CBL+19] is a public dataset focusing on data for autonomous driving. The dataset includes thousands of scenarios with dense traffic recorded

in the cities of Boston and Singapore. Each of the manually selected scenes has a length of 20s and comprises various driving maneuvers of different locations, weather conditions and vehicle types. Apart from a huge amount of camera and lidar data, the nuScenes dataset also contains 40000 labeled frames with 1400000 radar sweeps and 1400000 object bounding boxes. For data collection, the experimental vehicle is equipped with six camera sensors, one spinning lidar sensor on top of the vehicle's roof, five radar sensors and a IMU to measure the ego motion information. For this thesis the radar data generated by the radar sensor is of particular interest, where three sensors are installed at the vehicle's front and two at the rear. In addition to the fact that the nuScenes dataset actually contains radar data, the sensor deployed is indeed the same model as used in this thesis, namely, the Continental ARS 408-21 long range radar sensor. However, the radar data of the nuScenes dataset differs slightly from the data in this thesis, which has a significant impact on the resulting target lists. Section 4.1 already discusses that it is possible to configure the ARS 408-21 radar sensor in different ways, either the sensor outputs 250 radar targets or only half of them, but with additional quality information. The radar sensors of the experimental vehicles operating in nuScenes for data collection are configured to generate only 125 radar reflections with extra information on each measured target. Besides the spatial position, radial velocity and RCS value per radar target, this quality information provides the standard deviation of all measured variables. Further, it includes details about the validity state of a radar target, the ambiguity state of the radial velocity and a false alarm probability for each measured target. All this additional information allows filtering the radar targets according to certain criteria, such that the resulting target list includes higher quality measurements. Basically, the radar data comprises exactly the same information as intended for the object detection system, but the target lists are significantly sparser due to half the number of measured radar reflections, which makes object detection more difficult.

The open source *nuScenes devkit* [Mot] implements functions to extract all data from the nuScenes dataset in a convenient way, but also to filter the target lists with respect to certain criteria. In case of radar data, the filtering is based on the quality information generated by the sensor. For comparison to the radar dataset which is introduced in Section 4.2.3, the radar data preparation module deactivates all these filter options. This means, even if the sensor rates a measurement as invalid, the radar target is still added to the output target list. Nevertheless, this results in a radar target list with a maximum of 125 reflections per measurement cycle or in the jargon of the nuScenes dataset, in one radar sweep. Table 4.3 displays the full statistics on the nuScenes radar dataset after the described data preparation process. One outstanding advantage is that an extremely large number of objects are labeled in the dataset. However, a disadvantage that complicates object detection is the sparse radar target list in a frame or in the extracted patch. The presumption is that the capability of object detection solely on radar data is difficult, but not impossible. Despite this fact, the nuScenes dataset is excellent for a generally

Criterion	Class	Training	Validation	Testing (Frames)
dataset distribution	frames	65%	17%	18%
		42703	11388	11723
	patches	509775	130956	134489 (1174700)
labeled instances	object	75509	18777	20464
	car	49611	12690	14463
	truck	22610	5469	5288
	bike	663	149	183
	pedestrian	2625	469	530
patch distribution	object	52.9%	51.9%	51.7% (5.9)%
		269826	67975	69558 (69893)
	clutter	47.1%	48.1%	48.3% (94.1)%
		239949	62981	64931 (1104807)
	car	28.6%	28.2%	30.6% (3.5)%
		145565	36876	41173 (41054)
	truck	23.0%	22.7%	20.0% (2.3)%
		117070	29760	26820 (27274)
	bike	0.3%	0.2%	0.3% (0.0)%
		1492	327	421
	pedestrian	1.1%	0.8%	0.9% (0.0)%
		5699	1012	1144
average targets	frame	109	111	113 (107)
	patch	15	16	15 (14)
	object	5	5	5
	car	3	3	3
	truck	7	8	7
	bike	2	2	3
	pedestrian	2	2	2
minimum and maximum targets	frame	2/125	3/125	4/125 (3/125)
	patch	2/75	2/69	2/60 (2/61)
	object	2/29	2/30	2/40
	car	2/25	2/22	2/18
	truck	2/29	2/30	2/40
	bike	2/6	2/4	2/7
	pedestrian	2/8	2/8	2/6

Table 4.3: The nuScenes dataset: All statistics on the radar dataset considering frames and patches, plus instance information. For training and validation the balanced patch data is sufficient, but during testing data with single patches and also complete frames exists.

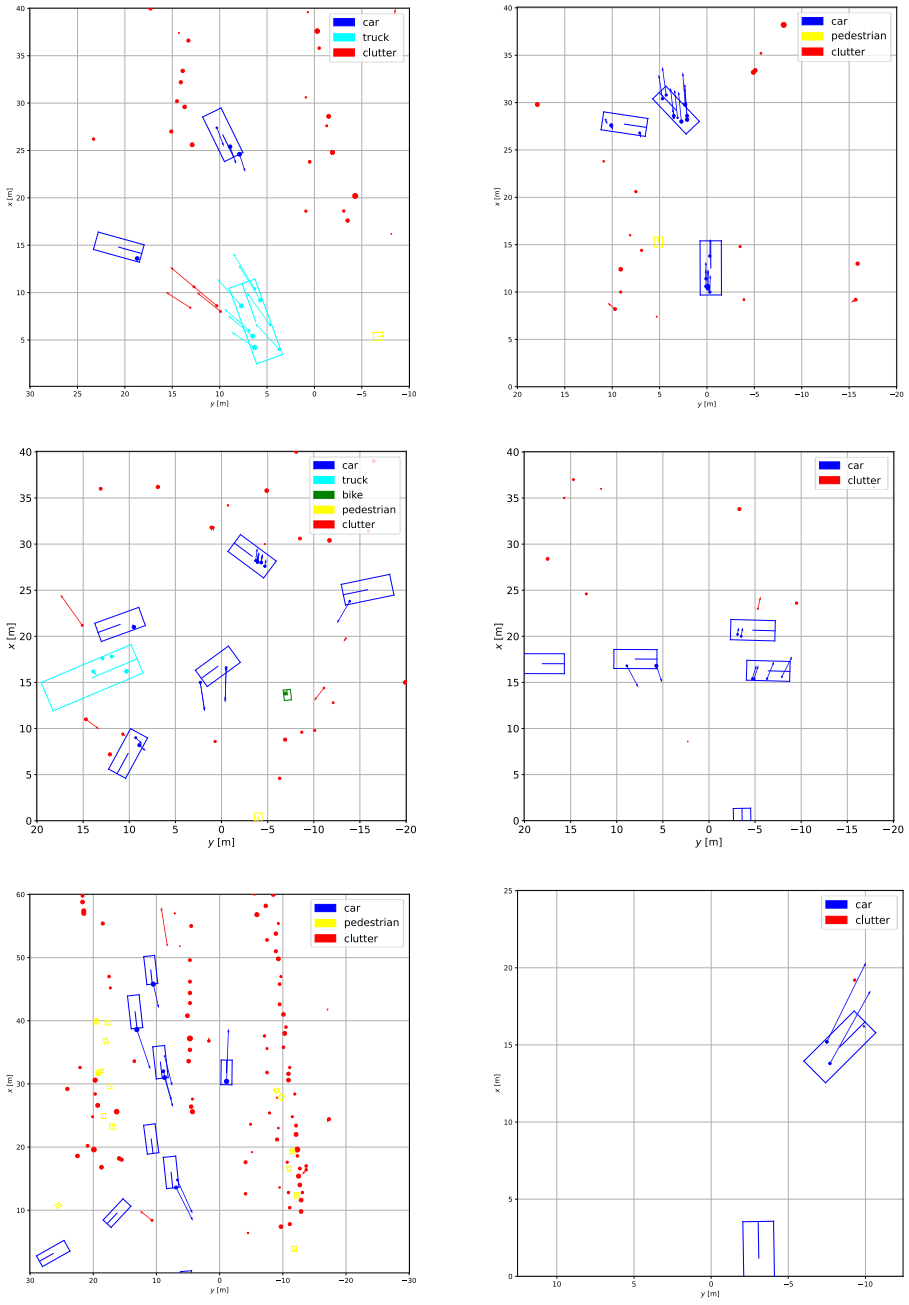


Figure 4.3: Impressions from the nuScenes dataset: Examples of different radar target lists that are measured in an urban environment. In most of the samples, the objects only generate very few radar reflections, even though the objects are quite close to the sensor.

applicable evaluation, because this data is publicly available, and thus, the results are reproducible. Section 5.1.3 demonstrates the performance of the radar object detection system based on nuScenes data, where the results on individual patches are considered as well as the overall object detection ability on complete radar frames.

The nuScenes dataset comprises labeled radar targets from lots of different traffic scenarios in an urban environment. Figure 4.3 shows a small selection of these scenes with fewer or more traffic participants in the situation, and some of them in rainy weather conditions. The examples on the left side demonstrate the quantity of labeled objects, as well as the occurrence of different classes. However, all of the selected samples illustrate the sparse number of radar reflections belonging to an object. The example at the bottom left, in which a high number of objects are present in the scene, emphasizes this fact. Most of the vehicles still generate radar measurements but not enough for a reliable bounding box estimation. Further, it can be observed that there are several pedestrians in the scenes, however not even one radar target is assigned to them, which makes object detection impossible. Another surprising aspect of this sample is that objects hardly contain any radar targets, but a relatively large number of clutter reflections are present in that target list. This is especially astonishing, because a radar is supposed to recognize moving targets better due to the additional radial velocity information. One more conspicuous observation is that the measuring sensor provides only few radar targets even for objects that are very close to the ego vehicle. Especially, the example at the bottom right side reveals this, where despite a good location of the object to the radar sensor, it still produces merely two reflections. Needless to say that this is not the situation in all scenarios, but rather the nuScenes dataset also consist of a large number of frames with sufficient measured radar targets belonging to the object of interest. In the sample on the top right, two of the existing objects generate as many radar reflections, which enables the detection system to estimate a classified bounding box.

In conclusion, the nuScenes dataset is well suited for evaluating the performance of the radar object detection system in general. Due to the rather sparse radar target lists with only a maximum of 125 targets per measurement cycle, the conditions for the object detector are not ideal. Therefore, it must be assumed that the detection results ultimately are not going to be the best, which mainly affects the metrics as these evaluate the result over the whole dataset. Nevertheless, the nuScenes dataset comprises a lot of radar frames with target lists which are successfully processed and the proposed object detection system outputs reliable results. Section 5.1 presents the complete evaluation on this dataset using several metrics, including a typical one, to score the overall object detection capability. The crucial point that makes it essential to perform an investigation on the nuScenes dataset is that it is publicly available, and hence, the results are reproducible without exception. This evaluation is indispensable to allow other approaches for object detection in radar data the opportunity for comparison under the same conditions against the proposed method.

4.2.3 The Radar Dataset

The *Radar Dataset* is an internal dataset with labeled radar data which is recorded with the experimental vehicle of Ulm University. For the labeling process of radar targets, an open source labeling tool [PRM18; Tec] is modified for radar data. This tool allows a semi-automated labeling, as it internally includes a simple tracking system based on a Kalman filter [Kal60]. After the labeling, all objects in the scene and the associated radar targets reflected from these objects are annotated. The Radar Dataset comprises a lot of radar target lists recorded with the experimental vehicle in rural and provincial surroundings around Ulm University. In order to have the widest possible selection of traffic scenarios available, a number of recordings are made in different traffic situations and under various weather conditions. Since the labeling effort is not to be underestimated, the Radar Dataset limits the selection of recordings to merely the eleven best ones. But even these are not fully annotated, because the goal of this dataset is to cover as many different scenarios as possible with minimal labeling effort. As in many recorded sequences, quite similar situations are repeatedly present, just parts of the recordings are conscientious picked out and actually labeled. The final dataset is indeed not very large, but it contains enough radar data from many typical common scenarios which frequently occur in everyday traffic. In the scope of this thesis, the Radar Dataset has to satisfy two main purposes: Firstly, the principal objective is to show that the radar object detection system works and is suitable for the perception of the vehicle environment in a real system. For this purpose, it requires a diversity of traffic situations with various traffic participants and under different conditions. Secondly, the radar detection system is intended to be deployed in real applications, specifically on the experimental vehicle of Ulm University as test platform. Although the radar data all originate from the area around Ulm University, the final Radar Dataset is far sufficient to demonstrate the generalization capability of the radar object detection system proposed in this thesis, as well as being applicable in real-world applications.

The Radar Dataset comprises several carefully selected sequences with consecutive radar frames. Depending on how many distinct traffic situations occur in a sequence, these are of longer or shorter duration. An important fact is that radar sensors generate noise measurements, such that successive scenarios which look similar to the human eye produce quite different radar frames. As a consequence, even in two consecutive measurement cycles of the same scene the radar target lists may noticeably be different. After selecting appropriate radar frames, the labeling process is performed with human assistance. Afterwards the dataset preparation presented above automatically converts the annotated data to the predefined data format to enable the tooling for the radar object detection system. Furthermore, this resulting dataset is divided into three parts to distinguish between data for training, validation and testing purposes. In this part, meticulous care and attention

Criterion	Class	Training	Validation	Testing (Frames)
dataset distribution	frames	64%	16%	20%
		5210	1291	1642
	patches	117694	29206	37594 (279655)
labeled instances	object	16354	4041	4546
	car	13704	3407	3493
	truck	1071	262	787
	bike	598	153	194
	pedestrian	981	219	72
patch distribution	object	54.1%	54.2%	55.4% (7.6%)
		63667	15825	20827 (21296)
	clutter	45.9%	45.8%	44.6% (92.4%)
		54027	13381	16767 (258359)
	car	44.2%	44.4%	34.3% (4.6%)
		52073	12972	12898
	truck	6.1%	6.3%	19.3% (2.8%)
		7198	1830	7250 (7719)
	bike	1.4%	1.5%	1.3% (0.2%)
		1675	426	486
	pedestrian	2.3%	2.0%	0.5% (0.1%)
		2721	597	193
average targets	frame	189	188	190 (176)
	patch	25	26	32 (34)
	object	6	6	8
	car	5	5	5
	truck	13	13	13
	bike	3	3	3
	pedestrian	3	3	3
minimum and maximum targets	frame	25/250	25/250	29/250 (24/250)
	patch	2/120	2/101	2/112 (2/113)
	object	2/40	2/32	2/42
	car	2/25	2/27	2/27
	truck	2/40	2/32	2/42
	bike	2/10	2/12	2/8
	pedestrian	2/9	2/8	2/6

Table 4.4: The Radar dataset: All statistics on this radar dataset considering entire frames and patches, together with instance information. The data with balanced patches is used for training and validation, for testing purpose, both patch and frame datasets are available.

is necessary to ensure that training and test data are completely independent of each other. Therefore, the labeled radar data of all sequences is completely assigned to either the training data or the test dataset. In case that the sequences are from the same recording, the division process assures that the scenes are separated from each other according to the location of the event. Table 4.4 shows the distribution and some more statistics of the Radar Dataset. Compared to the nuScenes dataset, the average targets distribution of the Radar Dataset is much higher, as already expected, because there are more radar targets present per frame or patch. As a result, slightly more radar reflections are generated per object, where especially objects of the truck class even show significantly more reflections. Like the Astyx and nuScenes datasets, this dataset is imbalanced in terms of objects per class. However, this thesis presents a radar dataset that convinces predominantly with the high quality of real measured radar target lists in comparison to the other two datasets.

Altogether, the Radar Dataset contains a relatively large number of cars, which is comprehensible, because in rural and provincial areas this type of object appears more often. However, the creation of this radar dataset focuses on the fact that per scene, or more precisely per radar frame, several objects are always present at the same time. A closer examination on the nuScenes dataset reveals that many radar samples comprise only a few objects. In scenarios with more objects located in the FOV, the radar sensor measures a surprisingly small number of reflections per object. Figure 4.3 confirms this negative fact with certain matching examples from the prepared nuScenes dataset. In contrast, the sensor configuration applied in thesis ensures that the target lists are denser and even in situations with congested traffic, the ARS 408-21 radar sensor measures several targets per object. The impressions in Figure 4.4 and 4.5 highlight that the sensor is able to capture multiple reflections from the same object, even with numerous objects around. Both figures impressively illustrate the potential of correctly configured high-resolution radar sensors. Furthermore, this radar sensor is to generate enough measured reflections that even the extent of objects is recognizable, and to reliably separate measurements of objects that are close to each other. The first and second radar frames on the left side of Figure 4.4 demonstrate this capability quite well. In theory, the ARS 408-21 radar sensor may recognize objects up to a maximum range of 250m, where at large distances the sensor measures a single reflection per object. Certainly, at a range up to 100m, this sensor is still able to generate several radar targets for an object. The second and third radar frame on the left side of Figure 4.5 visualize this situation, where especially large objects have more radar reflections. Moreover, the sensor measures sufficient radar targets that allow object detection when even multiple objects are moving in succession. The three examples that are visualized on the right side of Figure 4.5 support this statement for different traffic scenarios. Thus, the Radar Dataset contains many more such positives samples demonstrating that the applied radar sensor is well suited for the object detection task in radar data.

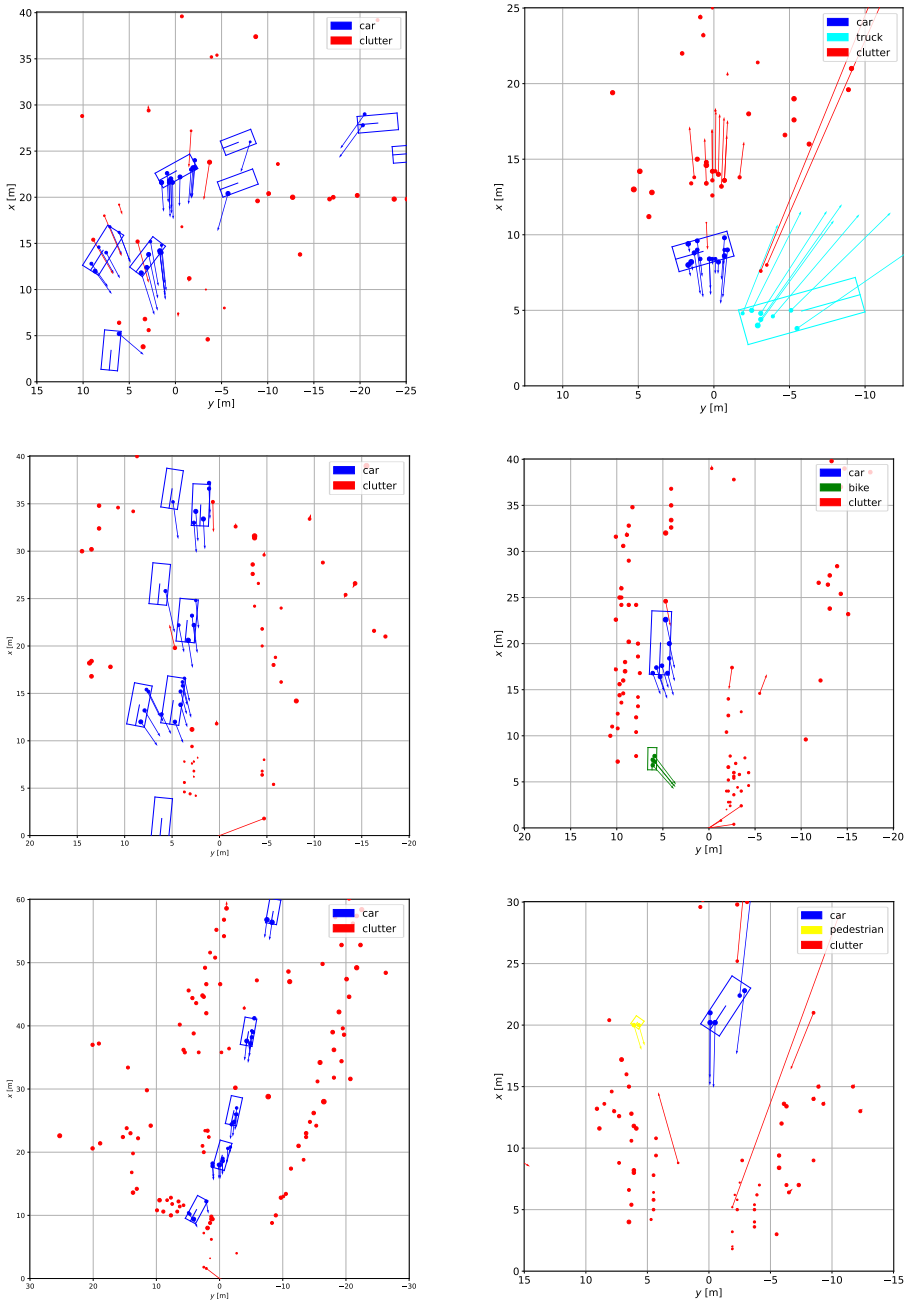


Figure 4.4: Impressions from the Radar Dataset: Several examples of radar target lists containing multiple traffic participants in rural and provincial environment. Almost any object generates sufficient reflections, thus the object detector is able to detect all of them.

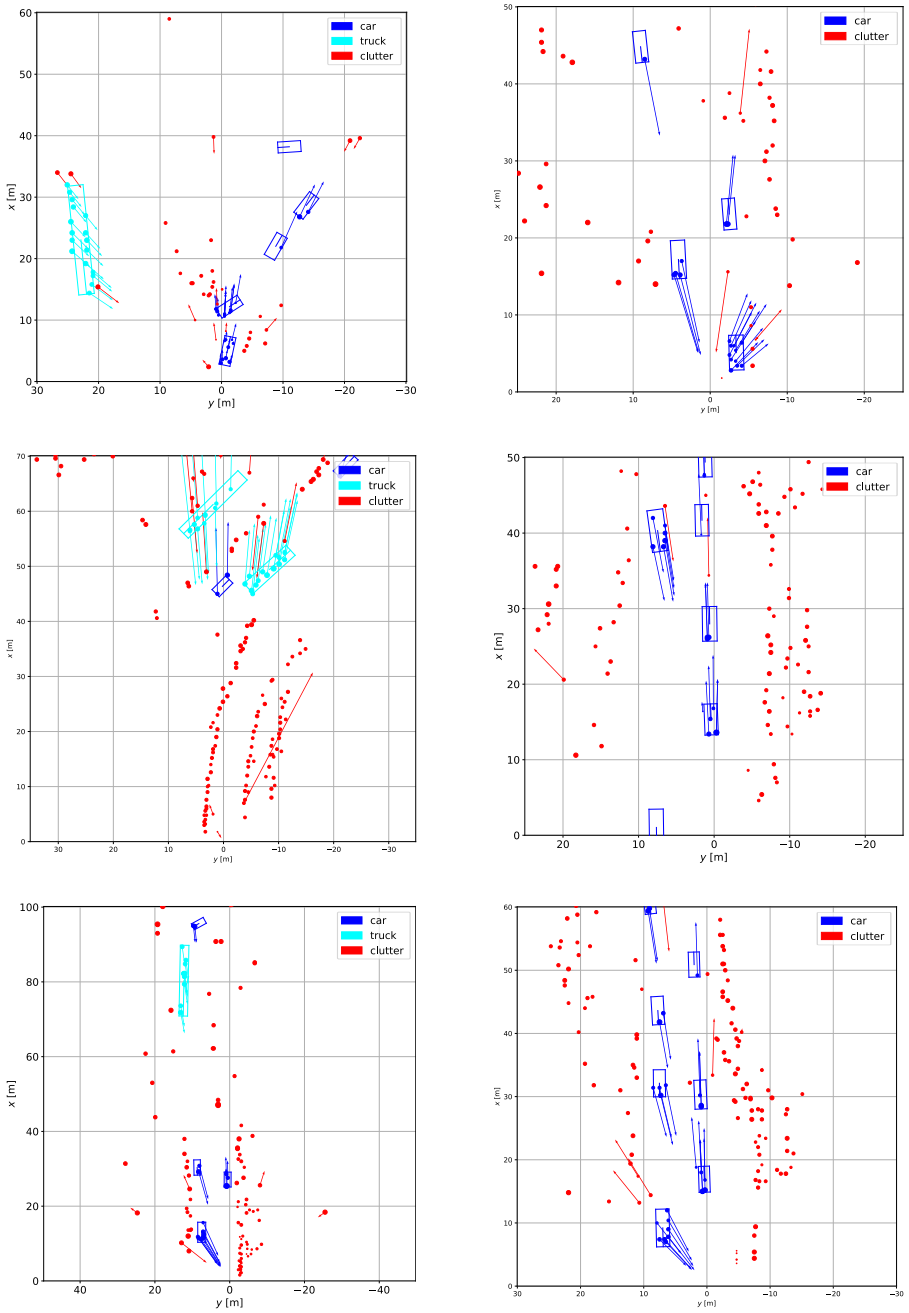


Figure 4.5: More impressions from the Radar Dataset: The high-resolution radar sensor is capable of measuring several radar targets reflected from objects in the near range, as well as from those that are quite far away. Moreover, the employed radar sensor can also identify multiple objects which are located behind each other.

The Radar Dataset does not contain a large amount of annotated radar data and labeled objects, but it is absolutely sufficient to demonstrate the functionality of the radar detection system. For further investigation, it is not a difficulty to re-train the neuronal networks of the detection system with any other radar target list, as long as all data is available in the defined data format. If the radar targets even originates from the same sensor, an already pre-trained model may very easily be further trained with additional annotated radar data. Finally to recapitulate, the proposed radar object detection system is not limited to any particular dataset. A major issue is that the applied radar dataset covers as much general traffic scenarios as possible. Furthermore, the generated radar targets depend on the sensor and a specific software employed on it, which is radar manufacturer proprietary and usually secret. Therefore, for the best possible results, it is recommended to ensure that all radar targets in a dataset originate from the same radar sensor with identical configuration. Moreover, it is suggested to deploy the freezed model, thus, the model with finalized parameters, on a system that is equipped with same sensors or at least sensors that produce similar radar data. But, there is no appropriate and public radar dataset that meets these requirements and provides radar targets similar to the deployed radar sensor from Section 4.1 above. For this reason, the proprietary Radar Dataset is created, where the focus is locally restricted to the area around Ulm University. However, this dataset may be expanded to cover additional areas.

Chapter 5

Evaluation

Finally, this section presents the evaluation of the radar object detection system. Therefore, Section 5.1 provides the results for the radar object detector which is proposed in this thesis. Furthermore, Section 5.2 shows the improvements that are achieved by the multi-object tracking system based on the radar object detections.

5.1 Radar Object Detection

This section evaluates the entire radar object detection system on real radar data and assesses all the results using common object detection metrics. Section 5.1.1 describes the experimental setup for the evaluation. Then, Section 5.1.2 introduces all metrics to rate the final output of the radar object detector, but also to analyze the intermediate results that are further processed internally. Section 5.1.3 presents the object detection results for the previously presented radar datasets. Eventually, Section 5.1.4 concludes the evaluation with a brief discussion and closing conclusion.

5.1.1 Experimental Setup

The investigations of the proposed radar object detection system are exclusively performed with real radar data from the datasets presented in Section 4.2 above. The training and testing is performed on a single GPU of type *NVIDIA GeForce GTX 1070* with Random Access Memory (RAM) of 8GB. The training process introduced in Section 3.1.4 involves the setting of several hyperparameters. Table 5.1 list the parameters which are chosen to be the same for all following examinations. This includes the weighting parameters for the multi-task loss of Equation (3.7) as well as the parameters for learning rate and batch normalization. According to the

Description of the Parameter			**Value**
weighting in multi-task loss	classification loss	w_{cls}	2.0
	segmentation loss	w_{seg}	1.0
	bounding box loss	w_{bbox}	10.0
	corner loss loss	w_{corner}	10.0
learning rate	initial rate in warm-up		0.000001
	warm-up iterations		2000
	initial rate		0.001
	decay rate		0.5
	minimum rate		0.000001
batch normalization	initial decay		0.5
	decay rate		0.5
	maximum decay		0.99

Table 5.1: Settings for the evaluation setup: Overview of all parameters with values which are used in the evaluation of radar object detections.

proposal of the warm-up heuristic in [GDG+17], the learning rate is set extremely low at the beginning for a certain number of training iterations. This method ensures that the parameters of the Adam optimizer stabilize during this time. After the warm-up process, the learning rate is increased to the actual initial learning rate, and then gradually reduced using a decay rate to a certain minimum value as a lower limit. Furthermore, the training performs batch normalization with a specific initial decay and a decay rate to gradually increase the decay up to a maximum value as upper limit. In addition to these design parameters, the batch size and the number of epochs are important. The batch size depends on the selected radar data, more precisely on the amount of radar targets in one measurement cycle. Thus, a radar target list from the Astyx dataset requires much more memory on the GPU than one from the Radar Dataset or nuScenes dataset. Consequently, the actual number of epochs varies according to the specific training run, and is therefore, indicated in the respective evaluations on the datasets. The training is terminated as soon as the intermediate results on the validation data no longer improve. Finally, the evaluation validates the model with frozen parameters on testing data. For testing, the minimum output probability is set to 0.0 for an object detection. This implies that all provided radar object hypotheses are a valid output and no filtering is active.

5.1.2 Metrics

In order to evaluate the radar object detection system, a meaningful metric is necessary. In literature there are already some metrics that are very well suited to measure the performance of such an object detector. However, probably the most popular metric that describes the detection results in terms of a single scalar value is the *average precision* for object detection. The average precision metric requires an introduction of additional metrics to be fully comprehended, namely, *precision*, *recall* and *Intersection over Union*. An overview of various performance metrics for object detection algorithms is provided in [PNS20]. This section introduces all metrics that are applied to analyze the radar object detection system as a whole, but also to rate the results of all individual networks for classification, segmentation and bounding box estimation, which are the essential parts of the complete radar object detector.

Precision, Recall and $\mathbf{F_1}$ Score

The two metrics, precision and recall, are commonly used to score the performance of a respective system for classification problems. For a given classification task, for example, the issue to classify a complete radar target list or each radar target in such a list, the precision metric measures the accuracy of all predictions produced from a classifier. The precision for a particular class is mathematically specified as

$$P = \frac{TP}{TP + FP} = \frac{TP}{\text{total positive predictions}} \in [0, 1], \tag{5.1}$$

where TP describes the number of correct positive predicted samples and FP is the number of the incorrect positive predicted ones. This can be explained by a simple example concerning the classification or segmentation module of the radar object detector. If the radar target list of a patch belongs to an object of the car category, and the classification network predicts the class car, then it is a TP. In case that it belongs to another class, for instance truck or clutter, but the classifier still outputs car as classification, it is a FP when considering the car category. The same is valid for the segmentation network, where it is a binary classification problem. Hence, precision for a given class is the percentage of all correct predicted samples and describes how many of the positive predictions are relevant. A precision with value 1 signifies that all predictions are correct. In contrast, the recall for a specific category measures how well the positive samples are correctly identified, and it is defined as

$$R = \frac{TP}{TP + FN} = \frac{TP}{\text{total positive results}} \in [0, 1], \tag{5.2}$$

where false negatives (FN) are the number of incorrectly negative predicated samples. Just to pick up the above example with the classification unit of the radar object detection system again: if a target list comprises a car, but the classifier predicts another category, it is a FN. For the segmentation network, this is the case, if a radar target belongs to an object, but it is associated to the non-object class. Thus, the recall for a given class describes the percentage of how many of all the actually relevant results are found. A recall with value 1 implies that all relevant results are correct predicted by the utilized classifier. For completeness, the true negatives (TN) are the samples which a classifier correctly predicts negative. This means that if an object does not belong to the car category, the classification module will not output car as the predicted class for the radar target list. Alternatively, the segmentation module is capable of correctly predicting the class for a radar target that does not belong to the object, when predicting the category for each individual radar target.

In order to determine the performance of a classification system, precision and recall must always be considered simultaneously. In fact, reducing the number of FP results in a high precision, but it will automatically decrease the recall. And the other way around, diminishing the number of FN causes the recall to increase, but a decrease of the precision. Consequently, a good precision but a bad recall and vice versa do not constitute a good performance of a classifier. It is important to find a trade-off between the precision and recall performance. The application of a neuronal network even allows this trade-off to be simply adjusted by varying the threshold of a final softmax layer. In object detection tasks, the focus is often on achieving a high precision. This may be justified with the argument that in real applications the effects of a missing detection may have worse consequences. However, it depends on the application of the object detector. In an automotive system, it can be very dangerous if objects are detected incorrectly in front of the vehicle, and as a reaction the automated vehicle brakes. Accordingly, the objective is to tune the networks in such a way to concurrently maximize the precision as well as the recall. In order to capture the results of this maximization at a glance, the *F_1 score* is an appropriate metric. The F_1 score incorporates precision and recall, and combines them through

$$F_1 = 2\,\frac{P \cdot R}{P + R} \in [0, 1], \tag{5.3}$$

to a single scalar value. The F_1 score calculates the harmonic mean of precision and recall. As a consequence, if precision and recall are high, then the F_1 score is high, too. However, as soon as one metric drops, the F_1 score also decreases. A perfect precision and recall, that is both metrics return the value 1, results in a perfect F_1 score which also has the value 1. Whereas in the worst case, the F_1 score is 0, which happens when precision or recall are 0. This thesis lists values for all these metrics to measure the performance of the classification and segmentation networks.

Intersection over Union

In order to analyze the results of the bounding box estimation module, another metric is required. This metric has to compare two bounding boxes by considering all parameters, these are the position, orientation and extent, before eventually providing one single number. The Jaccard index [Jac12] is a measure for the similarity between two sample sets. For analyzing predicted bounding boxes, two boxes are the sample set, which need to be compared based on their areas. In this context, the Jaccard index is also known as the Intersection over Union (IoU) metric, which is defined as

$$IoU = \frac{(\text{Pred BBox}) \cap (\text{GT BBox})}{(\text{Pred BBox}) \cup (\text{GT BBox})} = \frac{\text{area of overlap}}{\text{area of union}} \in [0, 1], \tag{5.4}$$

where *Pred BBox* is the area of the predicted bounding box and *GT BBox* is the area of the ground truth bounding box. If the prediction overlaps exactly with the ground truth, then the IoU equals 1. Whereas, with no overlap of the two bounding boxes, the metric takes its minimum value at 0. In order to score the IoU of all predicted bounding boxes in a given dataset, the evaluation simply averages all IoU of the respective bounding box estimates. The resulting mean IoU (mIoU) is a good measure for the overall bounding box estimation performance of an object detector.

In general, the definition of the IoU is for different geometric areas in two and even in three dimensions. This thesis applies the IoU, only to compare the predicted 2D bounding boxes with the ground truth boxes. However, it is possible to interpret the IoU metric in a point-by-point manner as already applied in [EEG+15], this is called segmentation accuracy. This thesis introduces that metric as the pointwise IoU for scoring the segmentation results. Therefore, the pointwise IoU is specified as

$$IoU = \frac{TP}{TP + FP + FN} \in [0, 1], \tag{5.5}$$

where TP, FP and FN refer to classified targets. Accordingly, that IoU allows an assessment of the classification ability for individual targets in a radar target list. The interpretation of the given equation is that the pointwise IoU represents the overlap of predicted segmentation and ground truth, divided by the union of them. Hence, this ends in the general definition of the IoU as defined in Equation (5.4) with particular attention to the last part of the term. With the specification from Equation (3.3), the segmentation differs between two classes, the object and non-object class. Thus, the evaluation for the segmentation calculates a pointwise IoU for both classes and averages them to get a single value that represents the final segmentation performance. This metric is called the pointwise mIoU and is given by

$$mIoU = \frac{IoU_{obj} + IoU_{\overline{obj}}}{2}, \tag{5.6}$$

where IoU_{obj} is the segmentation result for the object class and $IoU_{\overline{obj}}$ belongs to the prediction for the non-object category. The next sections reveals that the IoU is not only suitable for segmentation or bounding box comparison, but it is an essential part to measure the overall performance of the entire radar object detection system.

Average Precision

The metrics presented so far allow to evaluate the individual modules of the object detection system, these are the classification, segmentation and bounding box estimation components. However, these metrics are not applicable for an analysis of the actual capability to detect objects. Indeed, the average precision (AP) is a popular metric for exactly that purpose. This metric offers a possibility to score object detection results, and thus, it allows to measure the performance of an object detector. In theory, AP computes the mean precision over all recall values in the range from 0 to 1. The two Equations (5.1) and (5.2) show that for the calculation of precision and recall, the number of TP, the total number of positive predictions and the total number of positive results are required. In an object detection task, a prediction is correct, that corresponds to a TP, if the object's class is correctly estimated and the IoU between predicted and ground truth bounding box exceeds a specific threshold. After ranking all the results regarding the predicted confidence score, the calculated precision is plotted against the recall in the precision-recall curve. In general, the AP is the area under the precision-recall curve, which is defined as

$$AP = \int_0^1 p(r)\,\mathrm{d}r \in [0,1], \tag{5.7}$$

where $p(r)$ is a function which returns the precision at specific recall value. Since the computation of an integral is not trivial in practical applications, this thesis applies the interpolated AP [SM86] to evaluate the prediction results of the radar object detection system. The interpolated AP calculates the area under the precision-recall curve by summing up the mean precision for a set of eleven equally distributed recall levels. As a consequence, the interpolated AP is computed by using the definition

$$AP = \frac{1}{11} \sum_{r \in [0, 0.1, \ldots, 1]} p_{intrpl}(r), \tag{5.8}$$

where the function $p_{intrpl}(r)$ returns the interpolated precision value for a specific recall level. The interpolation function for a precision at a given recall is realized by

$$p_{intrpl}(r) = \max_{\tilde{r} \geq r} p(\tilde{r}), \tag{5.9}$$

where $p(\tilde{r})$ is the precision for the recall value $\tilde{r}$. In words, this function interpolates the precision at recall r by using the maximum precision value measured for any recall value $\tilde{r}$ that exceeds the currently observed recall r. Consequently, this results in a precision-recall curve in which the precision values decrease monotonously for a recall from 0 to 1. That interpolation technique reduces the susceptibility to small variations when ranking all the predictions of the network by their confidence score.

5.1.3 Results

After the introduction of all metrics, the next step is to analyze the results of the proposed radar object detection system based on these metrics. For this purpose, the evaluation in this thesis is divided into two main parts. Firstly, an assessment of the individual modules is performed by applying the precision, recall and F_1 score for the classification and segmentation component and the IoU for the predictions of the bounding box estimation unit. Therefore, the evaluation rates the modules one after the other. Basically, the classification performance is measured, that is the ability to classify the radar target list in a patch. Afterwards, the evaluation of the segmentation and bounding box estimation is only performed if the classification result for the patch under consideration is predicted correctly. Since the bounding box estimation depends on the segmentation result of the previous module, false assigned radar targets directly affect the prediction of a bounding box. In case of a poor segmentation and a poor bounding box estimation, the evaluation cannot distinguish whether it is only due to weak segmentation, or due to an additional bad bounding box estimation. And secondly, a rating of the object detection results is carried out, that is scoring the final output of the complete radar object detection system by utilizing the explained interpolated AP metric. In the further course of this thesis, the term AP refers to interpolated AP as defined in Equation (5.8) above. For the specification of the object detection performance, the evaluation applies three different thresholds to determine whether a prediction is considered as a TP or not. A common threshold value for object detection is $IoU_{th} = 0.5$ which is often used in literature, as for example in [EGW+10]. This thesis also evaluates the object detection performance using $IoU_{th} = 0.25$ and $IoU_{th} = 0.1$ as additional thresholds to calculate the metrics. Although, operating at a low threshold can result in relatively inaccurate object detection, this analysis is reasonable. Since a radar sensor may output in quite noisy measurements targets and at the same time also rather few targets per object, it is very difficult to estimate an accurate bounding box. That is why it is useful to observe the performance even at lower thresholds to determine whether the proposed detection approach is able to find objects at all. Further, the evaluation conducts this analysis on different radar data, namely the three which are presented Section 4.2 in detail. Another goal of the evaluation is to justify the design of the radar object detection system as well as the choice of

Description of the Parameter	Value
batch size	32
decay step for learning rate	45000
decay step for batch normalization	45000

Table 5.2: Settings for dataset: Overview of all specific parameters with values which are chosen in the evaluation based on the Radar Dataset.

different parameters. These investigations are carried out only on the Radar Dataset, because the implementation of the object detector as proposed in Section 3.1 is based on that radar data. Subsequently, a separate training is performed for each described radar dataset, and then the evaluation determines the performance of the object detection system. Hence, this thesis provides both, a detailed analysis on the Radar Dataset, and for general replicability also on a publicly available radar data.

The Radar Dataset

The first evaluation measures the performance on the *Radar Dataset* presented in Section 4.2.3, which includes the radar data recorded by the radar sensor described in Section 4.1 above. Since an important issue is that the proposed radar object detection system can be deployed on a real-world application, these results are crucial. Further, this radar data provides the basis for the development and all investigations of the proposed method for object detection solely on radar data. This section presents detailed results of the outcome produced by the object detector's main modules, as well as the final detection result from the complete radar object detection system. Apart from the stage of designing the detector, it includes topics such as the choice of input features, the decision on how to chose the patches and the adjustment of the network's architecture. The ultimate implementation of the radar object detection system is founded on investigations using the Radar Dataset. Accordingly, this section presents the findings of these small experiments during research. This provides the opportunity to comprehend all decisions, but also to assess what effects a different design may have on the performance. Table 5.2 lists all parameters used for training based on the Radar Dataset. The training monitoring supervises the training progress after each epoch by evaluating the current results on the validation dataset, and stops the training when all results are satisfactory.

Table 5.3 shows the number of epochs for the respective models after which the training stopped. Subsequently, the evaluation measures the actual performance on the test dataset with a batch size of one. Since the test data is independent from the

Model	Epochs
PointNet	31
PointNet++	31

Table 5.3: Training epochs: The number of epochs after which the training is stopped for the particular model based on the Radar Dataset.

other data and the training process does not see any sample of this dataset, that is a valid dataset distribution for testing purpose. Table 5.4 shows the results of all single modules for the PointNet and PointNet++ architectures. In the classification and segmentation task, the model using PointNet++ outperforms the PointNet model in almost all metrics. The evaluation for bounding box estimation shows the introduced mIoU, as well as the percentage of how often the measured IoU of individual boxes exceeds a specific threshold. An interesting fact is that the bounding estimation module with PointNet provides slightly better results for the car, bike and pedestrian class. In summary, both versions show good performance for the respective three main components. Nevertheless, the ability of object detection is crucial, this requires a well-working interaction of all modules including preprocessing and postprocessing. Table 5.5 depicts the final results of the entire radar object detection system, that is the measured performance on how well the proposed method is able to detect objects from various classes. With respect to the detection results measured by the AP, the radar object detector with PointNet++ architecture shows better performance for objects independent of their actual category. When considering the individual classes, the PointNet++ model provides a significant improvement for the car category. Whereas, the PointNet has a better performance at a stricter threshold and nearly same results for the other thresholds when considering trucks. A further analysis of the results reveals that the object detector often predicts one or more cars instead of a truck. Naturally, this leads to a poor performance for the detection ability of trucks, because the evaluation considers it as a false prediction. However, in a real application it is much more important that an object is recognized at all, rather than a precise classification of the object. For the purpose of classification, the usage of lidar and camera sensors is better suited. As the results reveal, the proposed radar object detector is certainly capable of recognizing objects, regardless of the PointNet or PointNet++ architecture. Upon examination of the bike and pedestrian group, it turns out that both architectures show a detection performance which is quite low. The poor results in those categories are definitely due to unbalanced distributions of classes in the dataset. Overall, the Radar Dataset contains too few samples to make a meaningful statement regarding these classes. But still, the presented result indicates that the radar object detector is in principle capable of detecting bikes and pedestrian. However, the proof of this requires a larger dataset with an approximately balanced distribution. The key finding from this evaluation

Classification	Class	Samples	Precision	Recall	F_1 Score
PointNet	clutter	16767	**0.848**	**0.976**	**0.908**
	car	12898	0.679	**0.797**	**0.733**
	truck	7250	**0.770**	0.318	0.450
	bike	486	0.621	0.148	0.239
	pedestrian	193	0.064	0.016	0.025
PointNet++	clutter	16767	**0.884**	**0.969**	**0.924**
	car	12898	0.723	**0.902**	**0.803**
	truck	7250	**0.906**	0.349	0.504
	bike	486	0.412	0.218	0.285
	pedestrian	193	0.345	0.104	0.159

Segmentation	Class	Targets	Precision	Recall	F_1 Score	mIoU
PointNet	object	88117	**0.975**	**0.985**	**0.980**	0.911
	car	54246	0.939	0.905	0.922	**0.914**
	truck	33646	**0.951**	**0.908**	**0.929**	0.871
	bike	216	0.912	0.963	0.937	0.934
	pedestrian	9	0.75	1.0	0.857	0.846
PointNet++	object	102292	**0.948**	**0.918**	**0.933**	0.919
	car	64191	0.943	0.917	0.930	**0.921**
	truck	37733	**0.958**	**0.920**	**0.939**	0.886
	bike	306	0.993	0.889	0.938	0.934
	pedestrian	62	1.0	0.726	0.841	0.839

Bounding Box	Class	Boxes	mIoU	IoU ≥ 0.5	IoU ≥ 0.7
PointNet	object	12664	0.624	0.786	0.420
	car	10282	**0.649**	**0.824**	**0.480**
	truck	2307	0.516	0.626	0.165
	bike	72	0.502	0.569	0.097
	pedestrian	3	0.688	1.0	0.0
PointNet++	object	14294	0.629	0.801	0.418
	car	11640	**0.641**	**0.813**	**0.461**
	truck	2528	0.583	0.760	0.237
	bike	106	0.451	0.434	0.085
	pedestrian	20	0.372	0.350	0.050

Table 5.4: Evaluation of the individual modules: Results for classification, segmentation and bounding box estimation components of the radar detection system. This evaluation analyzes the predictions of all individual patches and is performed on the Radar Dataset.

Detection	Class	IoU_{th}	Precision	Recall	F_1 Score	mIoU	AP
PointNet	object	0.5	0.463	0.572	0.511	0.721	0.452
		0.25	**0.601**	0.742	0.664	0.636	0.638
		0.1	**0.733**	**0.907**	**0.811**	0.552	**0.830**
	car	0.5	**0.485**	**0.711**	**0.577**	**0.724**	**0.586**
		0.25	**0.567**	**0.831**	**0.674**	**0.676**	**0.697**
		0.1	**0.582**	**0.854**	**0.692**	**0.663**	**0.703**
	truck	0.5	0.326	0.123	0.179	0.656	0.121
		0.25	0.399	0.151	0.219	0.602	0.128
		0.1	0.503	0.191	0.277	0.513	0.137
	bike	0.5	0.180	0.083	0.113	0.638	0.018
		0.25	0.281	0.129	0.177	0.552	0.057
		0.1	0.303	0.139	0.191	0.528	0.120
	pedestrian	0.5	0.009	0.014	0.011	0.689	0.046
		0.25	0.009	0.014	0.011	0.689	0.046
		0.1	0.009	0.014	0.011	0.689	0.046
PointNet++	object	0.5	0.465	0.604	0.525	0.714	0.519
		0.25	**0.607**	0.790	0.687	0.630	0.680
		0.1	**0.720**	**0.937**	**0.814**	0.558	**0.850**
	car	0.5	**0.491**	**0.755**	**0.595**	**0.716**	**0.625**
		0.25	**0.579**	**0.891**	**0.702**	**0.666**	**0.742**
		0.1	**0.591**	**0.909**	**0.717**	**0.657**	**0.802**
	truck	0.5	0.312	0.107	0.159	0.652	0.087
		0.25	0.431	0.147	0.220	0.585	0.133
		0.1	0.498	0.170	0.254	0.532	0.140
	bike	0.5	0.139	0.103	0.118	0.646	0.031
		0.25	0.250	0.186	0.213	0.522	0.050
		0.1	0.271	0.201	0.231	0.494	0.080
	pedestrian	0.5	0.023	0.042	0.030	0.678	0.005
		0.25	0.063	0.111	0.080	0.497	0.018
		0.1	0.070	0.125	0.090	0.470	0.019

Table 5.5: Evaluation of the entire object detector: Results after performing the complete pipeline of the radar object detection system from the input radar targets to the output radar object hypotheses. This evaluation analyzes the final predictions of radar objects and measures the object detection performance on the Radar Dataset.

Features	**Classification & Segmentation**			**Bounding Box Estimation**		
	Position	RCS	Radial Vel	Position	RCS	Radial Vel
I	✓	✓	✓	✓	✓	✓
II	✓	–	✓	✓	–	✓
III	✓	✓	–	✓	✓	–
IV	✓	–	–	✓	–	–
V	✓	✓	✓	✓	–	–

Table 5.6: Combinations of radar features: Various options to choose input features for the respective modules of the radar object detector.

is that the proposed radar object detection system successfully operates in an area where mainly vehicles, such as cars, are present. All things considered, the method for object detection solely on radar data is absolutely applicable for real operation. In the end, for the application in an automatic driving mode, the environment perception of the vehicle does not only employ the proposed radar object detector, but rather, the system operates additional detectors based on lidar and camera data. Then, a following multi-object tracking system fuses all generated hypotheses into one smoothed detection result to realize an object-based environment perception.

During the designing of the radar object detector, the choice of important features is essential. The radar sensor of this thesis applies the signal processing as described in Figure 2.1 and provides a radar target list. Every measured radar target comprises range and azimuth, and respectively after the internal transformation: the spatial position in Cartesian coordinates, the radial velocity and RCS value. These are all quantities the sensor is able to measure directly. In particular the radar sensor ARS 408-21 may provide additional information, for example a state that indicates whether a radar target is ambiguous or invalid, but also dynamic properties that specify whether the radar target is moving or static. Such information cannot be measured directly, but is obtained by internal algorithms. Since the idea of the proposed radar object detector is to process the radar data as raw measurements, information based on unknown algorithms is not desirable. For this reason, the object detection system solely processes those features which are measurable by the radar sensor, thus the spatial information, the radial velocity and the RCS value.

In order to be able of determining which influence the respective features have on the object detection result, the following evaluation compares the impact of the input features. For this purpose, the models are trained and evaluated with different combinations of features as input. Table 5.6 shows all examined feature options which are carried out within the scope of this experiment for both architectures. The first investigation covers the standard approach that is the proposed one for the final

Model	Option	Epochs
PointNet	Features II	30
PointNet	Features III	27
PointNet	Features IV	31
PointNet	Features V	25
PointNet++	Features II	29
PointNet++	Features III	23
PointNet++	Features IV	31
PointNet++	Features V	31

Table 5.7: Training epochs: The number of epochs after which the training is stopped for different model options based on the Radar Dataset.

radar object detector and which forwards all available features into the networks. The next examinations deal with the cases that either RCS values or radial velocity or both features are omitted. The final experiment uses all features as input for the classification and segmentation network, but the bounding box estimation is only performed on the spatial position of radar targets. This analysis compares the detection metrics of the model using testing data of the Radar Dataset. Table 5.7 lists the training duration for the different model combinations. Furthermore, Table 5.5 presents the object detection results when applying all available radar features. The outcomes for all other feature combinations are depicted in Table 5.8 for the PointNet model and Table 5.9 for the PointNet++ architecture. As expected, the PointNet and PointNet++ models with all measured radar features show the best performance for object detection. An interesting aspect of this experiment is to observe the influence of the specific features. The study *Features IV* proves the importance of the radial velocity and RCS values as features. Without these radar features, the detection results are really poor and reasonable object detection is not possible. Adding even one of these features already improves the ultimate outcome of the object detector significantly. A closer look reveals that the influence of radial velocity, that is investigation *Features II*, leads to a larger improvement than the RCS value, which is examination *Features III*. However, when forwarding all features, the improvement of the performance is again noticeable. The last study, namely *Features V*, examines the effect of radial velocity and RCS values for bounding box estimation. The detection results prove that the features are also important to determine a bounding box, as adding these features provides a further improvement. Especially in the detection ability of cars, the enhancement is extremely significant. All findings discussed in this evaluation are valid for both architectures. Although in comparison of the particular feature combinations, all PointNet++ models always yield slightly better results than those with PointNet what is exactly to be expected.

Detection	Class	IoU_{th}	Precision	Recall	F_1 Score	mIoU	AP
PointNet	object	0.5	0.351	0.522	0.419	0.714	0.380
		0.25	**0.481**	0.716	0.578	0.659	0.598
Features II		0.1	**0.607**	**0.903**	**0.726**	0.525	**0.770**
	car	0.5	**0.374**	**0.666**	**0.479**	**0.715**	**0.455**
		0.25	**0.452**	**0.804**	**0.578**	**0.659**	**0.598**
		0.1	**0.473**	**0.843**	**0.606**	**0.637**	**0.609**
	truck	0.5	0.195	0.041	0.067	0.671	0.091
		0.25	0.268	0.056	0.093	0.590	0.091
		0.1	0.335	0.070	0.116	0.507	0.091
	bike	0.5	0.065	0.036	0.046	0.566	0.018
		0.25	0.148	0.083	0.106	0.470	0.019
		0.1	0.167	0.093	0.119	0.434	0.021
	pedestrian	0.5	0.015	0.056	0.024	0.676	0.003
		0.25	0.048	0.181	0.076	0.500	0.032
		0.1	0.056	0.208	0.088	0.457	0.041
PointNet	object	0.5	0.135	0.483	0.211	0.708	0.277
		0.25	0.186	0.667	0.291	0.609	0.422
Features III		0.1	**0.231**	**0.829**	**0.362**	0.524	**0.552**
	car	0.5	**0.152**	**0.617**	**0.244**	**0.710**	**0.352**
		0.25	**0.189**	**0.770**	**0.304**	**0.664**	**0.432**
		0.1	**0.202**	**0.823**	**0.324**	**0.615**	**0.470**
	truck	0.5	0.083	0.050	0.062	0.631	0.091
		0.25	0.109	0.065	0.081	0.568	0.091
		0.1	0.126	0.075	0.094	0.512	0.091
	bike	0.5	0.0	0.0	0.0	0.0	0.0
		0.25	0.003	0.005	0.004	0.333	0.001
		0.1	0.003	0.005	0.004	0.333	0.001
	pedestrian	0.5	0.0	0.0	0.0	0.0	0.0
		0.25	0.0	0.0	0.0	0.0	0.0
		0.1	0.0	0.0	0.0	0.0	0.0

Detection	Class	IoU_{th}	Precision	Recall	F_1 Score	mIoU	AP
PointNet	object	0.5	0.080	0.350	0.130	0.700	0.067
		0.25	0.123	0.537	0.200	0.577	0.205
Features IV		0.1	**0.166**	**0.727**	**0.271**	0.473	**0.287**
	car	0.5	**0.090**	**0.441**	**0.150**	**0.701**	**0.084**
		0.25	**0.124**	**0.605**	**0.205**	**0.612**	**0.138**
		0.1	**0.136**	**0.665**	**0.226**	**0.573**	**0.155**
	truck	0.5	0.025	0.039	0.031	0.668	0.061
		0.25	0.042	0.065	0.051	0.565	0.061
		0.1	0.047	0.074	0.058	0.522	0.061
	bike	0.5	0.0	0.0	0.0	0.0	0.0
		0.25	0.0	0.0	0.0	0.0	0.0
		0.1	0.0	0.0	0.0	0.0	0.0
	pedestrian	0.5	0.0	0.0	0.0	0.0	0.0
		0.25	0.002	0.014	0.003	0.444	0.000
		0.1	0.002	0.014	0.003	0.444	0.000
PointNet	object	0.5	0.392	0.544	0.456	0.708	0.355
		0.25	**0.547**	0.760	**0.636**	0.607	**0.574**
Features V		0.1	**0.667**	**0.926**	**0.775**	**0.530**	**0.801**
	car	0.5	**0.411**	**0.686**	**0.514**	**0.710**	**0.417**
		0.25	**0.507**	**0.847**	**0.634**	**0.648**	**0.569**
		0.1	**0.529**	**0.885**	**0.662**	**0.629**	**0.586**
	truck	0.5	0.267	0.088	0.132	0.647	0.051
		0.25	0.349	0.114	0.172	0.589	0.129
		0.1	0.411	0.135	0.203	0.527	0.137
	bike	0.5	0.137	0.036	0.057	0.612	0.018
		0.25	0.275	0.072	0.114	0.519	0.046
		0.1	0.294	0.077	0.122	0.491	0.046
	pedestrian	0.5	0.006	0.014	0.009	0.598	0.023
		0.25	0.024	0.056	0.034	0.457	0.023
		0.1	0.024	0.056	0.034	0.457	0.023

Table 5.8: Evaluation of the feature impact: Object detection results using various combinations of radar features as input into PointNets. Table 5.5 shows the findings of option *PointNet Features I* when all the measured features are forwarded to the neuronal networks.

Detection	Class	IoU_{th}	Precision	Recall	F_1 Score	mIoU	AP
PointNet++	object	0.5	0.411	0.574	0.479	0.714	0.423
		0.25	**0.544**	0.760	0.634	0.627	0.613
Features II		0.1	**0.666**	**0.930**	**0.777**	0.544	**0.786**
	car	0.5	**0.433**	**0.708**	**0.537**	**0.718**	**0.548**
		0.25	**0.515**	**0.842**	**0.639**	**0.665**	**0.662**
		0.1	**0.537**	**0.878**	**0.666**	**0.646**	**0.672**
	truck	0.5	0.263	0.159	0.198	0.654	0.092
		0.25	0.385	0.233	0.290	0.575	0.140
		0.1	0.506	0.306	0.382	0.476	0.211
	bike	0.5	0.137	0.052	0.075	0.631	0.091
		0.25	0.233	0.088	0.127	0.512	0.091
		0.1	0.233	0.088	0.127	0.512	0.091
	pedestrian	0.5	0.049	0.056	0.052	0.623	0.011
		0.25	0.111	0.125	0.118	0.494	0.037
		0.1	0.111	0.125	0.118	0.494	0.037
PointNet++	object	0.5	0.291	0.531	0.376	0.712	0.336
		0.25	**0.386**	0.705	0.499	0.625	**0.547**
Features III		0.1	**0.435**	0.794	**0.562**	0.575	**0.604**
	car	0.5	**0.314**	**0.673**	**0.428**	**0.715**	**0.408**
		0.25	**0.379**	**0.811**	**0.516**	**0.658**	**0.541**
		0.1	**0.394**	**0.843**	**0.537**	**0.641**	**0.560**
	truck	0.5	0.175	0.079	0.109	0.630	0.091
		0.25	0.263	0.118	0.163	0.548	0.118
		0.1	0.297	0.133	0.184	0.507	0.122
	bike	0.5	0.012	0.010	0.011	0.599	0.002
		0.25	0.012	0.010	0.011	0.599	0.002
		0.1	0.012	0.010	0.011	0.599	0.002
	pedestrian	0.5	0.003	0.014	0.006	0.557	0.001
		0.25	0.007	0.028	0.011	0.471	0.001
		0.1	0.007	0.028	0.011	0.471	0.001

Detection	Class	IoU_{th}	Precision	Recall	F_1 Score	mIoU	AP
PointNet++	object	0.5	0.124	0.409	0.190	0.718	0.113
		0.25	**0.173**	0.572	0.266	0.614	**0.249**
Features IV		0.1	**0.204**	0.674	**0.313**	0.548	**0.304**
	car	0.5	**0.131**	**0.521**	**0.209**	**0.719**	**0.141**
		0.25	**0.168**	**0.668**	**0.268**	**0.643**	**0.198**
		0.1	**0.181**	**0.720**	**0.289**	**0.610**	**0.230**
	truck	0.5	0.073	0.048	0.058	0.650	0.091
		0.25	0.109	0.072	0.087	0.562	0.091
		0.1	0.132	0.088	0.105	0.499	0.091
	bike	0.5	0.0	0.0	0.0	0.0	0.0
		0.25	0.009	0.010	0.010	0.460	0.036
		0.1	0.009	0.010	0.010	0.460	0.036
	pedestrian	0.5	0.006	0.028	0.009	0.660	0.001
		0.25	0.008	0.042	0.014	0.565	0.002
		0.1	0.008	0.042	0.014	0.565	0.002
PointNet++	object	0.5	0.438	0.574	0.497	0.715	0.389
		0.25	**0.589**	0.773	0.668	0.622	0.602
Features V		0.1	**0.702**	**0.921**	**0.797**	0.550	**0.818**
	car	0.5	**0.468**	**0.703**	**0.562**	**0.719**	**0.500**
		0.25	**0.567**	**0.852**	**0.681**	**0.659**	**0.622**
		0.1	**0.589**	**0.886**	**0.708**	**0.641**	**0.638**
	truck	0.5	0.251	0.160	0.196	0.653	0.115
		0.25	0.367	0.234	0.286	0.564	0.163
		0.1	0.440	0.281	0.343	0.499	0.182
	bike	0.5	0.168	0.113	0.135	0.602	0.031
		0.25	0.298	0.201	0.240	0.521	0.083
		0.1	0.298	0.201	0.240	0.521	0.083
	pedestrian	0.5	0.012	0.014	0.013	0.663	0.002
		0.25	0.024	0.028	0.026	0.543	0.003
		0.1	0.024	0.028	0.026	0.543	0.003

Table 5.9: Evaluation of the feature impact: Object detection results using several combinations of radar features as input into PointNets++. Table 5.5 shows the results of option *PointNet++ Features I* when all the measured features are forwarded to the neuronal networks.

Patch Size	20	22	24	26	30	40	50
Cls Precision Clutter	**0.958**	0.922	0.926	0.947	0.941	0.947	0.906
Cls Precision Car	0.915	**0.922**	0.919	0.91	0.916	0.861	0.891
Cls Precision Truck	0.962	0.959	0.970	0.978	0.955	**0.988**	**0.988**
Cls Precision Bike	0.754	0.92	0.640	**1.0**	0.753	0.775	0.75
Cls Precision Pedestrian	**1.0**	**1.0**	**1.0**	**1.0**	0.574	**1.0**	0.912
Cls Recall Clutter	0.942	**0.953**	0.932	0.938	0.929	0.903	0.926
Cls Recall Car	**0.956**	0.93	0.927	0.95	0.936	0.953	0.914
Cls Recall Truck	0.896	0.862	**0.929**	0.902	0.926	0.831	0.880
Cls Recall Bike	0.639	0.554	0.663	0.494	0.699	**0.747**	0.253
Cls Recall Pedestrian	**1.0**	**1.0**	**1.0**	0.871	**1.0**	**1.0**	**1.0**
Cls F_1 Score Clutter	**0.950**	0.937	0.929	0.942	0.935	0.925	0.916
Cls F_1 Score Car	0.935	0.926	0.923	**0.930**	0.926	0.905	0.902
Cls F_1 Score Truck	0.928	0.908	**0.949**	0.938	0.940	0.903	0.931
Cls F_1 Score Bike	0.693	0.692	0.651	0.661	0.725	**0.761**	0.378
Cls F_1 Score Pedestrian	**1.0**	**1.0**	**1.0**	0.931	0.729	**1.0**	0.954
Seg Precision Car	0.962	**0.968**	0.965	0.961	0.961	0.964	0.965
Seg Precision Truck	0.954	0.960	0.953	0.957	0.957	**0.962**	0.957
Seg Precision Bike	0.965	**1.0**	0.991	0.980	**1.0**	0.993	0.935
Seg Precision Pedestrian	**0.951**	0.860	0.404	0.724	0.593	0.6	0.762
Seg Recall Car	0.957	0.952	0.941	**0.967**	0.945	0.950	0.952
Seg Recall Truck	0.957	**0.962**	0.949	0.955	0.943	0.917	0.906
Seg Recall Bike	0.948	**1.0**	0.922	0.958	0.931	0.859	**1.0**
Seg Recall Pedestrian	0.858	0.920	0.894	0.848	**0.929**	0.876	0.876
Seg F_1 Score Car	0.960	0.960	0.953	**0.964**	0.953	0.957	0.958
Seg F_1 Score Truck	0.955	**0.961**	0.951	0.956	0.950	0.939	0.931
Seg F_1 Score Bike	0.957	**1.0**	0.955	0.969	0.964	0.921	0.966
Seg F_1 Score Pedestrian	**0.902**	0.889	0.557	0.781	0.724	0.712	0.815
BBox IoU Car	0.674	**0.691**	0.681	0.691	0.672	0.682	0.683
BBox IoU Truck	0.626	0.672	0.670	**0.687**	0.654	0.670	0.677
BBox IoU Bike	0.555	**0.629**	0.583	0.537	0.562	0.466	0.608
BBox IoU Pedestrian	0.388	**0.442**	0.059	0.240	0.204	0.202	0.432

Table 5.10: Choice of patch size: Comparison between different patch sizes based on the results produced from the respective modules. The metrics are evaluated for each category after classification (Cls), segmentation (Seg) and bounding box estimation (BBox) tasks.

A question that arises quite early in the development of the radar object detection system, is the size of generated patches. As already mentioned, a patch must be sufficiently large to contain the object of interest in its entirety. However, it is difficult to decide what the optimal patch size is. Because of that, the next experiment tests the performance of respective modules on different patch sizes. Since the complete Radar Dataset was not yet labeled at the time of this evaluation, it is only performed on a subset of the data. Further, this investigation is not based on the test dataset, but on the validation data directly during the training process. Table 5.10 shows the results of individual modules after a training process over 51 epochs of the PointNet model. This evaluation examines precision, recall and F_1 score for the classification and segmentation results, as well as the IoU on the predicted bounding box. These analyses are considered separately for all object classes to determine the change of outcomes by the patch size on the particular class categories. The metric values reveal that the respective modules of the radar object detectors produces best results on patches with a size of 22m the most. Even though the results are quite specific to a subset of the Radar Dataset, this patch size is reasonable, because it is large enough to include almost all kinds of considered objects. That investigation explains why the patch proposal component chooses the heuristic value of 22m as patch size.

The last decision during the design of the radar object detection system concerns the network architecture for classification purpose. As described in Section 3.1.3, the classification network consist of a mini PointNet for feature transforming. In the original PointNet approach, [QSMG17] suggests a T-Net for the input data and one for the alignment of local extracted features. Whereas, in Frustum PointNets, [QLW+18] completely forgoes all transformer networks. This experiment demonstrates the effect and differences on the object detection result when using transformer networks. Table 5.11 shows the various combinations of mini networks to normalize the input data and locale features. The proposal of adding T-Nets for data transformation can only be realized with PointNet architecture. The investigation examines the detection results for all possible combinations. For this purpose, a separate training is conducted for every model, and afterwards its performance is measured using the detection metrics. The classification model in the first study, that is *T-Net I*, does not contain any transformer network as applied in Frustum PointNet [QLW+18]. The two examinations *T-Net II* and *T-Net III* consider the case when exactly one T-Net is deployed, either for input data or for local features. Figure 3.10 visualizes the architecture with feature transforming. Where the alignment of input data happens immediately when the data is inserted into the network, that is before applying the first shared MLPs. Except for the fact that different dimensions are necessary, the alignment of input data is identical to the feature transformation. The last investigation, which is *T-Net IV*, examines the usage of both transformer as proposed in PointNet [QSMG17]. Table 5.12 lists the training duration for the different variations. Further, Table 5.5 presents the results for study *T-Net III*, which is the configuration that is used in the final version of the radar object detection

T-Net	Classification PointNet	
	Input Transforming	Feature Transforming
I	–	–
II	✓	–
III	–	✓
IV	✓	✓

Table 5.11: Choice of T-Nets: Various options to combine transformer networks in the classification network of the radar object detector.

Model	Option	Epochs
PointNet	T-Net I	30
PointNet	T-Net II	31
PointNet	T-Net IV	26

Table 5.12: Training epochs: The number of epochs after which the training is stopped for the model variations based on the Radar Dataset.

system. Further, Table 5.13 illustrates the outcomes for all other T-Net combinations. This evaluation reveals that the various architectures actually differ only slightly in their results. Since the patches are comparable to the frustums from [QLW+18], it is actually to be expected that according to the statements in [QLW+18], no transformer networks should be necessary. This assertion can neither be confirmed nor clearly refuted with the presented analysis on the Radar Dataset. For that reason, this thesis does not recommend which model design is best suited for the object detection task in radar data. The proposed radar object detector in this thesis employs the variant T-Net III for two reasons. Firstly, since the described rotation of patches, as visualized in Figure 3.4 is applied, the input data is normalized regarding the position. The normalization of radial velocity and RCS value by an affine transformation is quite hard to follow. That is why the expectation of a large improvement using input transformation network is unlikely. Secondly, the detection metrics reveal that with a feature transforming, the object predictions are better compared to option T-Net I and T-Net II. In comparison to T-Net IV, the feature transformer yields marginally worse detection results, but for the universal object class, both model versions perform the same. As already emphasized, that decision is difficult to make and is here optimized for the challenges in this thesis. As a matter, a definite statement requires further research on a radar dataset with more samples.

A good performance of the radar object detector is essential for the application on real-world data. However, for actual operation in live mode on an automated vehicle,

Detection	Class	IoU_{th}	Precision	Recall	F_1 Score	mIoU	AP
PointNet	object	0.5	0.394	0.574	0.467	0.712	0.433
		0.25	**0.521**	0.761	0.619	0.624	0.632
T-Net I		0.1	**0.642**	**0.938**	**0.762**	0.540	**0.823**
	car	0.5	**0.411**	**0.731**	**0.526**	**0.713**	**0.563**
		0.25	**0.487**	**0.867**	**0.624**	**0.663**	**0.678**
		0.1	**0.501**	**0.892**	**0.642**	**0.650**	**0.685**
	truck	0.5	0.225	0.053	0.086	0.635	0.091
		0.25	0.326	0.078	0.125	0.547	0.091
		0.1	0.428	0.102	0.164	0.458	0.131
	bike	0.5	0.097	0.057	0.072	0.602	0.017
		0.25	0.159	0.093	0.117	0.490	0.091
		0.1	0.168	0.098	0.124	0.477	0.091
	pedestrian	0.5	0.0	0.0	0.0	0.0	0.0
		0.25	0.0	0.0	0.0	0.0	0.0
		0.1	0.0	0.0	0.0	0.0	0.0
PointNet	object	0.5	0.421	0.572	0.485	0.714	0.438
		0.25	**0.558**	0.759	0.643	0.625	0.632
T-Net II		0.1	**0.680**	**0.925**	**0.784**	0.544	**0.826**
	car	0.5	**0.454**	**0.721**	**0.557**	**0.716**	**0.571**
		0.25	**0.538**	**0.854**	**0.660**	**0.665**	**0.681**
		0.1	**0.554**	**0.880**	**0.680**	**0.651**	**0.687**
	truck	0.5	0.208	0.086	0.122	0.629	0.091
		0.25	0.300	0.125	0.176	0.550	0.119
		0.1	0.385	0.160	0.226	0.467	0.127
	bike	0.5	0.167	0.072	0.101	0.653	0.018
		0.25	0.238	0.103	0.144	0.574	0.113
		0.1	0.238	0.103	0.144	0.574	0.113
	pedestrian	0.5	0.009	0.028	0.013	0.608	0.023
		0.25	0.009	0.028	0.013	0.608	0.023
		0.1	0.009	0.028	0.013	0.608	0.023

Detection	Class	IoU_{th}	Precision	Recall	F_1 Score	mIoU	AP
PointNet	object	0.5	0.438	0.601	0.507	0.711	0.498
		0.25	**0.577**	0.791	0.667	0.625	0.653
T-Net IV		0.1	**0.688**	**0.942**	**0.795**	0.553	**0.831**
	car	0.5	**0.461**	**0.762**	**0.574**	**0.713**	**0.599**
		0.25	**0.542**	**0.896**	**0.676**	**0.665**	**0.708**
		0.1	**0.553**	**0.914**	**0.689**	**0.656**	**0.766**
	truck	0.5	0.242	0.061	0.098	0.638	0.091
		0.25	0.354	0.089	0.142	0.555	0.091
		0.1	0.490	0.123	0.197	0.447	0.136
	bike	0.5	0.164	0.093	0.118	0.629	0.018
		0.25	0.273	0.155	0.197	0.528	0.080
		0.1	0.273	0.155	0.197	0.528	0.080
	pedestrian	0.5	0.0	0.0	0.0	0.0	0.0
		0.25	0.007	0.014	0.009	0.308	0.002
		0.1	0.007	0.014	0.009	0.308	0.002

Table 5.13: Evaluation of the T-Net impact: Object detection results using several combinations of T-Net for alignment in the classification PointNet. Table 5.5 shows the results of option *T-Net III* when feature transforming of the extracted local features is performed.

the capability of real-time processing is at least of the same importance. Real-time operation is ensured when a measurement cycle is completely processed before the next radar measurements are pending. Since the radar sensors in this thesis delivers new measurements about every 70ms, the object detector has to process a complete radar target list in this time at the latest. The following evaluation demonstrates whether the proposed radar object detection system is capable of doing this. As described in Section 3.2, it is possible to use half precision floating point format or single precision floating point format for the calculations inside the neuronal networks. Table 5.14 presents the inference time in real operating mode using three radar sensors of the type ARS 408-21 simultaneously. As expected, the radar object detector is slightly faster with half precision, but with averagely significantly of less than 3ms, not that much. Nevertheless, the experiment shows that the radar object detection system is definitely real-time capable at both precision formats with impressive inference times. With that successful experiment, the real application of the object detection system is quantitatively proven. Moreover, this radar object detector is already successfully deployed in the automated vehicle of Ulm University.

Inference Time	Radar FL	Radar FC	Radar FR	Full radar setup
Float16 Precision	18.5ms	18.3ms	18.4ms	18.4ms
Float32 Precision	21.4ms	20.8ms	21.2ms	21.1ms

Table 5.14: Comparison of the inference times: Measured average times of the object detector during inference on different sensors instances in real operating mode, that are mounted on the front left (FL), front center (FC) and front right (FR) of the experimental vehicle. The evaluation compares resulting inference times when using half precision floating point format (FP16) and single precision floating point format (FP32) for the trained model parameters.

Description of the Parameter	Value
batch size	16
decay step for learning rate	50000
decay step for batch normalization	50000

Table 5.15: Settings for dataset: Overview of the parameters with values which are utilized in the evaluation based on the Astyx dataset.

Model	Epochs
PointNet	6
PointNet++	5

Table 5.16: Training epochs: The number of epochs after which the training is stopped for the particular model based on the Astyx dataset.

Astyx HiRes2019 Dataset

This section shows the performance of the radar object detection system on the public Astyx HiRes2019 dataset. Section 4.2.1 presents the corresponding radar data after dataset preparation. It is already mentioned that the radar data is sometimes extremely unanticipated and the following object detection results must be treated with caution. Nevertheless, this evaluation presents the object detection results for the sake of completeness. Table 5.15 lists the general parameters for the training. Since a target list of one measurement cycle comprises a lot of radar targets, the batch size is chosen smaller to prevent an overflow of the RAM in the GPU. In order to avoid an overfitting of the models, the training process is stopped when the parameters tend to overfit based on the average loss values over the entire validation

Classification	Class	Samples	Precision	Recall	F_1 Score
PointNet	clutter	12394	0.555	**0.857**	0.673
	car	17747	**0.788**	**0.635**	**0.703**
	truck	3209	0.194	0.010	0.019
	bike	207	0.0	0.0	0.0
	pedestrian	68	0.0	0.0	0.0
PointNet++	clutter	12394	0.555	**0.923**	0.693
	car	17747	**0.899**	**0.659**	**0.761**
	truck	3209	0.143	0.001	0.001
	bike	207	0.0	0.0	0.0
	pedestrian	68	0.0	0.0	0.0

Segmentation	Class	Targets	Precision	Recall	F_1 Score	mIoU
PointNet	object	904995	0.788	0.744	0.765	0.757
	car	897809	0.788	**0.745**	**0.766**	**0.757**
	truck	7186	**0.873**	0.601	0.712	0.592
	bike	0	–	–	–	–
	pedestrian	0	–	–	–	–
PointNet++	object	937683	0.820	0.776	0.798	0.784
	car	937311	0.820	0.776	0.800	**0.784**
	truck	372	**0.987**	**0.790**	**0.878**	0.706
	bike	0	–	–	–	–
	pedestrian	0	–	–	–	–

Bounding Box	Class	Boxes	mIoU	IoU ≥ 0.5	IoU ≥ 0.7
PointNet	object	11303	0.445	0.445	0.166
	car	11271	**0.445**	**0.445**	**0.166**
	truck	32	0.398	0.531	0.0
	bike	0	–	–	–
	pedestrian	0	–	–	–
PointNet++	object	11699	0.462	0.508	0.172
	car	11697	**0.462**	**0.508**	**0.172**
	truck	2	0.266	0.0	0.0
	bike	0	–	–	–
	pedestrian	0	–	–	–

Table 5.17: Evaluation of the individual modules: Results for classification, segmentation and bounding box estimation components of the radar object detector. This evaluation analyzes the predictions concerning all extracted patches and is performed on Astyx data.

Detection	Class	IoU_{th}	Precision	Recall	F_1 Score	mIoU	AP
PointNet	object	0.5	0.098	0.250	0.141	0.667	0.142
		0.25	0.204	0.520	0.293	0.506	0.305
		0.1	**0.255**	0.652	**0.367**	0.441	**0.388**
	car	0.5	**0.100**	**0.273**	**0.147**	**0.667**	**0.145**
		0.25	**0.204**	**0.554**	**0.298**	**0.510**	**0.312**
		0.1	**0.246**	**0.670**	**0.360**	**0.454**	**0.378**
	truck	0.5	0.0	0.0	0.0	0.0	0.0
		0.25	0.0	0.0	0.0	0.0	0.0
		0.1	0.035	0.048	0.040	0.147	0.004
	bike	0.5	0.0	0.0	0.0	0.0	0.0
		0.25	0.0	0.0	0.0	0.0	0.0
		0.1	0.0	0.0	0.0	0.0	0.0
	pedestrian	0.5	0.0	0.0	0.0	0.0	0.0
		0.25	0.0	0.0	0.0	0.0	0.0
		0.1	0.0	0.0	0.0	0.0	0.0
PointNet++	object	0.5	**0.157**	0.281	0.202	0.653	0.185
		0.25	0.287	0.557	0.379	0.513	0.366
		0.1	0.346	0.610	0.442	0.463	0.428
	car	0.5	**0.156**	**0.304**	**0.207**	**0.654**	**0.202**
		0.25	0.287	**0.557**	**0.379**	**0.513**	**0.366**
		0.1	0.340	**0.652**	**0.447**	**0.466**	**0.443**
	truck	0.5	0.0	0.0	0.0	0.0	0.0
		0.25	**1.0**	0.048	0.091	0.275	0.091
		0.1	**1.0**	0.048	0.091	0.277	0.091
	bike	0.5	0.0	0.0	0.0	0.0	0.0
		0.25	0.0	0.0	0.0	0.0	0.0
		0.1	0.0	0.0	0.0	0.0	0.0
	pedestrian	0.5	0.0	0.0	0.0	0.0	0.0
		0.25	0.0	0.0	0.0	0.0	0.0
		0.1	0.0	0.0	0.0	0.0	0.0

Table 5.18: Evaluation of the entire object detection system: Results after forwarding the radar targets into the whole radar object detection system to identify the radar object hypotheses. This evaluation analyzes the overall detection results based on the Astyx dataset.

data. Table 5.16 shows the number of epochs for each model. It is rather conspicuous that overfitting of parameters starts after a few epochs with this dataset. This is a sign that the Astyx radar dataset is not suitable for deep learning methods, because it is far too limited in size and quality. Moreover, the dataset contains not enough distinct radar samples to be able to train any model with generalization capability.

After freezing all model parameters, the evaluation forwards the testing data into the radar object detector. Table 5.17 illustrates the results of the classification, segmentation and bounding box estimation modules. Both, PointNet and PointNet++ are able to classify clutter patches as well as regions with a car in its center. But neither of them is able to detect any bike or any pedestrians. Furthermore, since only two samples are correctly classified, the ability to recognize trucks is not given. However, both architectures provide good segmentation results of correct classified radar target lists. Although, the bounding box estimation module predicts a box for each detected object, but the overlap with the ground truth is generally unconvincing. Table 5.18 presents the overall object detection performance for both model architectures. The results reveal that the trained object detector is only capable of detecting cars in the Astyx radar data. Even when applying a low IoU threshold in the AP metric, the detection performance is not conclusive. But as expected, the PointNet++ model provides slightly enhanced object detection results.

NuScenes Dataset

The evaluation on the nuScenes dataset, which is described in Section 4.2.2, is an important part of this thesis, since the radar data fulfills two major aspects. Firstly, the nuScenes dataset is publicly available, which means that all results obtained are reproducible. And secondly, the radar data originates from the same sensor type that is described in Section 4.1 above. However, this data differs from that in the Radar Dataset, because the sensor is differently configured and provides no more than half the number of radar targets per measurement cycle. The training with the radar data extracted from the nuScenes dataset is performed in the same way as based on the Radar Dataset. Table 5.19 shows the general training parameters. Further, Table 5.20 lists the number of training epochs for the respective models. It is interesting that the results do not improve even after a minor number of epochs. This is due the fact that the various preprocessed training samples contain only sparse radar target lists and the networks produce comparatively similar features. In order to avoid an overfitting of the model parameters, training is stopped in time.

After the training procedure and finding the optimal parameters, the particular models are evaluated on testing data. Table 5.21 presents the results of the classification, segmentation and bounding box estimation module for PointNet and PointNet++

Description of the Parameter	Value
batch size	32
decay step for learning rate	125000
decay step for batch normalization	125000

Table 5.19: Settings for dataset: Overview of all the specific parameters with values chosen for the evaluation based on the nuScenes dataset.

Model	Epochs
PointNet	9
PointNet++	11

Table 5.20: Training epochs: The number of epochs after which the training is stopped for a particular model based on the nuScenes dataset.

model. In the classification of radar patches, both architectures show roughly the same performance. What is striking about this evaluation is that PointNet performs slightly better in segmentation and bounding box estimation for car and truck classes, as well as the comprehensive category for objects. For the bike class, a classification with neither architecture is feasible. However, since there are only a few extracted examples available, these results are not really representative. Although there are not that many samples for pedestrians, a recognition is definitely possible with both architectures. Table 5.22 shows the final detection outcomes of the complete radar object detector. This analysis again reveals the same behavior as the evaluation of the individual modules. The PointNet model yields better results than the PointNet++ architecture for the classes car, truck and pedestrian, as well as the general category comprising all objects. No detection is possible for the bike class, no matter which type of architecture is used. Due to the small number of bike samples in this dataset, the results for this class must also be treated with caution. Despite that, this method is still capable of detecting the other object classes and especially objects in general.

The nuScenes dataset does not offer optimal conditions for object detection in radar data as the measured target lists are extremely sparse. Nevertheless, this experiment demonstrates that the approach presented in this thesis allows an object detection in radar data originating from the nuScenes dataset. Compared to the object detection results with data from the Radar Dataset, the nuScenes dataset provides significantly worse performance. However, it must be considered that the Radar Dataset contains much denser radar target lists and these are also of much higher quality. Especially with the proposed technique it is true that the more radar targets an object generates, the more likely it is that the radar object detector identifies an accurate detection.

Classification	Class	Samples	Precision	Recall	F_1 Score
PointNet	clutter	64735	**0.918**	**0.930**	**0.924**
	car	41173	0.784	**0.837**	**0.810**
	truck	26820	**0.787**	0.700	0.741
	bike	421	0.150	0.007	0.014
	pedestrian	1144	0.741	0.552	0.633
PointNet++	clutter	64735	**0.924**	**0.921**	**0.922**
	car	41173	**0.776**	**0.857**	**0.814**
	truck	26820	0.791	0.683	0.733
	bike	421	0.258	0.019	0.035
	pedestrian	1144	0.639	0.628	0.633

Segmentation	Class	Targets	Precision	Recall	F_1 Score	mIoU
PointNet	object	265055	0.902	0.938	0.919	0.891
	car	111731	**0.949**	**0.979**	**0.964**	**0.955**
	truck	151820	0.868	0.907	0.887	0.815
	bike	6	0.750	1.0	0.857	0.860
	pedestrian	1498	0.865	0.975	0.917	0.909
PointNet++	object	263393	0.896	0.933	0.914	0.885
	car	115201	**0.943**	**0.970**	**0.956**	**0.946**
	truck	146488	0.860	0.904	0.881	0.804
	bike	28	1.0	1.0	1.0	1.0
	pedestrian	1676	0.889	0.958	0.922	0.915

Bounding Box	Class	Boxes	mIoU	IoU ≥ 0.5	IoU ≥ 0.7
PointNet	object	53870	0.554	0.642	0.250
	car	34471	**0.585**	**0.715**	**0.310**
	truck	18764	0.501	0.520	0.146
	bike	3	0.184	0.0	0.0
	pedestrian	632	0.418	0.350	0.038
PointNet++	object	54312	0.517	0.567	0.184
	car	35280	**0.566**	**0.687**	**0.278**
	truck	18306	0.426	0.343	0.007
	bike	8	0.269	0.0	0.0
	pedestrian	718	0.437	0.376	0.053

Table 5.21: Evaluation of the individual modules: Results for classification, segmentation and bounding box estimation parts of the radar detection system. This rating investigates the predictions of all individual patches and is accomplished on the nuScenes dataset.

Detection	Class	IoU_{th}	Precision	Recall	F_1 Score	mIoU	AP
PointNet	object	0.5	0.208	0.518	0.297	0.673	0.355
		0.25	**0.314**	0.782	**0.448**	0.575	0.562
		0.1	**0.356**	**0.887**	**0.508**	0.529	0.640
	car	0.5	**0.212**	**0.612**	**0.315**	**0.681**	**0.435**
		0.25	**0.285**	**0.821**	**0.423**	**0.608**	**0.645**
		0.1	**0.301**	**0.866**	**0.446**	**0.586**	**0.670**
	truck	0.5	0.134	0.200	0.161	0.634	0.077
		0.25	0.271	0.404	0.325	0.507	0.242
		0.1	0.324	0.483	0.388	0.454	0.280
	bike	0.5	0.0	0.0	0.0	0.0	0.0
		0.25	0.0	0.0	0.0	0.0	0.0
		0.1	0.048	0.006	0.010	0.150	0.015
	pedestrian	0.5	0.076	0.165	0.104	0.601	0.030
		0.25	0.190	0.409	0.259	0.475	0.111
		0.1	0.216	0.466	0.295	0.441	0.121
PointNet++	object	0.5	0.212	0.480	0.294	0.665	0.328
		0.25	**0.335**	0.757	0.465	0.597	0.546
		0.1	**0.395**	**0.893**	**0.547**	0.503	0.637
	car	0.5	**0.242**	**0.604**	**0.346**	**0.673**	**0.424**
		0.25	**0.330**	**0.824**	**0.471**	**0.597**	**0.636**
		0.1	**0.351**	**0.874**	**0.501**	**0.574**	**0.658**
	truck	0.5	0.064	0.100	0.078	0.577	0.060
		0.25	0.227	0.355	0.277	0.437	0.181
		0.1	0.294	0.460	0.359	0.376	0.245
	bike	0.5	0.0	0.0	0.0	0.0	0.0
		0.25	0.105	0.010	0.020	0.280	0.091
		0.1	0.105	0.010	0.020	0.280	0.091
	pedestrian	0.5	0.069	0.225	0.106	0.606	0.019
		0.25	0.167	0.544	0.256	0.478	0.103
		0.1	0.184	0.597	0.281	0.453	0.118

Table 5.22: Evaluation of the entire object detection system: Results after performing the complete pipeline of the radar object detector from input radar target list to radar object hypotheses. This evaluation analyzes the detection results based on nuScenes data.

5.1.4 Discussion

The detailed evaluation confirms that the radar object detection system is capable of recognizing objects in sparse radar target lists. In order to prove this capability, the system is evaluated on three different radar datasets. Two of them, the Astxy dataset and the nuScenes dataset, are publicly accessible. This allows anyone to reproduce the presented results. Indeed, beginning with the Astxy dataset, the radar object detector shows rather poor performance on this radar data. However, this is because the quality of the Astyx dataset is poor for two reasons. Firstly, the radar targets are not corrected using an ego motion compensation. Secondly, the measured radial velocities are not reasonable, because the velocities do not match to the measured object. For this reason, the results on the Astyx data is not considered to evaluate the actual performance of the radar object detection system in the context of this thesis. In contrast, the results on the nuScenes dataset are extremely promising. The radar object detector provides good results in detecting objects, and works especially well for vehicles, but still acceptably for trucks. Unfortunately, the radar target lists in the nuScenes dataset are rather sparse in general, such that a detection of bikes and pedestrians is not really possible. Nevertheless, it is a great success to demonstrate that the radar object detector performs well on a public radar dataset.

The system performs exceptionally well on the self-generated radar dataset, which is called the *Radar Dataset*. In this dataset, the radar target lists are much denser than in nuScenes dataset, which allows a much more reliable detection of objects. As a result, the radar object detector is able to recognize objects from different classes. However, the detection system also has difficulties to detect bikes and pedestrians, whereby this is caused by the fact that the dataset is unbalanced with regard to these classes. The best results by far are achieved with vehicles as well. Compared to the nuScenes dataset, these detection results are significantly improved. In summary, the radar object detection system demonstrates that it is possible to detect objects only based on radar data. Furthermore, the proposed technique for radar object detection is real-time capable, and thus, it is suitable for a deployment in real-world systems.

5.2 Radar Object Tracking

This section presents the results of the multi-object tracking using the output of the radar object detection system proposed in Section 3.1.2 of this thesis. The tracking system applies the measurement model, that is introduced in Section 3.3, which allows to estimate the motion and the extent of detected objects. In order to qualitatively assess this tracking system, the evaluation is divided as follows.

Description of the Parameter		Value
extraction probability	object detection	0.01
	tracked object	0.05
standard deviation	acceleration	5m/s^s
	yaw acceleration	0.5rad/s^s
persistence probability	track	0.99
detection probability	object	0.7
clutter intensity	measurement	0.0001

Table 5.23: Settings for the multi-object tracking system: Overview of all the parameters with values which are chosen for the LMB filter. A detailed explanation of these parameters is given in [Reu14].

Section 5.2.1 describes the vehicle setup for this investigation. Section 5.2.2 highlights the improvements achieved by multi-object tracking of the radar object detections over time. And finally, Section 5.2.3 concludes the findings with a brief discussion.

5.2.1 Experimental Vehicle Setup

The setup for this evaluation consists of a experimental vehicle with three high-resolution radar sensors, as presented in Section 4.1, to analyze the improvements when applying tracking for a further processing of the obtained radar object detections. The multi-object tracker is a standard LMB filter, as suggested in [Reu14] and briefly introduced in Section 2.3, with the incorporation of object classification based on the Dempster-Shafer theory from [DLR77; Sha76]. Furthermore, a feasible approach to consider object classification in a tracking system is introduced in [Mun11]. That advanced version of the LMB filter is applied for tracking purpose of the radar object detection system, however, the concrete implementation is not a part of the thesis.

In order to avoid withholding information from the object detector, the number of radar targets per patch is set to the maximum value over all patches of a measurement cycle. Whenever a patch contains less targets, the existing ones are upsampled. The choice of parameters for the multi-object tracking system are application-specific and all respective values are heuristically determined. Table 5.23 lists all parameters which are used for the LMB filter. The multi-object tracker processes all object detections provided by the radar object detection system. The goal is to discard as few as object hypotheses as possible, however, absolute unrealistic ones are rejected to limit the calculation complexity. Further, the spatial distribution of an object is

described by using a GM, where the LMB filter outputs the component with the highest weight as current estimate. The extraction probability is deliberately set to a low value which allows a better comparison of the tracking result with the outcomes of the radar object detector. The prediction utilizes the CTRV model to describe the motion of existing tracks. Finally, the LMB filter performs a multi-object tracking based on object hypotheses which are identified in different radar target list. As a consequence, the object tracking performs a data fusion to ensures that all extracted information from the radar object detection system, which is deployed on three equal radar sensors into one consistent and smoothed result for environment perception.

The evaluation for multi-object tracking on radar object hypotheses is of a qualitative nature, because no ground truth data is available for complete tracking sequences. Although the radar data is labeled for training purpose, it is not suitable for the tracking evaluation, as not all consecutive radar frames are annotated. Consequently, the analysis compares individual traffic scenes that describe distinctive situations and demonstrate the benefits of a subsequent multi-object tracking. The focus of this evaluation is not to prove how much a tracking system improves the recognition of objects, but rather to emphasize that the extracted detections from the radar object detector is processable with a generally established algorithm for object perception. The ability of an LMB filter to fuse measurements from different origins and smooth the resulting trajectories of tracks over time is already demonstrated in [Reu14]. The analysis of this thesis reveals in a few examples that an LMB filter allows combining object hypotheses extracted from three different radar target lists, which are partly contradictory, into one reasonable detection result. The validation is absolutely adequate to demonstrate this fact. In summary, the evaluation examines three real traffic scenes which occur in the region around Ulm University. Accordingly, the testing scenarios take place in rural and urban areas with common traffic situations.

5.2.2 Improvements

The first situation is located out of town on a rural road with an additional left turn lane. The ego vehicle is on the lane that leads straight ahead and travels at a speed of about 70km/h. Two vehicles are in the same lane at a larger distance and four vehicles take the second lane to turn left. Figure 5.1 visualizes the radar targets from three sensors and a front view camera image, together with the detection results of the respective radar detectors and the final object tracking result. Especially for the two left-turning vehicles the object detection systems do not all produce the same results, because some object hypotheses have a false orientation. Nevertheless, due to the objects already tracked and the prediction performed before the measurement update, the LMB filter is able to confirm the existing objects by the most probable radar object detections. The second vehicle from the left demonstrates the benefit of

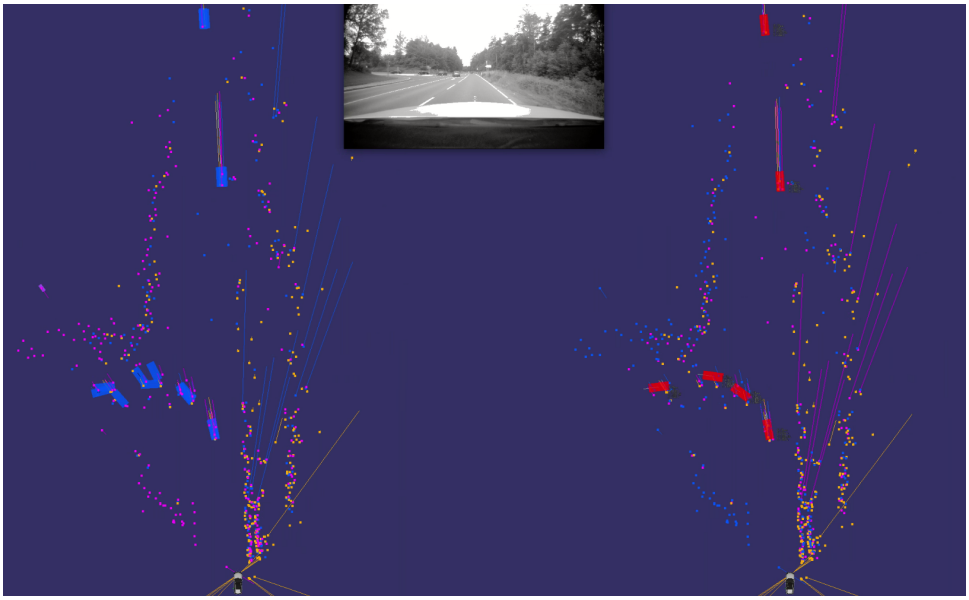


Figure 5.1: Radar object tracking in a rural area: The left side visualizes all object hypotheses which are identified in three radar target lists from different sensors as blue rectangles. The right side illustrates the multi-object tracking result based on all radar object detections, where tracks are visualized as red rectangles.

a multi-object tracking system. Although, two of three radar object detectors provide a hypothesis with an incorrect estimate for the orientation, the single detection has a stronger influence during the filter update, as this object hypothesis fits better to the currently existing track. Finally, when comparing the tracking result with the real traffic scene, it is confirmed that the LMB filter estimates all objects in the environment correctly, both in the near range and in the far range of the radar sensors.

The second traffic scenario takes place in the middle of the campus at Ulm University. This is an interesting scene, because objects of several classes are present, namely, vehicles, pedestrians and even a streetcar, which is assigned to the truck category. In the lane of the ego vehicle three vehicles are driving ahead in a row. Further, on the tracks to the left of that road lane a streetcar is moving towards the ego vehicle. On the right side on the sidewalk, two pedestrians are walking close to the vehicle and in the same direction. Figure 5.2 illustrates the detection result of all radar object detectors, together with the outcome after the fusion of all object hypotheses processed by a multi-object tracking system. This example also demonstrates that

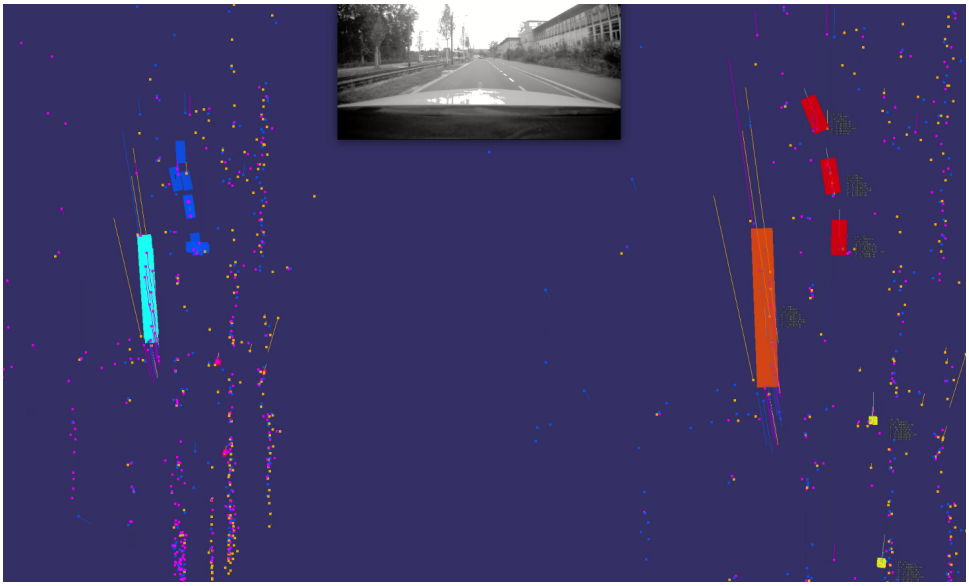


Figure 5.2: Radar object tracking in an urban area: On the left side, the object detections are shown. The radar detector recognizes objects with different classifications, where cars are blue, trucks are cyan and pedestrians are magenta. The tracking system combines all detections to one result. On the right side, the tracking result is shown as red cars, orange trucks as well as yellow pedestrians.

the LMB filter combines various radar object hypotheses from different sources into a single tracking result. Two fundamental capabilities of an object tracking algorithm are highlighted. Firstly, the vehicles again verify that the object tracker allows to deal with contradictory measurement input and finds the optimal solution, which matches the reality here as well. Secondly, the LMB tracker does not necessarily need a hypothesis per object from each radar object detector at all times. In this situation, the pedestrians do not generate enough radar reflections in each sensor, so that not all radar object detection systems yield a hypothesis for both of them. Since the LMB filter already estimates tracks for the two pedestrians from previous time steps, just one of three possible detections may be sufficient to confirm the presence of the pedestrian. This proves that the multi-object tracking system is able to provide a robust and realistic estimation result that involves both an accurate estimate for dynamic motion data and the classification information for all objects.

Last but not least, the third traffic situation impressively reveals the full potential of environment perception with the radar setup proposed in this thesis. Due to the

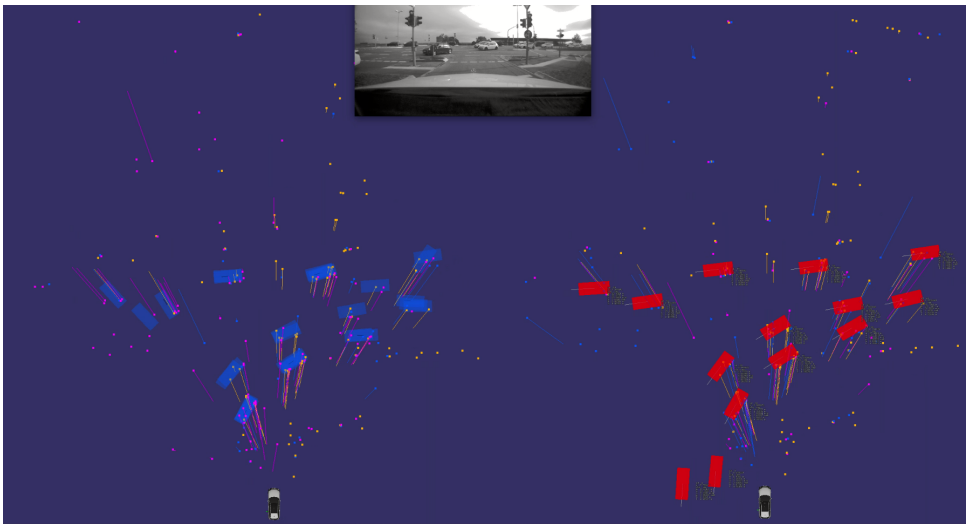


Figure 5.3: Radar object tracking at a multi-lane intersection: The left side illustrates all and even partially contradictory object hypotheses from the radar object detectors as blue rectangles. The right side demonstrates the more robust multi-object tracking result where the tracked objects which are depicted as red rectangles.

deliberately designed redundant coverage of the sensors' FOV in the near range, it is actually possible to track a multitude of objects in that area. In this scene, the ego vehicle is located at the outskirts of Ulm University campus and standing at a rural intersection. From this point of view, the crossroad consists of four lanes on the right, where two lanes lead straight ahead and two of them turn to the left in direction of the ego vehicle. This example refers to the same sequence from which the scene in Figure 3.1 originates, which is used to explain the complete pipeline of radar object detection system step by step. Since the ego vehicle is positioned directly in front of the stop line, the three radar sensors have a direct view on the entire intersection. This is a quite challenging scenario for environmental perception, because there simultaneously are numerous objects in a confined space at the crossroad. Further, all the objects are close to each other and partially obscure one another. Figure 5.3 presents the object detection result in the entire radar data and the multi-object tracking output. In combination, at least one object hypothesis is provided by one of the radar object detectors for each vehicle that is located in the covered area of the respective sensors. Due to the tracking information previously accumulated, the multi-object tracker is able to realistically represent the motion data of every individual vehicle. The two vehicles at the bottom edge are no longer in one of the sensor's FOV, which is why no radar reflections are produced and therefore none of

the object detectors can identify any hypothesis at all. However, the radar detection systems previously recognize these objects such that the LMB filter sets up tracks and calculates an estimate for them. This allows the tracker to continue outputting tracks for both vehicles, where the estimation is fully based on the prediction. In summary, this example of a real traffic situation demonstrates again how successfully the multi-object tracking algorithm stabilizes detected hypotheses from the radar object detection system over time. Thus, a single measurement with an incorrect state, for example an object detection with wrong orientation, or even missing measurements for an actually existing object do not cause any trouble for the tracker. In summary, the multi-object tracking system processes all radar object detections to produce an accurate and robust environment perception model at an object level.

5.2.3 Discussion

This qualitative evaluation already demonstrates the positive impact of a multi-object tracking system when combining information from several sensors. Although, the results of the individual radar object detection system are sometimes contradictory, the object tracking system is able to combine them appropriately to achieve the best possible result. Another advantage is that even if the object detectors do not provide detections of previously tracked objects at all times, the tracking system is still able to further estimate the object's motion by predicting it. As soon as object detections are available again, the prediction is corrected accordingly. Finally, the multi-object tracker not only combines the detection results from different sources appropriately, but also filters the motion of the objects over time, so that it yields a smoothed trajectory for each object. In summary, an object tracking system is always recommended for a robust and reliable perception of objects in the environment.

Chapter 6

Conclusion and Future Work

This final chapter of the thesis summarizes the research and findings of the dissertation. The first section recaps all novel developments and their results concerning object detection in radio detection and ranging (radar) data. Although, the new technologies perform well, there is still potential for improvements. The second part suggests approaches on how to enhance the radar object detection system further.

6.1 Conclusion

Environment perception is essential for automated driving functions, that means the vehicle must recognize which objects are located in the current surroundings. Modern high-resolution automotive radar sensors are widely installed in today's vehicles and are well suited to solve the object detection task. This thesis proposes a novel technique to detect objects of various classes in sparse radar target lists produced from that sensors. The radar object detection system consist of three main components for classification, segmentation and bounding box estimation. The classification module divides the entire radar target list into regions of interest and classifies these areas, which implies that one category is assigned to each of the corresponding sublists. The radar targets of a patch and the associated classification information are forwarded to the segmentation. This component determines for each radar target, whether it belongs to the object of interest or not, that means the segmentation classifies each individual radar target. After a masking process, which removes all radar targets that do not belong to the object, the bounding box estimation module estimates an amodal 2D bounding box in two steps. Firstly, a Transformer PointNet (T-Net) specifies the object's center in the extracted radar targets and the following transformation shifts the target list into this center. Secondly, a regression PointNet determines the parameters for the bounding box based on the classification information and segmented radar target list. The first unit, the patch proposal,

generates a region of interest for each radar target, which results in various object hypotheses. Since an object generates several radar reflections, this leads to the fact that different bounding boxes are generated for one object in reality. For that reason, the last unit, the object extraction, combines all recognized hypotheses and outputs exactly one detection per object. Hence, the proposed radar object detection systems processes a real radar target list, and therein identifies multiple object hypotheses of actually existing objects in form of classified and orientated 2D bounding boxes.

The presented radar object detection system is evaluated exclusively on real radar data. In total, the evaluation analyzes object detection results on three different radar datasets, where two of them are publicly available. Another dataset is self-generated, that means radar data is recorded and annotated accordingly. The prepared radar target list for the so-called *Radar Dataset* contains high-resolution radar data from various real traffic situations in rural as well as urban regions. In order to ensure that the object detection results may be reproduced by anyone, further evaluations are performed on the two public datasets, the Astyx HiRes2019 dataset and the nuScenes dataset. The nuScenes data are even generated from the same sensors which are used for the Radar Dataset. However, since the configuration of the used sensors are not advantageous, the sensors provide only half as much radar targets. The radar object detection system performs really well on all datasets which proves that this technique is suitable for operation in real-world traffic. Eventually, the object detector shows the best performance on radar data of the Radar Dataset. Although, the radar target lists are sparse, the configuration of the sensors ensures to generate enough reflections per object for an improved detection. In fact, the proposed radar object detection system is so effective that it is successfully deployed on a real system for real-time environment modeling while driving in road traffic.

In order to provide a robust environment perception over time, a multi-object tracking system combines the output of three identical radar sensors, on which the presented radar object detector is operated simultaneously. The utilized radar sensors have an overlapping Field of View (FOV) for the most part in the near range. Still, this does not imply that the radar detecting systems provide same detections for the same object. In order to combine various object detections, which are sometimes even contradictory, this thesis realizes a multi-object tracking using a Labeled Multi-Bernoulli (LMB) filter. For this purpose, the thesis describes a measurement model to process the outputs of all deployed radar object detection systems in parallel. The obtained results of the multi-object tracking system demonstrates that the proposed radar object detection system enables a recognition of objects in the surroundings based on high-resolution radar data. The tracking algorithm not only stabilizes the results of the radar object detectors, but ultimately, provides a reliable environment perception at an object level. This object detection technique performs well on high-resolution radar data from real traffic situations and is successfully integrated as an essential part in an environment perception setup applied for automated driving.

6.2 Future Work

Although the proposed radar object detection system produces impressive results, it has the capability for further improvements. This section outlines two possible enhancements to make the detection algorithm more powerful. Actually, the output of the radar detector is one single hypothesis for each object in form of a 2D bounding box. Such an object detection comprises an estimate for position, orientation and extent of the object as well as a classification information. A radar target contains a radial velocity information, which is an essential factor for the estimation of an object hypothesis, particularly when estimating the orientation. Since an object usually generates several radar reflections, and the radial velocities differ slightly depending on the position, in principle it, is feasible to predict a velocity for the object. The regression network to determine bounding box parameters is basically suitable to generate an estimate for the velocity as well. Then, the output of the radar object detection pipeline must be modified in such a way that a bounding box comprises another state, specifically that of an object's velocity. In the context of the thesis, it is not intended to estimate a velocity for an object of interest, because the self-generated Radar Dataset does not include any reliable values for this component. In general, it is possible to annotate the object velocity, but this requires successive radar target lists. Since the time intervals of labeled radar data from the three sensors employed for the Radar Dataset are not uniform and mostly with a longer time span, no reliable object velocities can be obtained. Even if this thesis does not consider the object velocity, this does not imply that it is not possible to directly estimate a velocity for an object based on radar data using the presented approach.

For object detection in radar data using the proposed technique, it is essential to initially generate patches from the input radar target list. Next, multiple networks identify a maximum of exactly one object, which is located in the center of a patch, for each partial target list. Eventually, the last unit in the pipeline combines the intermediate results from all patches into one object detection result for the complete radar target list. This principle performs excellently and is absolutely practical for real-world application. But, another approach is to identify regions of interest internally using a neuronal network. That implies that the network is initially able to find objects anywhere in the radar target list, and subsequently predicts their classification information and bounding boxes. The *Deep Hough Voting* proposed in [QLHG19] is a method for that purpose when processing 3D light detection and ranging (lidar) data. Therefore, a VoteNet identifies the center of objects by voting individual points of the input data. Then, a sampling process groups and aggregates all votes to obtain clusters with potential objects. These are comparable to regions of interest that are classified as object patches. The last module predicts the class and the bounding box parameters for a 3D object. Similar to PointNet and PointNet++, the methodology of VoteNet is expected to be applicable to radar targets as well.

In conclusion, this thesis proposes a novel radar object detection system which constitutes a fully functional technology for object detection in high-resolution radar data in real-time, and that exhibits the capability for further research and development.

Acronyms

δ-GLMB	δ-Generalized Labeled Multi-Bernoulli	34
ADAS	Advanced Driver Assistance System	1, 37
ADMA	Automotive Dynamic Motion Analyzer	78
AP	average precision	99, 102
CAN	Controller Area Network	76
CFAR	Constant False Alarm Rate	9
CNN	Convolutional Neural Network	20, 55
CPU	Central Processing Unit	68
CS	Chirp Sequence	8
CTRV	Constant Turn Rate and Velocity	73
DBSCAN	Density-Based Spatial Clustering of Applications with Noise	15
DGPS	Differential Global Positioning System	78
DNN	Deep Neuronal Network	21
EKF	Extended Kalman Filter	34
FC	Fully Connected	56
FMCW	Frequency Modulated Continuous Wave	8, 76
FN	false negatives	100
FOV	Field of View	32, 39, 134
FP	feature propagation	60
FP	false positives	71
FSK	Frequency Shift Keying	7
GM	Gaussian Mixture	35, 73
GPU	Graphics Processing Unit	13, 42
IMU	Inertial Measurement Unit	78

IoU	Intersection over Union 53
JPDA	Joint Probabilistic Data Association 18
lidar	light detection and ranging 1, 6, 37, 135
LMB	Labeled Multi-Bernoulli 34, 72, 134
LSTM	Long Short-Term Memory 22
MFSK	Multi Frequency Shift Keying 8
MHT	Multiple Hypotheses Tracking 33
mIoU	mean IoU 101
MLP	Multi-Layer Perceptron 26, 56
MRG	Multi-Resolution Grouping 28
MSG	Multi-Scale Grouping 28, 59
NMS	Non-Maximum Suppression 33, 51
OBB	Orientated Bounding Box 15
radar	radio detection and ranging ..... i, iii, v, vii–xii, 1, 5, 37, 133
RAM	Random Access Memory 97
RANSAC	Random Sample Consensus 16
RCS	Radar Cross Section 6, 39
RFS	Random Finite Set 33
SA	Set Abstraction 59
SSG	Single Scale Grouping 28, 59
SVM	Support Vector Machine 18
T-Net	Transformer PointNet 26, 49, 133
TN	true negatives 100
TP	true positives 71
UKF	Unscented Kalman Filter 34

Symbols

Notation

x	scalar
$\underline{x}$	vector
$\underline{x}^{\mathrm{T}}$	transpose of a vector
$\mathcal{M}$	set with elements
$\mathbb{N}$	natural numbers
$\mathbb{R}$	real numbers
$\mathbb{R}^n$	n-dimensional real space

Radar Object Detection

x	radar target x position
y	radar target y position
v_r	radar target radial velocity
$\tilde{v}_r$	ego motion compensated radial velocity
σ	radar target RCS value
$\mathcal{C}_{cls}$	set with class mapping for classification
c	class for classification
$\hat{c}$	predicted class for classification
$\mathcal{C}_{seg}$	set with class mapping for segmentation
o	class for segmentation
$\hat{o}$	predicted class for segmentation
$\underline{z}$	classification scores
x_c	object box x center position
y_c	object box y center position
θ	object box heading angle
l	object box length
w	object box width

Multi-Object Tracking

x	track x position
y	track y position
θ	track orientation
v	track velocity
l	track length
w	track width
ω	track yaw rate
$\underline{x}$	track state
$\underline{z}$	measurement vector
$\underline{\hat{x}}$	estimated track state
k	time step k
$k+1$	next time step $k+1$

Bibliography

[Bal81] Ballard, Dana H.: *Generalizing the Hough Transform to Detect Arbitrary Shapes*. In: *Pattern Recognition*, volume 13, no. 2, pages 111–122, 1981.

[BF88] Bar-Shalom, Yaakov and Fortmann, Thomas: *Tracking and Data Association*. Academic Press, Inc., 1988.

[Bis06] Bishop, Christopher M.: *Pattern Recognition and Machine Learning*. Springer, 2006.

[BP99] Blackman, Samuel S. and Popoli, Robert F.: *Design and Analysis of Modern Tracking Systems*. Artech House Publishers, 1999.

[Bre01] Breiman, Leo: *Random Forests*. In: *Machine Learning*, volume 45, no. 6, pages 5–32, 2001.

[BRG+16] Beard, Michael; Reuter, Stephan; Granström, Karl; et al.: *Multiple Extended Target Tracking With Labeled Random Finite Sets*. In: *IEEE Transactions on Signal Processing*, volume 64, no. 7, pages 1638–1653, 2016.

[BWT11] Bar-Shalom, Yaakov; Willett, Peter K.; and Tian, Xin: *Tracking and Data Fusion*. YBS Publishing, 2011.

[Car97] Caruana, Rich: *Multitask Learning*. In: *Machine Language*, volume 28, no. 1, pages 41–75, 1997.

[Cau47] Cauchy, Augustin-Louis: *Méthode Générale pour la Résolution des Systèmes d'Équations Simultanées*. In: *Rendu des Séances de l'Académie des Sciences*, pages 536–538, 1847.

[CBL+19] Caesar, Holger; Bankiti, Varun; Lang, Alex H.; et al.: *nuScenes: A Multimodal Dataset for Autonomous Driving*. arXiv. 2019.

[Con] Continental Engineering Services GmbH: *ARS 408*. `https://conti-engineering.com/components/ars-408/`. Accessed: 2021-01-21.

[Cop16] Copeland, Michael: *What's the Difference Between Artificial Intelligence, Machine Learning and Deep Learning?* `https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/`. Accessed: 2020-08-22. 2016.

[CRSW21] Chamseddine, Mahdi; Rambach, Jason; Stricker, Didier; and Wasenmuller, Oliver: *Ghost Target Detection in 3D Radar Data using Point Cloud based Deep Neural Network*. In: *25th International Conference on Pattern Recognition (ICPR)*, pages 10398–10403, 2021.

[CT65] Cooley, James W. and Tukey, John W.: *An Algorithm for the Machine Calculation of Complex Fourier Series*. In: *Mathematics of Computation*, volume 19, pages 297–301, 1965.

[CV95] Cortes, Corinna and Vapnik, Vladimir: *Support-Vector Networks*. In: *Machine Learning*, volume 20, no. 3, pages 273–297, 1995.

[DEBK20] Dreher, Maria; Erçelik, Emeç; Bänziger, Timo; and Knol, Alois: *Radar-based 2D Car Detection Using Deep Neural Networks*. In: *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1–8, 2020.

[DGBD19] Danzer, Andreas; Griebel, Thomas; Bach, Martin; and Dietmayer, Klaus: *2D Car Detection in Radar Data with PointNets*. In: *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 61–66, 2019.

[DGD18] Danzer, Andreas; Gies, Fabian; and Dietmayer, Klaus: *Multi-Object Tracking with Interacting Vehicles and Road Map Information*. In: *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 589–595, 2018.

[DHS+14] Dubé, Renaud; Hahn, Markus; Schütz, Markus; Dickmann, Jürgen; and Gingras, Denis: *Detection of Parked Vehicles from a Radar Based Occupancy Grid*. In: *IEEE Intelligent Vehicles Symposium (IV)*, pages 1415–1420, 2014.

[DLR77] Dempster, A. P.; Laird, N. M.; and Rubin, D. B.: *Maximum Likelihood from Incomplete Data Via the EM Algorithm*. In: *Journal of the Royal Statistical Society: Series B (Methodological)*, volume 39, no. 1, pages 1–22, 1977.

[EEG+15] Everingham, Mark; Eslami, Saeid M. A.; Gool, Luc; et al.: *The Pascal Visual Object Classes Challenge: A Retrospective*. In: *International Journal of Computer Vision*, volume 111, no. 1, pages 98–136, 2015.

[EGW+10] Everingham, Mark; Gool, Luc; Williams, Christopher K.; Winn, John; and Zisserman, Andrew: *The Pascal Visual Object Classes (VOC) Challenge*. In: *International Journal of Computer Vision*, volume 88, no. 2, pages 303–338, 2010.

[EKSX96] Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; and Xu, Xiaowei: *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*. In: *2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 226–231, 1996.

[Far17] Faruque, Saleh (editor): *Radio Frequency Modulation Made Easy*. From the series *SpringerBriefs in Electrical and Computer Engineering*. 1st editon, Springer International Publishing, 2017.

[FB81] Fischler, Martin A. and Bolles, Robert C.: *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*. In: *Communications of the ACM*, volume 24, no. 6, pages 381–395, 1981.

[GAH+21] Griebel, Thomas; Authaler, Dominik; Horn, Markus; et al.: *Anomaly Detection in Radar Data Using PointNets*. In: *IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 2667–2673, 2021.

[GBC16] Goodfellow, Ian; Bengio, Yoshua; and Courville, Aaron: *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[GDG+17] Goyal, Priya; Dollár, Piotr; Girshick, Ross; et al.: *Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour*. arXiv. 2017.

[Gir15] Girshick, Ross: *Fast R-CNN*. In: *IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[Ho95] Ho, Tin Kam: *Random decision forests*. In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Volume 1, pages 278–282, 1995.

[HR10] Heuel, Steffen and Rohling, Hermann: *Pedestrian Recognition based on 24 GHz Radar Sensors*. In: *11th International Radar Symposium (IRS)*, pages 1–6, 2010.

[HR11] Heuel, Steffen and Rohling, Hermann: *Two-Stage Pedestrian Classification in Automotive Radar Systems*. In: *12th International Radar Symposium (IRS)*, pages 477–484, 2011.

[HR12] Heuel, Steffen and Rohling, Hermann: *Pedestrian Classification in Automotive Radar Systems*. In: *13th International Radar Symposium (IRS)*, pages 39–44, 2012.

[HS97] Hochreiter, Sepp and Schmidhuber, Jürgen: *Long Short-term Memory*. In: *Neural Computation*, volume 9, pages 1735–80, 1997.

[Hub64] Huber, Peter J.: *Robust Estimation of a Location Parameter*. In: *The Annals of Mathematical Statistics*, volume 35, no. 1, pages 73–101, 1964.

[Jac12] Jaccard, Paul: *The Distribution of the Flora in the Alpine Zone*. In: *New Phytologist*, volume 11, no. 2, pages 37–50, 1912.

[Jaz70] Jazwinski, Andrew: *Stochastic Processes and Filtering Theory*. Volume 64. Academic Press, 1970.

[JCYK08] Jung, Ho Gi; Cho, Young Ha; Yoon, Pal Joo; and Kim, Jaihie: *Scanning Laser Radar-Based Target Position Designation for Parking Aid System*. In: *IEEE Transactions on Intelligent Transportation Systems*, volume 9, no. 3, pages 406–424, 2008.

[JU04] Julier, Simon J. and Uhlmann, Jeffrey K.: *Unscented Filtering and Nonlinear Estimation*. In: *Proceedings of the IEEE*, volume 92, no. 3, pages 401–422, 2004.

[JUD00] Julier, Simon J.; Uhlmann, Jeffrey K.; and Durrant-Whyte, Hugh F.: *A new Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators*. In: *IEEE Transactions on Automatic Control*, volume 45, no. 3, pages 477–482, 2000.

[JUD95] Julier, Simon J.; Uhlmann, Jeffrey K.; and Durrant-Whyte, Hugh F.: *A new Approach for Filtering Nonlinear Systems*. In: *Proceedings of American Control Conference*. Volume 3, pages 1628–1632, 1995.

[Kal60] Kalman, Rudolf E.: *A New Approach to Linear Filtering and Prediction Problems*. In: *ASME Journal of Basic Engineering*, volume 82, no. 1, pages 35–45, 1960.

[KB15] Kingma, Diederik P. and Ba, Jimmy: *Adam: A Method for Stochastic Optimization*. In: *International Conference on Learning Representations, ICLR*, 2015.

[KNW+15] Kunz, Felix; Nuss, Dominik; Wiest, Jürgen; et al.: *Autonomous Driving at Ulm University: A Modular, Robust, and Sensor-Independent Fusion Approach*. In: *IEEE Intelligent Vehicles Symposium (IV)*, pages 666–673, 2015.

[LB98] LeCun, Yann and Bengio, Yoshua: *Convolutional Networks for Images, Speech, and Time Series*. In: *The Handbook of Brain Theory and Neural Networks*. MIT Press, 1998, pages 255–258.

[LGG+17] Lin, Tsung-Yi; Goyal, Priya; Girshick, Ross; He, Kaiming; and Dollár, Piotr: *Focal Loss for Dense Object Detection*. In: *IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[LHDW15] Lombacher, Jakob; Hahn, Markus; Dickmann, Jürgen; and Wöhler, Christian: *Detection of Arbitrarily Rotated Parked Cars Based on Radar Sensors*. In: *16th International Radar Symposium (IRS)*, pages 180–185, 2015.

[LHDW16] Lombacher, Jakob; Hahn, Markus; Dickmann, Jürgen; and Wöhler, Christian: *Potential of Radar for Static Object Classification using Deep Learning Methods*. In: *IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pages 1–4, 2016.

[LHDW17] Lombacher, Jakob; Hahn, Markus; Dickmann, Jürgen; and Wöhler, Christian: *Object Classification in Radar Using Ensemble Methods*. In: *IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pages 87–90, 2017.

[LLH+17] Lombacher, Jakob; Laudt, Kilian; Hahn, Markus; Dickmann, Jürgen; and Wöhler, Christian: *Semantic Radar Grids*. In: *IEEE Intelligent Vehicles Symposium (IV)*, pages 1170–1175, 2017.

[LYZ+20] Liang, Ming; Yang, Bin; Zeng, Wenyuan; et al.: *PnPNet: End-to-End Perception and Prediction With Tracking in the Loop*. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11550–11559, 2020.

[MAFK17] Mousavian, Arsalan; Anguelov, Dragomir; Flynn, John; and Košecká, Jana: *3D Bounding Box Estimation Using Deep Learning and Geometry*. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5632–5640, 2017.

[Mah07] Mahler, Ronald P. S.: *Statistical Multisource-Multitarget Information Fusion*. Artech House Inc., Norwood, 2007.

[Mit97] Mitchell, Tom M.: *Machine Learning*. McGraw-Hill, 1997.

[MK19a] Meyer, Michael and Kuschk, Georg: *Automotive Radar Dataset for Deep Learning Based 3D Object Detection*. In: *16th European Radar Conference (EuRAD)*, pages 129–132, 2019.

[MK19b] Meyer, Michael and Kuschk, Georg: *Deep Learning Based 3D Object Detection for Automotive Radar and Camera*. In: *16th European Radar Conference (EuRAD)*, pages 133–136, 2019.

[Mot] Motional: *nuScenes devkit*. `https://github.com/nutonomy/nuscenes-devkit`. Accessed: 2021-01-23.

[Mun11] Munz, Michael: *Generisches Sensorfusionsframework zur gleichzeitigen Zustands- und Existenzschätzung für die Fahrzeugumfelderkennung*. PhD Thesis, Ulm University, 2011.

[Mur12] Murphy, Kevin: *Machine Learning: A Probabilistic Perspective*. Volume 58. MIT Press, 2012.

[NGW+19] Nobis, Felix; Geisslinger, Maximilian; Weber, Markus; Betz, Johannes; and Lienkamp, Markus: *A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection*. In: *Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pages 1–7, 2019.

[NQ19] Nabati, Ramin and Qi, Hairong: *RRPN: Radar Region Proposal Network for Object Detection in Autonomous Vehicles*. In: *IEEE International Conference on Image Processing (ICIP)*, pages 3093–3097, 2019.

[PDKG20] Palffy, Andras; Dong, Jiaao; Kooij, Julian F. P.; and Gavrila, Dariu M.: *CNN Based Road User Detection Using the 3D Radar Cube.* In: *IEEE Robotics and Automation Letters*, volume 5, no. 2, pages 1263–1270, 2020.

[PLC00] Park, June-Me; Looney, Carl G.; and Chen, Hui-Chuan: *Fast Connected Component Labeling Algorithm Using a Divide and Conquer Technique.* In: *Computers and Their Applications*, 2000.

[PNS20] Padilla, Rafael; Netto, Sergio L.; and da Silva, Eduardo A. B.: *A Survey on Performance Metrics for Object-Detection Algorithms.* In: *International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 237–242, 2020.

[PRM18] Plachetka, Christopher; Rieken, Jens; and Maurer, Markus: *The TUBS Road User Dataset: A New LiDAR Dataset and its Application to CNN-based Road User Classification for Automated Vehicles.* In: *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2623–2630, 2018.

[QLHG19] Qi, Charles R; Litany, Or; He, Kaiming; and Guibas, Leonidas J: *Deep Hough Voting for 3D Object Detection in Point Clouds.* In: *IEEE International Conference on Computer Vision (ICCV)*, pages 9277–9286, 2019.

[QLW+18] Qi, Charles R.; Liu, Wei; Wu, Chenxia; Su, Hao; and Guibas, Leonidas J.: *Frustum PointNets for 3D Object Detection from RGB-D Data.* In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 918–927, 2018.

[QSMG17] Qi, Charles R.; Su, Hao; Mo, Kaichun; and Guibas, Leonidas J.: *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation.* In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.

[QYSG17] Qi, Charles R.; Yi, Li; Su, Hao; and Guibas, Leonidas J.: *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space.* In: *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 5099–5108, Curran Associates, Inc., 2017.

[Red18] Redmon Josephand Farhadi, Ali: *YOLOv3: An Incremental Improvement.* arXiv. 2018.

[Rei79] Reid, Donald B.: *An Algorithm for Tracking Multiple Targets.* In: *IEEE Transactions on Automatic Control*, volume 24, no. 6, pages 843–854, 1979.

[Reu14] Reuter, Stephan: *Multi-Object Tracking Using Random Finite Sets.* PhD Thesis, Ulm University, 2014.

[RHGS17] Ren, Shaoqing; He, Kaiming; Girshick, Ross; and Sun, Jian: *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 39, no. 6, pages 1137–1149, 2017.

[Ric14] Richards, Mark A. (editor): *Fundamentals of Radar Signal Processing*. 2nd editon, McGraw-Hill, 2014.

[RKDW16] Roos, Fabian; Kellner, Dominik; Dickmann, Jürgen; and Waldschmidt, Christian: *Reliable Orientation Estimation of Vehicles in High-Resolution Radar Images*. In: *IEEE Transactions on Microwave Theory and Techniques*, volume 64, no. 9, pages 2986–2993, 2016.

[RKK+15] Roos, Fabian; Kellner, Dominik; Klappstein, Jens; et al.: *Estimation of the Orientation of Vehicles in High-Resolution Radar Images*. In: *IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pages 1–4, 2015.

[Rud16] Ruder, Sebastian: *An Overview of Gradient Descent Optimization Algorithms*. arXiv. 2016.

[RVVD14] Reuter, Stephan; Vo, Ba-Tuong; Vo, Ba-Ngu; and Dietmayer, Klaus: *The Labeled Multi-Bernoulli Filter*. In: *IEEE Transactions on Signal Processing*, volume 62, no. 12, pages 3246–3260, 2014.

[Sch19] Scheel, Alexander: *Fully Bayesian Vehicle Tracking Using Extended Object Models*. PhD Thesis, Ulm University, 2019.

[SD19] Scheel, Alexander and Dietmayer, Klaus: *Tracking Multiple Vehicles Using a Variational Radar Model*. In: *IEEE Transactions on Intelligent Transportation Systems*, volume 20, no. 10, pages 3721–3736, 2019.

[SFY21] Svenningsson, Peter; Fioranelli, Francesco; and Yarovoy, Alexander: *Radar-PointGNN: Graph Based Object Recognition for Unstructured Radar Point-cloud Data*. In: *IEEE Radar Conference (RadarConf)*, pages 1–6, 2021.

[Sha76] Shafer, Glenn: *A Mathematical Theory of Evidence*. Academic Press, 1976.

[SHDW18a] Schumann, Ole; Hahn, Markus; Dickmann, Jürgen; and Wöhler, Christian: *Semantic Segmentation on Radar Point Clouds*. In: *21st International Conference on Information Fusion (FUSION)*, pages 2179–2186, 2018.

[SHDW18b] Schumann, Ole; Hahn, Markus; Dickmann, Jürgen; and Wöhler, Christian: *Supervised Clustering for Radar Applications: On the Way to Radar Instance Segmentation*. In: *IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pages 1–4, 2018.

[SHS+21a] Schumann, Ole; Hahn, Markus; Scheiner, Nicolas; et al.: *RadarScenes: A Real-World Radar Point Cloud Data Set for Automotive Applications*. Version 1.0. Accessed: 2022-01-20. 2021.

[SHS+21b] Schumann, Ole; Hahn, Markus; Scheiner, Nicolas; et al.: *RadarScenes: A Real-World Radar Point Cloud Data Set for Automotive Applications*. arXiv. 2021.

[SKA+21] Scheiner, Nicolas; Kraus, Florian; Appenrodt, Nils; Dickmann, Jürgen; and Sick, Bernhard: *Object Detection for Automotive Radar Point Clouds – A Comparison*. In: *IEEE Intelligent Transportation Systems Magazine*, pages 2–27, 2021.

[Sko08] Skolnik, Merrill (editor): *Radar Handbook*. 3rd editon, McGraw-Hill, 2008.

[SLH+20] Schumann, Ole; Lombacher, Jakob; Hahn, Markus; Wöhler, Christian; and Dickmann, Jürgen: *Scene Understanding With Automotive Radar*. In: *IEEE Transactions on Intelligent Vehicles*, volume 5, no. 2, pages 188–203, 2020.

[SM86] Salton, Gerard and McGill, Michael J.: *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.

[Sor85] Sorenson, Harold W.: *Kalman Filtering: Theory and Application*. IEEE Press, 1985.

[SR20] Shi, Weijing and Rajkumar, Raj: *Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud*. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1708–1716, 2020.

[SRKW16] Schlichenmaier, Johannes; Roos, Fabian; Kunert, Martin; and Waldschmidt, Christian: *Adaptive Clustering for Contour Estimation of Vehicles for High-Resolution Radar*. In: *IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pages 1–4, 2016.

[SRW08] Schubert, Robin; Richter, Eric; and Wanielik, Gerd: *Comparison and Evaluation of Advanced Motion Models for Vehicle Tracking*. In: *11th International Conference on Information Fusion (FUSION)*, pages 730–735, 2008.

[SSSW17] Schlichenmaier, Johannes; Selvaraj, Niranjan; Stolz, Martin; and Waldschmidt, Christian: *Template Matching for Radar-Based Orientation and Position Estimation in Automotive Scenarios*. In: *IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pages 95–98, 2017.

[ST13] Stutzman, Warren and Thiele, Gary (editors): *Antenna Theory and Design*. 3rd editon, John Wiley & Sons, 2013.

[SWHD17] Schumann, Ole; Wöhler, Christian; Hahn, Markus; and Dickmann, Jürgen: *Comparison of Random Forest and Long Short-Term Memory Network Performances in Classification Tasks Using Radar*. In: *Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pages 1–6, 2017.

[Tec] Technische Universität Braunschweig, Institut für Regelungstechnik: *TUBS Labeling Tool*. `https://github.com/TUBSDataset/TUBS_LabelingTool`. Accessed: 2021-02-06.

[Tou83] Toussaint, Godfried: *Solving Geometric Problems with the Rotating Calipers*. In: *IEEE Mediterranean Electrotechnical Conference (MELECON)*, pages 1–8, 1983.

[TZQ21] Tang, Xiaolin; Zhang, Zhiqiang; and Qin, Yechen: *On-Road Object Detection and Tracking Based On Radar and Vision Fusion: A Review*. In: *IEEE Intelligent Transportation Systems Magazine*, pages 2–27, 2021.

[Uhl92] Uhlmann, Jeffrey K.: *Algorithms for Multiple-Target Tracking*. In: *American Scientist*, volume 80, no. 2, pages 128–141, 1992.

[Vo08] Vo, Ba-Tuong: *Random Finite Sets in Multi-Object Filtering*. PhD Thesis, University of Western Australia, 2008.

[VV13] Vo, Ba-Tuong and Vo, Ba-Ngu: *Labeled Random Finite Sets and Multi-Object Conjugate Priors*. In: *IEEE Transactions on Signal Processing*, volume 61, no. 13, pages 3460–3475, 2013.

[WHLS15] Winner, Hermann; Hakuli, Stephan; Lotz, Felix; and Singer, Christina (editors): *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. From the series *ATZ/MTZ-Fachbuch*. 3rd editon, Springer Vieweg, 2015.

[Wit19] Wittpahl, Volker (editor): *Künstliche Intelligenz*. 1st editon, Springer Vieweg, 2019.

[Wol98] Wolff, Christian: *Radargrundlagen*. `https://www.radartutorial.eu`. Accessed: 2022-01-21. 1998.

[WRK+15] Werber, Klaudius; Rapp, Matthias; Klappstein, Jens; et al.: *Automotive Radar Gridmap Representations*. In: *IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pages 1–4, 2015.

[YGCU20] Yang, Bin; Guo Runsheng and Liang, Ming; Casas, Sergio; and Urtasun, Raquel: *RadarNet: Exploiting Radar for Robust Perception of Dynamic Objects*. In: *Computer Vision - ECCV 2020*, pages 496–512, 2020.

[YVB20] Yadav, Ritu; Vierling, Axel; and Berns, Karsten: *Radar + RGB Fusion For Robust Object Detection In Autonomous Vehicle*. In: *IEEE International Conference on Image Processing (ICIP)*, pages 1986–1990, 2020.

Publications

As First Author

Danzer, Andreas; Griebel, Thomas; Bach, Martin; and Dietmayer, Klaus: *2D Car Detection in Radar Data with PointNets.* In: *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 61–66, 2019.

Danzer, Andreas; Gies, Fabian; and Dietmayer, Klaus: *Multi-Object Tracking with Interacting Vehicles and Road Map Information.* In: *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 589–595, 2018.

Danzer, Andreas; Reuter, Stephan; and Dietmayer, Klaus: *The Adaptive Labeled Multi-Bernoulli Filter.* In: *19th International Conference on Information Fusion (FUSION)*, pages 1531–1538, 2016.

Danzer, Andreas; Reuter, Stephan; and Dietmayer, Klaus: *Detection of Dependencies between Objects in a Multi-Object Tracking System.* In: *Automatisiertes und vernetztes Fahren Symposium (AAET)*, pages 31–48, ITS automotive nord e.V., 2017.

As Co-Author

Buchholz, Michael; Gies, Fabian; Danzer, Andreas; et al.: *Automation of the UNICARagil Vehicles.* In: *29th Aachen Colloquium Sustainable Mobility*, pages 1531–1560, 2020.

Gies, Fabian; Danzer, Andreas; and Dietmayer, Klaus: *Environment Perception Framework Fusing Multi-Object Tracking, Dynamic Occupancy Grid Maps and Digital Maps.* In: *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3859–3865, 2018.

Nuss, Dominik; Thom, Markus; Danzer, Andreas; and Dietmayer, Klaus: *Fusion of Laser and Monocular Camera Data in Object Grid Maps for Vehicle Environment Perception.* In: *17th International Conference on Information Fusion (FUSION)*, pages 1–8, 2014.

Reuter, Stephan; Danzer, Andreas; Stübler, Manuel; Scheel, Alexander; and Granström, Karl: *A fast implementation of the Labeled Multi-Bernoulli filter using gibbs sampling.* In: *IEEE Intelligent Vehicles Symposium (IV)*, pages 765–772, 2017.

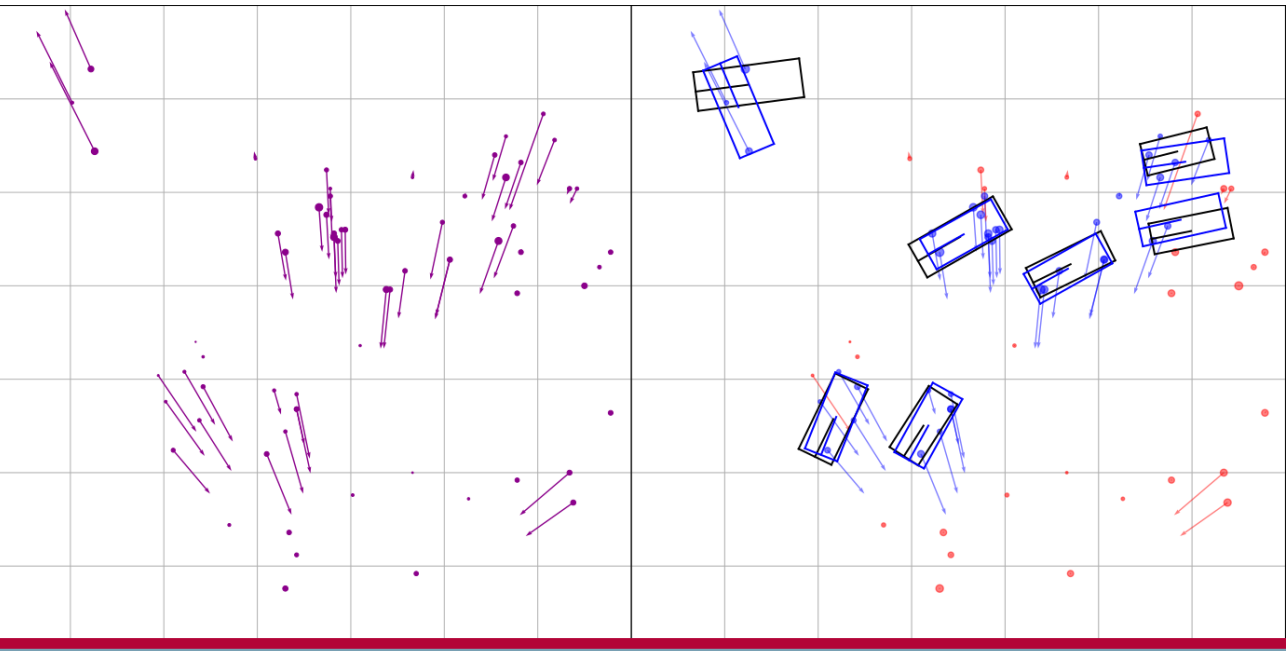
Supervised Theses

Class, Jonas: *Nutzung kartenbasierter Kontextinformation in Multi-Objekt-Tracking Verfahren*. Bachelor Thesis, Ulm University, 2017.

Griebel, Thomas: *Detection of Object Hypotheses in High-Resolution Radar Data Using Machine Learning Methods*. Master Thesis, Ulm University, 2019.

Hagmeyer, Lukas: *Detektion von Objekthypothesen in hochauflösenden Radardaten durch Box-Fitting-Methoden*. Bachelor Thesis, Ulm University of Applied Sciences, 2019.

Hermann, Charlotte: *Vorwärts-Rückwärts-Glättung von Objekttrajektorien unter Nutzung von Random-Finite-Sets*. Master Thesis, Ulm University, 2019.



ISBN 978-3-941543-68-3 (E-Book)