Radar-based road boundary estimation

Jochem Hoorneman

Master thesis report

Radar-based road boundary estimation

by

Jochem Hoorneman

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on May 20, 2021

Student number:	4231619	
Project duration:	April 1, 2019 – May 20, 2021	
Thesis committee:	Dr. ir. J.F.P. Kooij,	TU Delft, supervisor
	Dr. ir. M. Mazo Espinosa,	TU Delft
	Ir. A. Tasoglou,	2getthere

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Acknowledgement

I want to express my gratitude towards my supervisor Dr. Julian Kooij for guiding me through this project and aiding me through all the ups and downs. My gratitude also goes to Ir. Dimitrios Kotiadis and Ir. Athanasios Tasoglou of 2getthere and the company 2getthere itself for introducing me to the topic of radar-based road boundary estimation and helping me with the initial research directions. An extra thanks go to Athanasios for supervising me during my time at 2getthere. My thanks go to my friend Piotr Benedysiuk for providing feedback on my thesis report. I would like to thank my friend Jesse Mulderij for proofreading my thesis report and for being my studying buddy during these times of pandemic. And finally, I would like to thank my family for supporting me during the project.

Without all of you, this would not have been possible.

Jochem Hoorneman
Delft, April 2021

Source front page image: `https://www.uantwerpen.be/nl/onderzoeksgroep/onderzoeksgroep-stadsontwikkeling/onderzoek/infrastructuur-en-mobiliteit/highway-x-city/`

Abstract

The topic of automated driving is receiving increasing attention from the scientific community and automotive industry. A key task for an autonomous vehicle is the recognition of drivable area and, in an extension of this, detecting the road boundaries. State-of-the-art techniques often use camera and/or LIDAR sensors to perform this task. However, these sensors can be expensive and not suitable in low visibility situations such as during nighttime or in bad weather. Radar sensors are a viable alternative. The aims of this study are twofold: to introduce a technique of radar-based road boundary estimation (RBE) using radar sensors and to develop a suitable benchmark to compare different radar-based RBE techniques.

The RBE algorithm proposed in this study, Clustering-based Urban Road Boundary Estimation (CURBE), clusters the radar detections and incorporates domain knowledge to determine which of the clusters describe road boundaries (so-called boundary clusters). Additionally, we propose a novel benchmark to compare radar RBE algorithms. The benchmark makes use of the nuScenes data set. Each radar reflection in the data set is given a ground truth label. The radar reflection is given the label of boundary point or non-boundary point based on several sources such as the measured radial velocity, object annotations, and a detailed road map. To quantify the performance of an RBE algorithm, the estimated labels outputted by the RBE algorithm are compared to the ground truth labels using the metrics precision, recall, and F_1-score. CURBE is evaluated on the benchmark where it achieves an F_1-score of 30.4%.

We conclude that with the current version of CURBE and with the current method of measuring performance there are not enough grounds to consider this method as a reliable way of RBE. However, the approach shows promise. The method has the potential to significantly improve performance by using a clustering method that is designed for radar data or implementing a different way of selecting boundary clusters. Since the proposed benchmark uses the radar reflections in the data set as ground truth the performance measurement is less accurate in recordings with limited radar coverage, but overall it is concluded that the benchmark provides a fast and consistent way to measure and compare the performance of different RBE algorithms.

Contents

1 Introduction 1
1.1 Research questions . . . 3
1.2 Structure of the thesis . . . 4

2 Background information 5
2.1 Automotive radar sensors . . . 5
2.1.1 Radar basics . . . 5
2.1.2 Radar sensors compared to camera and LIDAR . . . 7

3 Related work 9
3.1 Clustering methods . . . 9
3.1.1 The DBSCAN algorithm . . . 10
3.1.2 Improvements on DBSCAN . . . 10
3.1.3 Clustering of radar data . . . 11
3.2 Road boundary estimation . . . 12
3.2.1 Radar-based RBE in general . . . 12
3.2.2 Existing works . . . 14
3.2.3 Discussion . . . 15
3.3 Automated driving data sets that include radar . . . 16
3.3.1 nuScenes data set . . . 17
3.3.2 Astyx data set . . . 18
3.3.3 Oxford RobotCar data set . . . 18
3.3.4 EU Long-term data set . . . 19
3.4 Contributions . . . 19

4 Methodology 21
4.1 The existing RBE method . . . 21
4.1.1 Approach . . . 21
4.1.2 Missing elements . . . 22
4.2 CURBE: Clustering-based Urban Road Boundary Estimation . . . 22
4.3 Static points variation of CURBE . . . 24

5 Road boundary detection benchmark 25
5.1 nuScenes data set in detail . . . 25
5.2 Ground truth labeling . . . 27
5.3 Other preprocessing steps . . . 28
5.3.1 General preprocessing: point selection and transformation . . . 28
5.3.2 Combining multiple radar sweeps . . . 28
5.3.3 Data storage . . . 29
5.4 Data set subsets . . . 29
5.5 Performance metrics . . . 29
5.6 Baseline . . . 30

6 Experiments 31
6.1 Description of experiments . . . 31
6.2 Results of hyperparameter optimization . . . 32
6.2.1 Presentation of results . . . 33
6.2.2 Discussion of results . . . 34
6.3 Results of CURBE run on the testing data partition . . . 35
6.3.1 Presentation of results . . . 35
6.3.2 Discussion of results . . . 36

6.4 Qualitative analysis of results 36

7 Conclusion and discussion 43
7.1 Answering the research questions. 43
7.2 Future work. 44

A CURBE pseudo code 47

B Radar sensor additional data 51

C Additional results 53

Bibliography 57

1 Introduction

In recent years the topic of automated driving has received much attention from the scientific community and automotive industry. A key task for an autonomous vehicle is the recognition of drivable area and, in extension of this, detecting the road boundaries. Several road boundary estimation (RBE) techniques exist, many requiring detailed maps with ego vehicle [1] localization using camera and/or LIDAR systems. However in situations where detailed maps are not available or the road situation has recently changed it is important to be able to detect the location of the road boundaries on the fly. Using a camera and/or LIDAR sensors can be a good approach to solve this task. Both sensor types provide a high-resolution representation of the environment that can be used in road boundary estimation as well as in other perception tasks of a self-driving vehicle, such as object detection. However, the sensors can be expensive and in low visibility conditions, such as nighttime, rain, or fog, the performance of these systems drops significantly. Radar sensors offer a good alternative in these situations: they do not need an external light source, maintain performance in all weather conditions, and are relatively cheap. In this thesis, we will research a radar-based RBE technique and introduce a benchmark to measure the performance of such techniques. .

To classify the degree of autonomy of a vehicle, the automotive industry uses the six levels of driving automation [48]. The levels go from zero, meaning no driving automation at all, to five, corresponding to a vehicle that can drive fully autonomous in all situations without human supervision, see Figure 1.1. As a reference: a vehicle with adaptive cruise control is classified as level one and the models from Tesla, which are often in the news for their autonomous driving features, are classified at level two and have several features of a level three vehicle as well.

There are two main approaches to reach automation level five vehicles. The first approach, taken by most traditional car manufacturers, is the gradual introduction of automated driving features in existing vehicles to enhance driver safety. These features will get more and more advanced, finally resulting in a fully self-driving vehicle.

The second approach is to design a fully autonomous vehicle from the ground up. Versions of these vehicles have been in use for several years already. These are fully autonomous, with no supervision needed from the passengers. Current systems often drive on a road separated from most other traffic and at limited speeds, but recently they are seeing applications in mixed traffic situations as well. These autonomous shuttles can be classified as automation level four. Examples of these systems in the Netherlands are the ParkShuttle in Rotterdam, see Figure 1.2, and the WEpod in Wageningen [10, 57].

The automation level four autonomous shuttles are the use-case for this thesis research. The roads that they drive on are often, but not always, surrounded by some form of separation: fences, guard rails, or bushes, which are detectable by radar.

When autonomous shuttles drive on segregated roads these are not completely isolated from other traffic. Sometimes certain types of other road users, such as bicyclists, are allowed on the same road, there can be pedestrian crossing stations and unforeseen obstacles can appear such as jaywalkers, debris, or animals. In mixed traffic situations additional road users are encountered: different types of vehicles that can be driving in the same lane or crossing the road. The ability to detect road boundaries is important for several reasons:

[1] The ego vehicle is the vehicle that contains the sensors to perceive the surroundings.

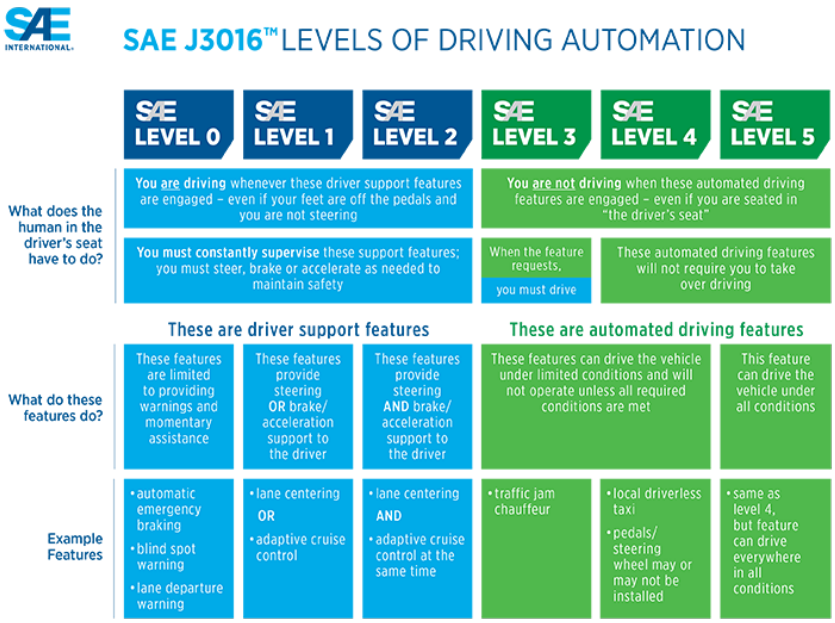


Figure 1.1: Industry standards of vehicle automation levels as defined by the international Society of Automotive Engineers (SAE international) in [48]. The road boundary estimation techniques presented in this paper focus on vehicles with automation level four and five. Source: [47].

1. it can assist in general path planning,
2. it can help determine whether detected obstacles are inside or outside the drivable area,
3. and it can assist in the planning of potential evasive maneuvers when an obstacle is detected in the driving path.

RBE solutions that use radar sensors can lead to a robust solution that works in all light and weather conditions. An extra, non-technical advantage of radar sensors, is that they are already an approved sensor in vehicles by EU law due to their widespread use in adaptive cruise control systems. This makes it so that autonomous shuttles using radar sensors can get approved for road use under current legal frameworks in EU countries without the need for new laws or one-time exceptions to be defined.

A radar senses the environment by emitting a radio signal and detecting reflections of that signal by objects in the surroundings. Different kinds of information can be extracted about the (part of an) object that

Figure 1.2: Example of an autonomous shuttle driving on a segregated road: the ParkShuttle in Rotterdam, the Netherlands.

reflected the signal, which is called a radar reflection or a radar point. The most important information is the 2D location on a horizontal plane or the 3D location in space, depending on the type of sensor, and the velocity of the radar reflection along the line of sight of the radar sensor, the radial velocity. The ego-motion compensated radial velocity can be used to determine if a radar point is static or moving. In Chapter 2 how a radar sensor works is described in more detail.

Within the field of radar-based RBE, there are several approaches as will be discussed in Section 3.2. For example, imaging radar is used in combinations with RBE techniques that are similar to vision-based methods or an occupancy grid can get constructed to use that as a base for RBE. Another approach is illustrated in Figure 1.3. Here a clustering algorithm is applied to the radar point cloud, where the expectation is that road boundary objects such as fences or guard rails will be contained in separate clusters, and the road boundary location is estimated from the clusters.

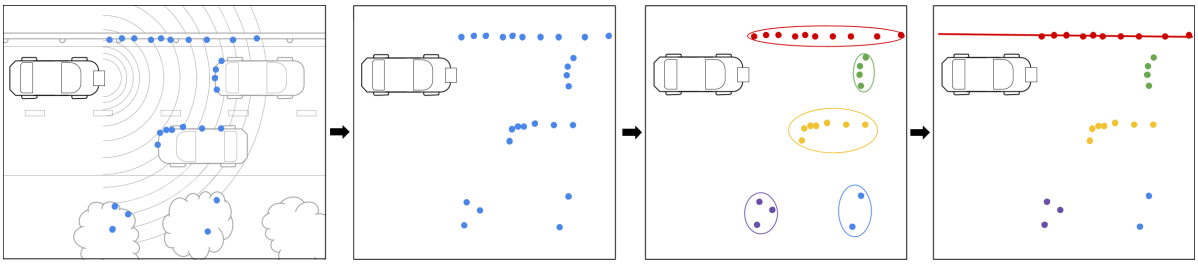


Figure 1.3: Example of clustering based road boundary estimation in a driving situation with guardrail, trees and other vehicles. Situations shown from left to right: the general driving situation, the information the radar sensor extracts, clustering of radar detections, estimating road boundary from appropriate cluster. The ego vehicle is shown in each situation as reference.

In this research, we focus on RBE for the use-case of autonomous shuttles. As an approach, we choose clustering-based RBE. For the use case, it is expected the road separators will be detectable by radar and clustering-based RBE will provide good results. This will result in a model-based approach that has the advantage of not needing any training as can be seen in machine learning approaches. In addition, the author had experience with clustering-based RBE from an internship at autonomous shuttle producer 2getthere.

1.1. Research questions

Taking the use-case and method of RBE into account, the following research questions are defined:

Is clustering-based RBE using radar data reliable enough to be used for autonomous shuttles?

Hypothesis: Since this approach tries to detect the road boundaries, as opposed to the absence of obstacles in the occupancy grid approach, these road boundaries have to be detectable by the radar sensors. The method will work well on roads with clearly defined road boundaries that are detectable by radar sensors. Straight roads or roads with only light curvature will make for more accurate road boundary estimations.

How does the existence of clear road boundary structures affect the performance of the RBE algorithm?

Hypothesis: Building on the hypothesis from the previous research question, the expectation is that more clear road boundary markers such as fences, guards rails or potentially even bushes will show a significant increase in performance.

Road boundaries are static objects and thus will have a low ego-motion compensated radial velocity. Two approaches can be used to include the ego-motion compensated radial velocity in the RBE process:

1. As an extra dimension in the clustering process, in addition to the location on the x- and y-axis. With the goal that static points not be clustered with moving points.

2. In a pre-processing step before clustering where all radar points with high ego-motion compensated radial velocity get filtered out, so only static points remain. Thereafter the radar points get clustered using the spatial coordinates.

This leads to the following research question:

Does the inclusion of ego-motion compensated radial velocity as a dimension in the clustering process lead to better RBE performance compared to filtering out non-static points before clustering?

Hypothesis: Even though including the ego-motion compensated radial velocity as an extra clustering dimension will give a more complete representation of the surroundings, it will not improve results. Since only static radar points are of interest for the final results, including non-static points will not give direct benefits while introducing an extra source of potential misdetections. For example, imagine a situation where a class of school children is walking in a row on the sidewalk. The radar sensor will detect a set of points close together approximately on a straight line, which will be clustered together. Then, in later processing steps of the RBE algorithm, it might be concluded incorrectly that this is a road boundary.

Inclusion of the ego-motion compensated radial velocity compared to ignoring it will show a significant increase in performance. However, the specific approach used to include it will have a marginal effect since both approaches should be capable of finding similar clusters of static points.

1.2. Structure of the thesis

This report is structured as follows: in Chapter 2 background information of how radar sensors work and how they compare to camera and LIDAR sensors is described. In Chapter 3 all relevant previous research papers on the topics of clustering and radar-based RBE are described. In addition publicly available automotive data sets that contain radar data are listed and the contributions of this thesis research are stated. Chapter 4 describes the RBE algorithm and its variations as used in this thesis research. Chapter 5 describes the benchmark used to evaluate the RBE algorithm. Chapter 6 describes the experiments done and discusses the results of those experiments. And finally, in Chapter 7 the conclusions are drawn and topics for future work will be suggested.

2

Background information

This chapter will describe basic information about how radar sensors work, introduce common terminology, and describe types of radar sensors often used for autonomous driving. Additionally, radar sensors are compared to the commonly used camera and LIDAR sensors.

2.1. Automotive radar sensors

Radar is a technique where objects are detected using radio waves. The term RADAR was introduced as an acronym for RAdio Detection And Ranging, but has evolved to be a common noun in English and a multitude of other languages and is therefore written without capitalization. The first types of radar sensors were developed for military use in the period before and during the second world war. Since then, it has been used extensively in, amongst others, military, aviation, and aerospace applications [46]. In the early 1970s, radar sensors were small enough to fit in the front grill of a normal vehicle and the first test vehicles using these sensors were deployed. This was for the purpose of automatic cruise control (ACC). In 1998 the first radar assisted ACC system was introduced in a production vehicle [39]. Over the years, radar sensors have improved more and more and nowadays radar sensors have become an important part of the sensor suite of autonomous vehicles [13].

2.1.1. Radar basics

A radar sensor consists of a transmitter and receiver. The transmitter emits radio waves that, such as visible light, scatter in all directions when it comes into contact with a surface. The portion of the radio wave that is reflected back to the radar receiver can be detected. Often, especially in automotive applications, the transmitter and receiver are in the same location. By comparing the emitted and received waves information about the radar reflection can be calculated. Radar can detect objects in 2D or 3D space, depending on the type of sensor. In this study only 2D radar will be considered since, as described in Section 3.3, most automotive radar data sets make use of 2D radar, including the data set that is used in this study. The four main properties a 2D radar detects are

1. the distance to the object,
2. the azimuth angle of the object,
3. the radial velocity of the object, and
4. the radar cross-section (RCS), a measure of how detectable an object is.

An example of what the output of a radar sensor can look like and commonly used terms are shown in Figure 2.1.

Material with high electric conductivity, such as most metals and wet ground, do reflect radio waves the most and are thus best detectable by radar sensors. Since radar sensors have a radio transmitter they do not depend on external illumination to detect the environment so the performance is not affected by the absence of sunlight or streetlights. In addition, the radio frequencies used for automotive radar sensors are minimally absorbed by air, and rain, fog, and snow are practically transparent to them. Thus, the relatively long range of

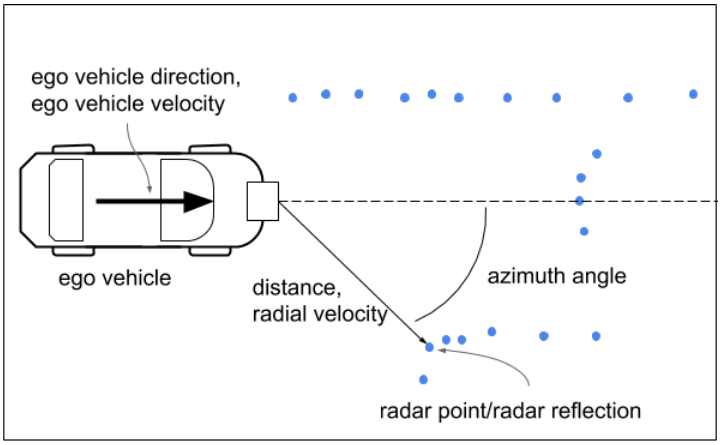


Figure 2.1: An example of what the output of a radar sensor can look like and related commonly used terms. Each radar point contains information about its location with a distance and an azimuth angle, its radial velocity and the radar cross-section (RCS) a measure of how detectable the radar point is.

radar sensors of up to 250 meters is not affected by weather conditions [31].

Earlier radar sensors sent out radio waves in pulses. Since these waves travel at approximately the speed of light the time difference between emitting and receiving of a radio wave can be used to measure the distance to objects. Nowadays in automotive radar sensors, a continuous wave is emitted [45]. To still be able to measure distance, the frequency of the radio wave is changed with time in a predictable pattern, for example, a sine or block wave. A radar sensor using both these principles is called a Frequency Modulated Continuous Wave (FMCW) radar. Due to the travel time, the reflected radio signals will be in a different phase of the frequency modulation compared to the waves that are currently emitted. With this information, the distance to the object is calculated. Another use of frequency modulation is to distinguish radio waves from different radar sensors when multiple radar sensors are present on the same vehicle. Two frequency bands are specified for automotive radar. One band at 24 GHz is used by short-range and older types of radar sensors. The ultra-wideband at this frequency will be phased out in both the EU and USA in 2022. Thus, the most recent radar sensors use a frequency band at 76 - 77 GHz.

With radar, the radial velocity of detected objects can be calculated directly using the frequencies of the emitted and received radio wave. When an object is moving the frequency of the radio wave reflected by it will change slightly: the so-called Doppler shift. This is used to calculate the velocity of the object. Only movement parallel to the direction of the radio wave causes Doppler shift, so only the velocity in that direction, the radial velocity, is calculated. The relative radial velocity Δv_{rad} between the ego vehicle and the radar point is calculated as:

$$\Delta v_{rad} = \frac{1}{2} c \frac{|f_e - f_r|}{f_e}, \tag{2.1}$$

with the speed of light c the frequency of the emitted radio wave f_e and the frequency of the received radio wave f_r. By subtracting the component of the ego vehicle velocity in the same direction the ego-motion compensated radial velocity is calculated. In the rest of the report, this ego-motion compensated velocity is meant when talking about the radial velocity of a radar point.

Before the measurements of a radar sensor are outputted for further use, various preprocessing steps are performed. From the received waves the basic information, distance, azimuth angle, radial velocity, and RCS, of each radar detection is calculated. Other preprocessing steps that can be performed are noise filtering, filtering of radar artifacts, and tracking of objects. The output that a radar sensor gives consists of the basic information of each detection and different types of metadata. This pre-processed data will be used as the starting point in this research.

Most of these quantities can be used in clustering and other RBE processes. However, it has to be noted that the RCS is not very suitable. The value of RCS can vary greatly. For example, a wet spot will often result in a much higher RCS than a nearby dry area of the same object.

The radar data returned by the sensor is in the form of a point cloud. The radar reflection is a point in N-D space, with N properties per reflection. For example distance, azimuth angle, and radial velocity. Another possible output format is a radar image. This is similar to a flash illuminated photo by a camera but then taken with radio waves instead of visible light. Radar images look like a photograph. Each pixel corresponds to the reflection intensity of the radio waves at that location.

2.1.2. Radar sensors compared to camera and LIDAR

Radar has to be compared to the two other sensor types often used for environmental perception on autonomous vehicles: camera and LIDAR sensors, to illustrate why it is a good choice as a basis for road boundary estimation. A summary of the comparison is shown in Table 2.1.

Table 2.1: Comparison of typical properties of radar, LIDAR and camera sensors used in autonomous driving applications. Sources: [5, 12, 43, 58]

Sensor type	Detects	Range	Resolution	Prize/sensor	Self-illuminating	Robust to bad weather
Radar	distance, azimuth angle, radial velocity	250m	10^2	€50 - €250	✓	✓
LIDAR	distance, azimuth angle, height	100m	10^5	€1000 - €15000	✓	X
Camera	location in 2D-projection, RGB information	100m	10^6	€50 - €500	X	X

Camera sensors have a high resolution, in the order of magnitude of 10^6 pixels per frame, compared to LIDAR and radar, and can detect colors of objects. This makes them well suited for object recognition tasks. Since most environmental perception tasks can be framed as object detection tasks, it makes the camera sensor a robust option. A lot of research has been done on all forms of object recognition using cameras since it is useful for non-automotive applications as well. Additionally, with their widespread use in other applications such as consumer electronics, the sensor type has seen a lot of development and therefore quality cameras are relatively cheap. A single camera sensor is unable of directly detecting distance to objects, while multi-camera setups require additional computations to calculate it. The effective range is constrained by pixel density. The computation time of object recognition tasks increases as pixel density goes up, introducing a fundamental limitation on the performance of camera sensors. Camera sensors require an external light source, ideally direct sunlight, and will have dramatically decreased performance at night, in heavy rain, mist, or snow.

LIDAR (LIght Detection And Ranging) is a method to detect the surroundings using a laser. The first form of a LIDAR sensor was designed in the 1960s and since then it was mainly used in meteorology, surface mapping, and archaeology. The first 3D LIDAR as is used in autonomous driving applications came in use with the 2007 DARPA challenge. A LIDAR sensor detects its surroundings by illuminating it with laser beams and detecting the reflections, in a technique similar to radar. Just as radar, a LIDAR sensor gives the azimuth angle and distance for each detected point. The resolution of a LIDAR sensor is high compared to radar: in the order of magnitude of 10^5 points are detected per frame. The sensor is not dependent on external light sources, so works well in low light conditions, but performance does decrease significantly with bad weather conditions. LIDAR sensors are relatively new and therefore relatively expensive, but prices are dropping fast: where the first automotive LIDAR sensors had a cost of around €50,000, current ones are dipping below €10,000 and this trend is expected to continue [26].

Just as camera sensors, radar has been around for a long time, so sensors have become relatively cheap. The resolution of radar is low, an order of magnitude of 10^2 detections per frame, but it has a range of up to 250m and works in all light and weather conditions.

In practice, autonomous vehicles will often have all three sensor types in various numbers in the environmental perception sensor suite to have the most complete view of the surroundings. Different sensors are best suited for different perception tasks, but there is a lot of overlap. Camera sensors will typically be used for object detection, LIDAR sensors to get a detailed 3D view of the nearby surroundings, and radar sensors for robust short and long-range obstacle detection.

3

Related work

The topic of this research, RBE using clustered radar points, is based on insights from previous related studies. Relevant findings from the literature are presented in this chapter.

Section 3.1 describes the methodology for efficient clustering of radar point clouds. First, relevant general clustering methods are described. Second, the clustering methods designed for radar point clouds specifically are discussed.

For online RBE different approaches can be used depending on the sensor type. Typical sensors on autonomous vehicles are camera, LIDAR, or radar sensors and RBE methods exist for all of them. In this review, only methods using radar data are described and can be seen in Section 3.2.

To compare different methods of RBE a suitable benchmark has to be chosen. A data set consisting of recorded sensor data from a vehicle driving in representative traffic situations is most appropriate, since it is very close to a real-world situation.

In Section 3.3 the publicly available datasets fit for this application will be compared.

3.1. Clustering methods

Clustering is a form of unsupervised machine learning where data is grouped in meaningful clusters. Considering a data set consisting of data points in N-dimensional feature space, where each data point is a combination of N features, data points close to each other can be assigned to the same cluster. Thus, data points in a cluster are similar to each other according to some metric.

When applying clustering on radar data the feature space can exist of information radar sensors detect such as distance, azimuth angle, radial velocity, and RCS. The goal is to group radar points belonging to the same object in clusters. Ideally, each object in the driving situation such as vehicles, bicycles, pedestrians, trees, and, of course, road boundary structures should be in their own cluster. Besides radar points clearly belonging to certain objects, there are often singular radar points not belonging to anything. These outliers can be misdetections, noise, or detections of a small object which are not relevant in the evaluation of the driving situations. The clustering algorithms that are considered in this research need to be able to handle those outliers.

There are hundreds of different clustering algorithms. There is not a single best clustering algorithm. It is strongly dependent on the application and, most often, clustering algorithms most fit for a certain application have to be found experimentally.

Most clustering algorithms can be categorized based on the underlying concept. A few popular clustering categories will be described here.

One is **centroid-based, or k-means, clustering**. Where k clusters with each their own centroid are specified and data points are assigned to the cluster which centroid they are closest to. In most versions of centroid-based clustering, k needs to be specified and this clustering method tends to favor similarly sized clustered. Both these properties make centroid-based clustering unfit for radar data where a beforehand unknown number of objects of various sizes exist.

Another method is **hierarchical clustering** where a hierarchy, represented as a tree structure or dendrogram, is build up. Data points connected by some metric be part of the same branch which, in turn, will

combine into a bigger branch consisting of other branches which are connected. The user can use the hierarchy to split the data in the desired way. Most hierarchical methods are not good in handling outliers, since they are often given their own branch and can affect the formation of other non-outlier branches. This makes this not ideal for clustering radar data either.

Distribution-based clustering is a method that assumes clusters can be modeled by some distribution. Data points most likely belonging to the same distribution are clustered. A well-known form of this is the Gaussian mixture model where a specified fixed number of Gaussian distributions are used to cluster the data. The parameters of the Gaussian distribution are optimized to best represent the data, then data points are assigned to the cluster whose Gaussian distribution has the highest value at that data point. When applied to radar data this method suffers from the same issue as centroid-based clustering: the number of clusters is not known beforehand. When the number of distributions is not specified, distribution-based clustering runs a high risk of overfitting. Adding more distributions will lead to a better representation of the data structure, but will inevitably lead to an overfitted model.

To cluster radar data a more robust clustering approach is preferred. The robust approach comes in the form of **density-based clustering**. This is a method where dense areas in the feature space are grouped into clusters. Data points not belonging to dense areas are often classified as outliers. A very popular version is the, in 1996 introduced, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [16]. Density-based clustering and especially DBSCAN is ideal for radar data. It can detect a beforehand unspecified number of clusters of arbitrary shape and handles outliers very well. In addition, the results of DBSCAN are mostly deterministic, removing the need to run the clustering algorithm multiple times. For those reasons, almost all clustering methods specialized for radar data are based on DBSCAN.

Because of its importance, the DBSCAN algorithm will be described first. Then common modifications of the DBSCAN algorithm will be described and finally, clustering algorithms designed for radar data will be discussed.

3.1.1. The DBSCAN algorithm

The DBSCAN algorithm has two parameters: minimum number of points n_{min} and threshold distance ε. Taking the same notation as used in [36], given a N-dimensional point in a set of K points $x_i = [x_{i,1}, x_{i,2}, \cdots, x_{i,N}]$ with $i = \{1, 2, \cdots, K\}$ the Euclidean distance between two points is calculated as

$$d(x_i, x_j) = \sqrt{\sum_{n=1}^{N} (x_{i,n} - x_{j,n})^2} \tag{3.1}$$

for $i \neq j$ and $i, j = \{1, 2, \cdots, K\}$.
If for x_i the number of points that satisfies

$$d(x_i, x_j) \leq \varepsilon \tag{3.2}$$

is greater or equal to n_{min}, x_i is a core point. Points within distance ε of a core point and who are not core points themselves are called reachable points. Points that are not a core point nor a reachable point are noise points. All core points that are within distance ε of other core points are grouped with the reachable points for all these core points into a cluster.

DBSCAN classification is illustrated in a 2D example in Figure 3.1. The minimum number of points $n_{\text{min}} = 4$ and the threshold distance ε are shown. All points in red have four or more points within a distance ε and are thus core points. The points in blue are within a distance ε of a core point but are not core points themselves, so they are reachable points. The yellow points are too far away from the core points and are thus noise points. All the points in red and blue form a cluster.

3.1.2. Improvements on DBSCAN

Besides its various beneficial properties, DBSCAN also has some disadvantages. There is a high computational burden since the distance for each point to each other point has to be calculated. Leading to a quadratically increasing computation time with an increasing number of points. Secondly, the way the algorithm works is strongly dependent on the search radius parameter and point number threshold. This requires the parameters to be tuned for each specific application.

Since the introduction of DBSCAN, many improvements have been suggested. The overview paper Kumar and Sivasathya [29] contains a detailed description of the most important variations that tackle different

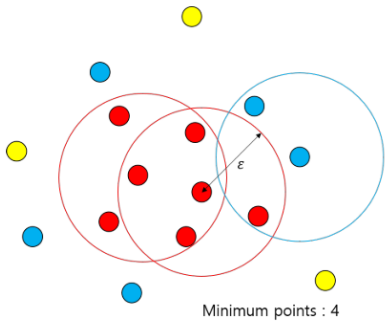


Figure 3.1: The principle of the DBSCAN algorithm, with parameters $n_{\min}$ as the minimum number of points and threshold radius ε. Core points are in red, reachable points in blue and noise points in yellow. The threshold radius is shown for three of the points. Source: [36].

issues of DBSCAN or are specialized and optimized for specific applications. This thesis will focus on the variations specifically designed for radar data as shown in Section 3.1.3, so most variations for general application will not be described here. Three notable studies are described next.

A famous variation on DBSCAN is OPTICS [2] which is a generalization of DBSCAN that makes it a hierarchical clustering method. Objects in the database get ordered such that spatially closest points become neighbors in the ordering. Clusters show up as valleys in the reachability plot. It does not need a search radius parameter. Instead, a distance threshold can be specified after the calculation of the algorithm and this will result in a different distribution of clusters. A downside of OPTICS is that is about 1.6 times slower compared to DBSCAN [2]. Since OPTICS several other versions of hierarchical DBSCAN have been introduced.

A remarkable paper was published in 2015 in Gan and Tao [17] that heavily criticizes the claimed running time of DBSCAN in the original paper Ester et al. [16]. They state that the running time of the original paper was misclaimed to be $O(n \log n)$, but requires $O(n^2)$ time instead. A variation of DBSCAN was proposed that runs in $O(n)$ time and the authors argued that their variation should become the new standard for density-based clustering from then on.

This led the authors of the paper that introduced DBSCAN [16] to publish a response in Schubert et al. [51] in 2017, twenty-one years after the publishing of the original paper. The authors discuss the criticisms raised in Gan and Tao [17] and argue that DBSCAN is still relevant. Both algorithms are executed on different examples and they show that for practical applications and with reasonable chosen parameters there is not much difference between the methods. The proposed improvements in Gan and Tao [17] “do not appear to offer practical benefits if the DBSCAN parameters are well-chosen and thus they are primarily of theoretical interest".

3.1.3. Clustering of radar data

Radar data introduces a few complicating factors when applying clustering methods to it. Radar point clouds are more dense closer to the radar sensor: a vehicle directly in front of the ego vehicle can have tens of radar reflections, while the same vehicle 50 meters away can have only a few radar reflections. Because of this, density-based clustering algorithms will tend to classify radar points far away as noise. On the other hand noise points close to the radar sensor can be close enough to each other to be grouped into a cluster. This issue is magnified by the relatively sparse nature of radar point clouds. This also makes it harder to differentiate between noise points and reflections of actual objects. Another factor to handle is the fact that different dimensions of a radar point have different scales of their value distribution. For example, the azimuth angle, measured in radians, is generally between $-\pi$ and π, while the distance, measured in meters, is generally between 0 and 150. When not correcting for this and simply taking the Euclidean distance between radar points using those two dimensions, this distance is mostly decided by the Euclidean distance and the azimuth angle has limited effect. A few clustering methods are designed specifically for radar data and will be described next.

In Kellner et al. [24] a clustering algorithm based on DBSCAN is proposed that deals with the non-equidistant sampling density amongst radar point dimensions and noise points close to the radar sensor that can be

grouped into clusters. The detection space of the radar sensor is separated into grids along the azimuth angle and range dimension to speed up the later distance calculations. The search radius parameter and point number threshold of the DBSCAN algorithm are both adaptive and do not need to be specified beforehand. They depend on the ratio between radial and angular distance for each cell. It is a radar sensor-specific ratio that can be calculated in advance. The search radius is only adjusted in the direction of the azimuth angle since the resolution in that direction is not constant with increasing distance. Using the adaptive search radius parameter and point number threshold, radar reflection of the same object closeby or far away will be clustered similarly. Since the distance criterion is dimensionless, the method can be enhanced by including the radial velocity and RCS information of each radar point.

Similar to the previously described work, in Li et al. [33] another grid-based DBSCAN algorithm is introduced. The method has similar goals to make a clustering algorithm that is optimized for radar data and is more robust by removing the need for the user-specified parameters. This method focuses specifically on a 3D grid of range, velocity, and azimuth angle.

The grid size in the range and radial velocity dimensions are determined by the bin size of a fast Fourier transform that is computed. For the azimuth angle dimension, it is determined by the common difference. By using polar coordinates instead of Cartesian at most one point gets assigned per grid cell. A window is drawn around each cell that increases with a bigger range. When the number of points in the window is smaller than a threshold it is set as a noise point, otherwise, it is a core point, and neighbors are assigned seed points that will be evaluated next.

In addition, clusters are tracked between frames to improve performance. The center point and average, minimum and maximum velocity of the cluster points are stored and used to estimate the location of the cluster in the next frame. The clusters of dynamic objects are classified as pedestrians, cyclists, or vehicles. Based on that classification a contour of fixed size is used to search for the cluster in the next frame. The method is validated by applying it to real-world radar data where stable performance is shown.

In Schumann et al. [52] a different approach is used. In this work, a form of a supervised clustering method for radar data is proposed. Learned behavior about the data gained from a supervised machine learning algorithm is combined with an unsupervised clustering algorithm. Examples of clusters of different objects (vehicles, bicycles, pedestrians) are analyzed and used to enhance the clustering algorithm.

In Scheiner et al. [49] a multi-stage clustering framework is proposed to better detect road users using radar data. In the first step, static background noise is filtered from the radar data. In the filtering criterion, the radial velocity and number of neighbor points are taken into account. The remaining points are then clustered using an adjusted version of the DBSCAN algorithm that requires fewer neighbor points for more remote detections. The x coordinate, y coordinate, and radial velocity are combined in a single Euclidean distance criterion. After this initial clustering additional information is extracted from the cluster. Using the approach in Kellner et al. [25] the real velocity of the cluster can be estimated. The cluster centers can be tracked over time and a smooth trajectory can be fitted to predict where the cluster will be in future time steps. This information is used in a second clustering step that only takes previously clustered points into account.

3.2. Road boundary estimation

In this section, RBE methods are investigated. For online RBE different approaches can be used depending on the sensor type. The typical sensors on autonomous vehicles: camera, LIDAR, or radar sensors all methods for RBE. In this review, only methods using radar data are investigated. Since 2010, eleven papers have been published that discuss radar-based RBE. First, a general overview on radar-based RBE is given and five frequently occurring techniques are described. Second, the specific approaches of the eleven papers are introduced. Last, the eleven papers are discussed and general observations are shown.

3.2.1. Radar-based RBE in general

The goal of radar-based RBE is to go from radar sensor measurements to a mathematical road model. This model is most often in the form of two curves for the left and right road boundary. Sometimes other representations are being used. For example, in Giese et al. [20] the future vehicle trajectory is represented with a single curve. As illustrated in Figure 3.2, we can see the top-level layout of a radar-based RBE algorithm as follows: data from current and potentially previous radar measurements is transformed, then the road model is fitted to the radar data which, finally, results in a mathematical representation of the road. An exception is Lee et al. [30], where the radar data transformation step is skipped and the road model is fitted directly to the

radar data.

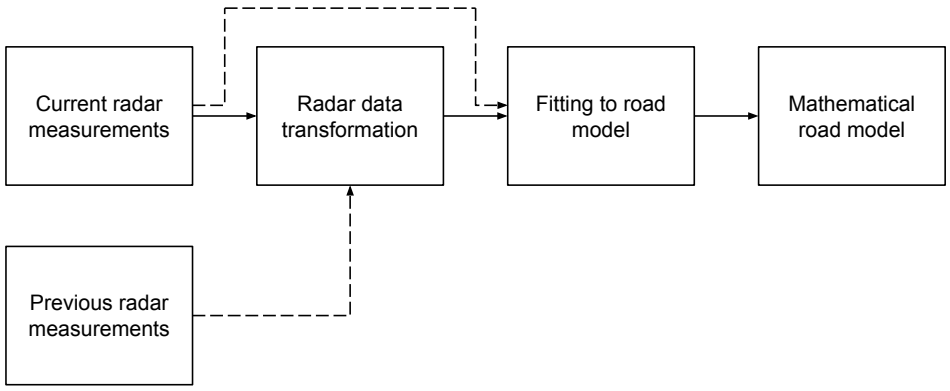


Figure 3.2: The top level layout of online radar based RBE. Data from current and potentially previous radar measurements is transformed, then the road model is fitted to the radar data which, finally, results in a mathematical representation of the road. An exception is Lee et al. [30], where the radar data transformation step is skipped and the road model is fitted directly to the radar data.

The existing works have varied approaches to radar-based RBE. Most actions of interest occur in the radar data transformation step. Five techniques that aid in that step occur in multiple papers:

- **Kalman filtering** is an algorithm to reduce the noise in a series of measurements. Not all values in the series need to be known beforehand, which makes it especially suitable for online applications. Kalman filtering can detect underlying trends in data and can also be used to predict future values in the series. Because of that it is often used in the tracking of objects in space (for example pedestrians or road boundaries) and predicting future locations [9, 37].

- The **Hough transform** is a feature extraction technique that is often used in the field of computer vision. The technique was introduced in 1972 in Duda and Hart [14] as a method to detect lines or curves in an image. Since then, countless variants have been designed to detect a wide variety of shapes. By using radar images instead of point clouds the Hough transform can be applied to radar data as well [7].

- **Occupancy grid** maps were first introduced in 1987 in Elfes [15] as a means of sonar-based mapping and navigation. Applied to autonomous vehicles, the underlying concept is to represent the local area surrounding the ego vehicle as a 2D plane in a top-down view that is divided into equally sized grid cells. The surrounding area is cropped to a $w \times h$ region of interest. This area is divided into a $M \times N$ grid, with each grid cell having a width of w/M and a length of h/N. Each cell holds the probability of it being occupied, which is often calculated by using Bayesian prediction with the sensor readings in that cell. To reduce the complexity of computing the probabilities, the probability of each cell's occupancy is assumed to be independently distributed with regard to all other cells. After the probabilities for each cell are calculated, the occupancy grid can be kept as such or be converted to an occupied/non-occupied M by N binary image and used as the input for further processing steps.

- **Clustering** is a technique to group data in meaningful clusters and is extensively described in Section 3.1. In the use case of RBE, the radar point cloud can be clustered in order to detect objects, or the road boundary estimations themselves can get clustered to simplify the output of the RBE algorithm.

- An **artificial neural network** (ANN) is a form of machine learning that is based on the biological brain. ANNs consist of sets of artificial neurons that are organized in layers. Each neuron gets a set of input values that it combines into a single output value by taking the weighted sum and using an (often) nonlinear activation function. This output value then feeds into the next layer of neurons. By giving an ANN large amounts of example data in combination with the desired outcomes, the ANN can learn by adjusting its weights how to map this input data to the desired outcome (for example a road boundary) [55].

3.2.2. Existing works

In the following sections, the existing radar-based RBE methods are described in chronological order.

In Meis et al. [40] a method that estimates the road boundaries using a Kalman filter is introduced. Tangential structures are used as measurements in the filter model. Their method is sensor independent: the authors show it to work on both radar and camera data.

The first application of an occupancy grid to radar-based RBE was introduced in Schreier and Willert [50]. From the radar data, the occupancy grid map is constructed that is used as the input for a morphological image analysis step. Four methods are used: morphological opening, morphological gradient, H-minima transformation, and watershed transformation as described in Soille [53]. The result of these four methods is so-called free-space segments. The actual free space is estimated by a two-dimensional B-spline closed free space contour. The boundaries of the free space segments serve as the virtual measurements for a Kalman filter that estimates the control points of the B-spline. Results are shown in two real-world examples where the free space estimation is overlayed on an image of the front-facing camera on the ego vehicle. In the two presented examples, the algorithm is shown to correctly estimate the free space.

In Guo et al. [22], data from an imaging radar is transformed using the Stripe Hough Transform (SHT), which is an adjusted version of the Hough transform for radar images that extracts linear features. Qualitative results are shown on artificial data consisting of several cases of two straight road boundaries with different levels of distortion.

Another paper from the same year uses a similar approach. In Zhang et al. [62] a curve Hough transform is used to extract features from imaging radar data. Distance and velocity are decoupled by target region image matching with a total variation minimization method. The method in action is shown on a single frame of recorded data on a straight road with guard rails.

In Kim and Song [27] the radar sensor had inaccuracies in the lateral distance. This was compensated using a probabilistic data association filter (PDAF). Two zones of interest are defined on the left and right sides of the ego vehicle. Both zones are split into a closeby and a far away zone. Targets in the far away zone are detected by the radar sensor. Once a stationary target enters the closeby zone it is out of detection range and the road boundary is estimated using discrete Kalman filters. This results in a set of boundary estimations, which are then clustered to decide if they belong to the left or right road boundary. Qualitative and quantitative results are shown on several real-world recordings. Performance is compared to RBE by a LIDAR-vision-based algorithm.

In Giese et al. [20] the occupancy grid is used as a base for a completely different approach compared to Schreier and Willert [50]: it is treated as an image and used as input for a convolutional neural network (CNN), that uses a variant of the Alexnet architecture as proposed in Krizhevsky et al. [28]. This CNN is used to estimate the future trajectory of the vehicle, which can be seen as a line representation of the road. The trajectory is modeled as a polygon model with a fixed number of nodes which are all equidistant. Qualitative results are shown in two situations where the algorithm performs well and two problem situations. Quantitative results show that the algorithm is able to correctly estimate the trajectory up to 100 meters on average.

In Lee et al. [30] an approach is introduced that uses circle fitting to estimate road boundaries for situations with curved roads. Moving objects are filtered out before applying the algorithm. The experiments are performed on highways with detectable guardrails. Their approach shows promise but has inadequate performance in the current state. The authors state that the development of the method will continue and improvements will be presented in future works.

In Li et al. [32, 34] the occupancy grid map from three radar sensors is merged to achieve an extended field of view. The merged occupancy grid map is converted to a binary map. The occupied grid cells are clustered using the Connected Component Labelling algorithm [35]. Small clusters with only a few grid cells are considered outliers and ignored in further steps. Boundaries of the occupied cell clusters are identified using the Moore-Neighbour Tracing algorithm. Using the Constant Turn Rate and Acceleration model [54] the future trajectory is estimated. Finally, for each position along the trajectory, the free space is defined as the set of grid cells on the shortest distance line between that position and the boundaries of the occupied cell clusters. Examples of the method in action are shown in two situations of own recorded data. No quantitative or qualitative analysis of results is included.

In Lim et al. [36], radar points are clustered using DBSCAN. These clusters are then directly used to fit a B-spline to estimate the road curvature. Results are shown on a bridge with road barriers which are very well detectable by radar. In that example, the method is shown to correctly estimate the road boundaries.

In Xu et al. [59] static and dynamic obstacles are separated first. Then, the occupancy map is constructed based on the static obstacles using modified Bayesian prediction. In the modification of the Bayesian predic-

tion, multiple frames are included where the most recent frame has the most influence. Probabilities from previous frames are given a weight that is exponentially smaller the further in the past the frame is. The road boundaries are estimated as a straight line by using a modified Random Sample Consensus (RANSAC) algorithm, that is applied to two regions of interest of the occupancy grid: one for the left and one right road boundary. Dynamic obstacles are clustered using DBSCAN in an independent algorithm. The method is applied to 100 selected frames of their own radar recordings and they report a F_1-score, recall, and precision of 79.67%, 77.56%, and 81.89% respectively. The method to define these quantitative results is not specified.

In Aihara and Fujimoto [1] a CNN is used to estimate the free space. LIDAR data is used to train the CNN on LIDAR data before it gets used to estimate free space using radar data. Because of the difference in detected points between LIDAR and radar sensors the LIDAR data gets thinned out first. Because a 3D LIDAR sensor is used and a 2D radar sensor, only LIDAR detections at a certain height are taken into account. An occupancy grid gets constructed as input for the CNN. Quantitative results are shown on a small data set and show mediocre results. The authors state to continue work on the method.

3.2.3. Discussion

The described studies show different approaches to the RBE problem. Studies differ in what form the radar data is used, in how the radar data is prepared, and how a road boundary is represented.

Table 3.1: Occurrence of common techniques used in the transformation of radar data in each of the papers. The study Lee et al. [30] solves the RBE problem by applying circle fitting directly to the radar data and uses none of the five techniques shown in this table.

Paper		Kalman filter	Hough transform	Occupancy grid	Clustering	Neural network
Meis 2010	[40]	✓				
Schreier 2012	[50]	✓		✓		
Guo 2015	[22]		✓			
Zhang 2015	[62]		✓			
Kim 2016	[27]	✓			✓	
Giese 2017	[20]			✓		✓
Lee 2018	[30]					
Li 2018	[32]			✓	✓	
Lim 2018	[36]				✓	
Xu 2019	[59]			✓	✓	
Aihara 2019	[1]			✓		✓

The papers that make use of the Hough transform [22, 62] use radar images as input, all the other works use the radar point cloud. This is explained by the origination of the Hough transform as a computer vision-based technique. By using radar images, the Hough transform can be used without the need for extensive alterations.

The usage of the five frequently-appearing techniques that were described in Section 3.2.1 is shown in Table 3.1. Kalman filtering and the Hough transform are more popular in the earlier works, whereas the more recent studies all use combinations of occupancy grids, clustering, and neural networks. The study Lee et al. [30] solves the RBE problem by applying circle fitting directly to the radar data and uses none of the five techniques shown in the table.

The occupation grid is the most commonly used technique. Studies that make use of occupation grids construct the grid as one of the first steps in the RBE algorithm. Where a radar point cloud is irregular, sparsely populated, and all points have a continuous coordinate location, the occupation grid gives a discrete representation of space that can be traversed by using indexes. This makes it significantly faster to lookup values. Together with the fact that the occupation grid can be seen as a low-resolution image, this opens up the possibility to use a wide variety of existing processing techniques to further improve the performance of the RBE algorithm.

In most studies, clustering is used to assist other methods. For example, in Kim and Song [27] estimated road boundaries are clustered to separate the left and right boundaries and in Xu et al. [59] only dynamic objects are clustered so they can get tracked over multiple frames. Only in Lim et al. [36] clustering is used as a stand-alone technique. There, the road boundaries are estimated directly from a clustered radar point cloud.

For the mathematical description of a road boundary, different approaches are in use. B-splines have been used for a long time to approximate road geometry [23], but are relatively complex to be fitted to low-resolution data such as radar point clouds. For that reason we see the road geometry modeled by more simple polynomials.

Results are presented in different ways in the described RBE studies, as illustrated in Table 3.2. Almost all studies have qualitative results and half of them include quantitative results as well. In Guo et al. [22] artificial data is used to show results, the other studies use real-world data. This data, however, is not made publicly available by any of the studies. There is no benchmark in use for radar-based RBE at the moment, which makes it hard to compare results. Although not ideal, quantitative results can be compared with each other even if the underlying data is not the same. This too, is not easy to do since each paper uses a different metric to quantify performance.

Table 3.2: Presentation of the results in each of the radar based RBE papers. It is illustrated whether qualitative or quantitative is shown if the data is artificial or real world data. Although almost all of the works use real world data, none of the data is publicly available.

Paper		Qualitative	Quantitative	Real data	Artificial data
Meis 2010	[40]	✓	✓	✓	
Schreier 2012	[50]	✓		✓	
Guo 2015	[22]	✓			✓
Zhang 2015	[62]	✓		✓	
Kim 2016	[27]	✓	✓	✓	
Giese 2017	[20]	✓	✓	✓	
Lee 2018	[30]	✓	✓	✓	
Li 2018	[32]	✓		✓	
Lim 2018	[36]	✓		✓	
Xu 2019	[59]	✓	✓	✓	
Aihara 2019	[1]		✓	✓	

3.3. Automated driving data sets that include radar

Until recently, publicly available data sets focused mainly on camera data and sometimes LIDAR data. Examples are the influential KITTI vision benchmark [18], the KITTI data set [19], the Oxford RobotCar data set [38] and the Berkeley Deep Drive data set [61]. The 2019 overview paper Guo et al. [21], which describes forty-five different automotive data sets, contains only one with radar data, but a few more have come out recently.

To find a suitable data set the demands and wishes should be clear. A data set should at least be publicly available and contain radar data. Other than that, several properties are beneficial to have but not essential.

- Information of the state of the ego vehicle: location, orientation, and velocity.
- Annotated data, so there is a ground truth that can be used to validate the performance of the used RBE algorithms.
- Detailed information of the road layout, especially the road boundaries.
- Existence of road boundaries that are clearly detectable by radar such as guard rails or fences.
- Inclusion of camera data for the qualitative evaluation of the algorithms.
- A large data set. A bigger data set will lead to more reliable experiments.
- Variety in driving situations, for example, weather conditions, the driving environment, and presence of other road users.

Based on these criteria four driving data sets containing radar data were found. The nuScenes data set [6] is an annotated, multi-sensor data set consisting of 1000 recordings of twenty seconds each. The Astyx data set [41] is a small, annotated data set, containing high-resolution 3D radar data. The Oxford Radar RobotCar data set [3] is the radar extension of the Oxford RobotCar data set [38]. It contains data from 280 km of driving through Oxford, the UK using an imaging radar. The EU Long-term data set [60] contains several recordings of the same urban route throughout the year and at different times of the day. More detailed specifications of these data sets are shown in Table 3.3.

Table 3.3: Specifications of driving data sets which include radar data.

Name	Size (in number of radar frames)	Radar sensor	Radar type	Sensor types	Variety	Annotation
nuScenes	260,000	Continental ARS 408-21	FMCW	camera, radar, LIDAR	Different weather and illumination	3D bounding boxes
Astyx	500	Astyx 6455 HiRes	FMCW	camera, radar, LIDAR		3D bounding boxes
Oxford Radar RobotCar	702,000	Navtech CTS350-X	Rotating FMCW	camera, radar, LIDAR, gps/ins	Urban route in different weather conditions	None
EU Long-term	152,000	Continental ARS 308-21	FMCW	camera, radar, LIDAR, gps/ins, imu	Urban route in different weather and light conditions	None

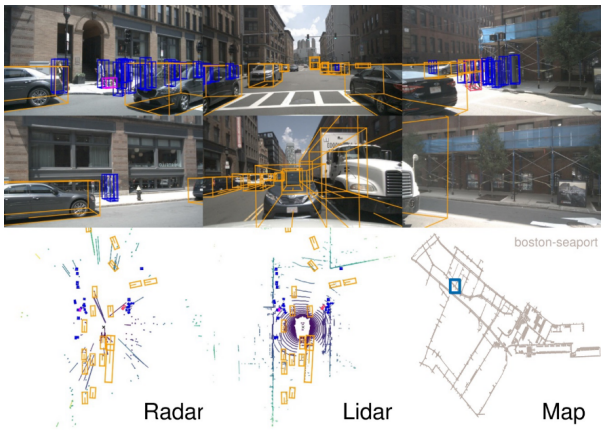


Figure 3.3: Snapshot of a scene of the nuScenes data set with camera, LIDAR and radar sensor data [6].

3.3.1. nuScenes data set

The nuScenes data set consists of 1000 twenty-second long driving scenes recorded in Boston Seaport, USA, and three districts in Singapore. The sensor suite consists of six cameras, a LIDAR sensor, five radar sensors, GPS and IMU. The five radar sensors provide a 360-degree view of every scene. The radar data is presented as a 2D point cloud. All scenes are annotated with 3D bounding boxes in twenty-three object classes, road boundaries are not annotated though. In addition, information is included to show where the road is and where it ends for all of the scenes in the form of a binary map with a resolution of 10 pixels per meter. An example of how the data and annotation look can be seen in Figure 3.3. The scenes consist of diverse situations in an urban setting: left- and right-handed driving, different weather conditions, diverse traffic conditions. Roads in Singapore have often high curbs and fences in the middle of the road, which are both good targets for detection with radar.

The nuScenes data set has a software development kit (SDK), written in Python. The SDK includes code to load point clouds from files and inspect them in various rendering situations.

Since the publication of the nuScenes data set in March 2019, a few extensions were published. In July 2019 a map expansion was released that includes annotations for different parts of the road: lanes, stop lines pedestrian crossings, etc. This could be of interest for this thesis research, but annotation seemed to focus on visual algorithms and did not include road boundary obstacles. Thus, for radar-based RBE the binary road map is deemed the most appropriate. Several other extensions providing more detailed information were published. For radar-based RBE, however, these are not of interest.

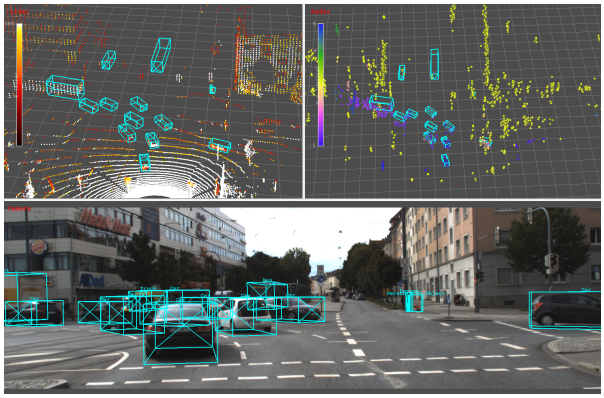


Figure 3.4: Synchronized frame of the Astyx data set with LIDAR, radar and camera data.

3.3.2. Astyx data set

The Astyx data set consists of 500 synchronized frames in suburb and highway setting at an undisclosed location. The sensor suite consists of a LIDAR, camera, and 3D radar. The Astyx radar produces around 1000 3D points per frame, whereas the radar sensors for the nuScenes and Oxford RobotCar data sets produce a few 100 2D points per frame. All sensors detect in front of the vehicle. Radar data is provided as a 3D point cloud. Vehicles and pedestrians are annotated with 3D bounding boxes. In Figure 3.4 a synchronized frame from the data set is shown. There is no variance in weather in this data set, only sunny weather with good visibility. There are sparse distinct road boundaries. The only ones available are low curbs, a guardrail, and a row of parked cars.

3.3.3. Oxford RobotCar data set

The Oxford RobotCar data set consists of thirty-two traversals of a route in the city center of Oxford, United Kingdom. The recording vehicle has a radar sensor, two 2D LIDARs, two 3D LIDARs, six cameras, and a GPS/INS receiver. The data from all these sensors is available. The radar sensor detects 360 degrees. The data of the radar sensor is provided as a PNG image where each pixel corresponds to a certain azimuth angle and distance. The value of the pixel is the radar cross-section (RCS). Additionally, ground truth-optimized radar odometry is provided. No objects are annotated in this data set. In Figure 3.5 an example of the radar data is shown. The route consists of a mostly urban setting. Different types of road boundaries are present: curbs, fences, walls.

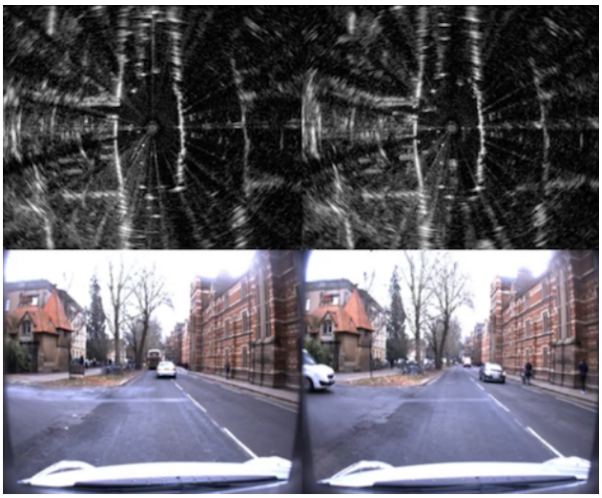


Figure 3.5: Example of camera and radar data from the Oxford Robotcar data set. Source: `https://ori.ox.ac.uk/oxford-radar-robotcar-dataset`.

3.3.4. EU Long-term data set

This data set is recorded on two routes in the downtown and a suburb of Montbéliard, France. There are eleven recordings of the 5 km long downtown route and two recordings of the 4.2 km long suburban route. The suburban route focuses on roundabouts and includes ten of them. The sensor suite consists of four cameras, three 3D LIDARs, one 2D LIDAR, GPS, IMU, and one radar sensor. Although only three recordings include radar data: one of the downtown route and both recordings of the roundabout route. The routes are recorded in different seasons, weather conditions, and times of the day. The data set is not annotated.

3.4. Contributions

As described in Section 3.2, although clustering is a part of many RBE algorithms, it is not used as the main element of the estimation. The only exception is the approach introduced in [36]. Additionally, only a few studies provide quantitative results. An even smaller number of studies describe how the quantitative results were obtained and none of the studies use publicly available data sets. This makes it hard to compare results between different methods.

The contributions of this thesis research are the following:

1. A clustering-based RBE algorithm using radar data is introduced. The envisioned application of the algorithm is for autonomous shuttles driving on (partly) segregated roads with clearly detectable road boundaries.

2. A novel benchmark for radar based RBE is proposed built on the extensive, publicly available nuScenes data set.

3. The introduced RBE algorithm is executed on this novel benchmark resulting in quantative and qualitative results.

4. The Software Development Kit (SDK) that was developed for this study has been made public at `https://github.com/jhoorneman/nuscenes_radar_extension`. The SDK has roughly 3500 lines of code and works as an extension of the nuScenes SDK. With the SDK of this study, all experimental results of this study can be duplicated. In addition, it enables ground truth labeling of radar point clouds; has functionality for general loading, editing, and saving of radar point clouds; has various diagnostic tools; includes an extensive array of rendering options for radar point clouds and annotations.

4

Methodology

This chapter introduces the RBE methodology for this thesis research: Clustering-based Urban Road Boundary Estimation or CURBE. It is based on the method used in [36] which is described in Section 4.1. In addition, Section 4.1 points out the parts [36] is missing to be a fully functional RBE algorithm. CURBE is introduced in Section 4.2. It describes the parts which are new and shows the complete algorithm in pseudo-code. A variation on CURBE that filters all non-static points before clustering is described in Section 4.3.

4.1. The existing RBE method

The method described in [36] clusters the radar point cloud and then fits a line through the clusters that contain the boundary points, the so-called boundary clusters. The method can be divided into three parts: the pre-processing step, clustering, and the actual road boundary line estimation. A visualization from [36] of the clustering and line fitting steps is shown in Figure 4.1.

4.1.1. Approach

The input of this detection method is a set of M radar points. The position of radar point p_m, with $m = \{1, 2, \cdots, M\}$, is expressed as

$$p_m = [R_m, \theta_m, v_m], \tag{4.1}$$

where R_m, θ_m and v_m are the radial distance, azimuth angle, and radial velocity respectively of the radar point. Because of the difference in scale between the radial distance and azimuth angle, this position description is transformed into Cartesian coordinates as

$$p_m = [R_m, \theta_m, v_m] \rightarrow q_m = [x_m, y_m, v_m] \tag{4.2}$$

via the transformations $x_m = R_m \sin\theta_m$ and $y_m = R_m \cos\theta_m$. There are absolute value differences between the x and y coordinates and the radial velocity, so all dimensions are normalized using the standard score as

$$x' = \frac{x - \mu}{\sigma} \tag{4.3}$$

where x', x, μ, and σ are the normalized parameter, original parameter, mean, and standard deviation respectively. After this normalization, all dimensions of a radar point $q_m = [x'_m, y'_m, v'_m]$ have the same scale, which makes it possible to use the Euclidean distance measure without one of the dimensions having minimal effect. A problem might arise from the fact that each dimension is divided by an individual σ. This means the x and y coordinates are scaled differently and that might deform the shapes of objects. However, when the radar point cloud before normalization is symmetric when rotated 90 degrees, the standard deviation of the x and y coordinates are equal and the normalization does not deform the shapes of objects. The radar data used in this study is from radar sensors that detect 360 degrees around the vehicle, which means the radar point cloud from the x and y coordinates is roughly circular and the rescaling has a limited effect on the form of objects.

The DBSCAN clustering algorithm [16] as described in Section 3.1.1 is applied on the M transformed and normalized targets q'_m using the Euclidean distance

$$d(q'_m, q'_n) = \sqrt{(x'_m - x'_n)^2 + (y'_m - y'_n)^2 + (v'_m - v'_n)^2} \tag{4.4}$$

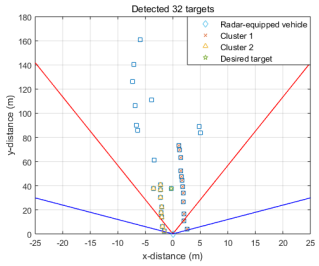
for $m \neq n$ and $n = \{1,2,\cdots,M\}$.

The boundary clusters are selected and spline interpolation is used to obtain a trend line that approximates the road boundary. To describe the road curvature a trend line of second-order or higher is required. In [36] a third-order trend line is chosen. The assumption that the left and right road boundaries are contained in two clusters is made. When a cluster is selected, four target points close to the ego vehicle are chosen and arranged by R_o^y with $o = \{1,2,3,4\}$ in ascending order. These four points are substituted in the equation of the third order trend line

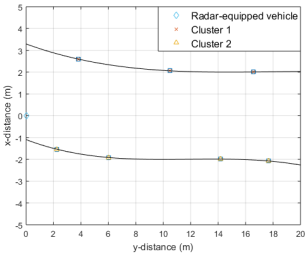
$$g(R_o^x) = a_1(R_o^y)^3 + a_2(R_o^y)^2 + a_3 R_o^y + a_4, \tag{4.5}$$

for each o. This gives a system of four equations that can be solved to estimate the coefficients $a_i (i = 1,2,3,4)$.

(a) An image of the situation.



(b) Clustering applied on the radar data.



(c) Line fitting based on the detected boundary clusters. This figure is rotated a quarter turn and mirrored along the y-axis compared to Figure 4.1b.

Figure 4.1: The RBE method from [36] applied on the Jamsil iron bridge in Seoul, South Korea as shown in their paper.

4.1.2. Missing elements

Two key parts of the RBE algorithm are not described in [36].

First, the selection method of the boundary clusters is not described. In the example shown in the paper, and as shown in Figure 4.1, there are only three clusters detected: the left and right road boundaries and a vehicle, where the last one will have a nonzero absolute radial velocity. In that situation the selection of appropriate boundary clusters is trivial: the left and right static clusters will be the base of the left and right road boundary estimation. However, the example shown is an idealized situation: a straight road with clearly detectable road boundaries on both sides of the road and a minimum number of other road users. In more complex situations there can be more than two clusters consisting of stationary points. For example in a situation with less clearly defined road boundaries or where more stationary objects are present such as trees, buildings, or parked vehicles. In such situations, an algorithm is required to select appropriate boundary clusters.

Second, how the four target points for the third-order trend line that approximates the road boundary are selected is not described in the paper. From Figure 4.1 it seems that the four points closest to the ego vehicle are selected. This is not a robust approach. Even when clearly detectable road boundary structures are present, such as the metal ones in Figure 4.1, radar point of those road boundaries will have a significant variation in lateral position due to the accuracy of the radar sensor. In a situation with boundary structures that are less clearly detectable by radar or more noise, this is not a robust approach.

A solution to both problems is proposed in the next section.

4.2. CURBE: Clustering-based Urban Road Boundary Estimation

We introduce the new algorithm CURBE. CURBE is a clustering-based RBE algorithm that results in a set of labels for each of the points in a radar point cloud that is either *boundary* or *non-boundary*. These estimated labels together with ground truth labels for each point will be used to measure the performance of CURBE, as is described in Section 5.5. As illustrated in Figure 4.2, CURBE consists of four major sub-processes: point cloud pre-processing, clustering, boundary cluster selection, and estimation label assignment which are executed sequentially. The complete CURBE algorithm is shown in Algorithm 1. CURBE operates under the assumption that the ego vehicle is driving on an approximate straight road.

The initial pre-processing steps are equal to [36]: Radar points are transformed to Cartesian coordinates as in Equation 4.2, then the x and y coordinates and radial velocity are normalized with the standard score

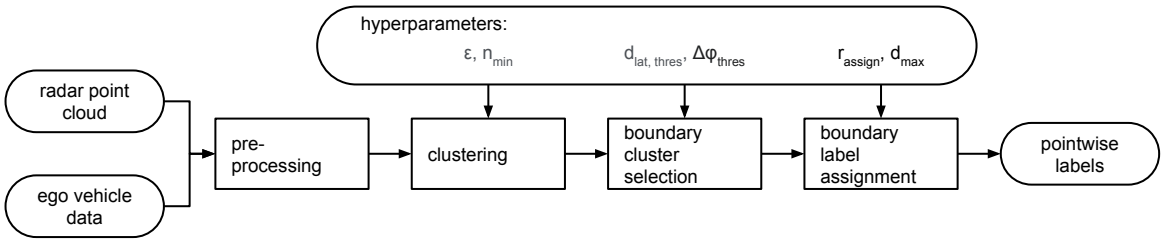


Figure 4.2: Overview of the sub-processes in CURBE.

as in Equation 4.3, and finally, the point clouds are clustered using DBSCAN with the Euclidean distance as described in Equation 4.4.

For the following steps, a new approach is proposed and will be described in this section. It includes the detection boundary clusters, the estimation of road boundary lines, and the translation of these estimated lines to the radar points. The introduced terms are illustrated in Figure 4.3.

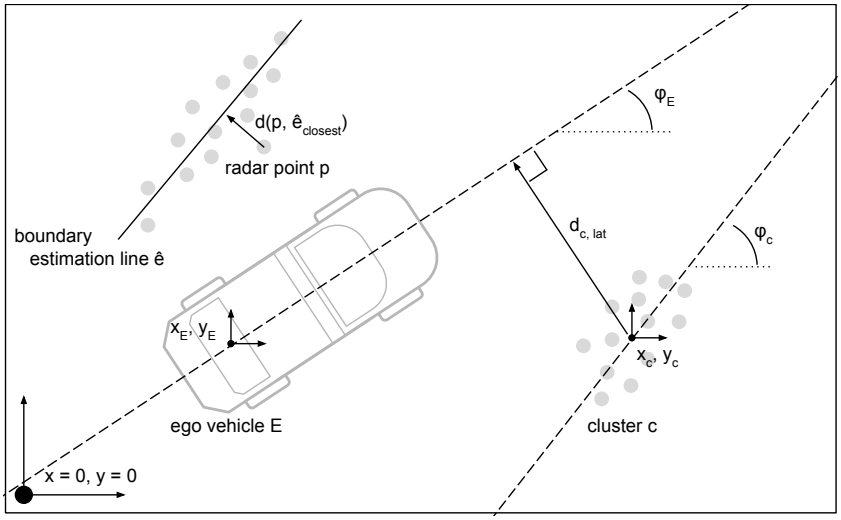


Figure 4.3: Illustration of the terms introduced in CURBE.

To determine which clusters will be considered boundary clusters, we propose a classifier. Besides the information of the clusters, the position and orientation of the ego vehicle are used as input. The ego vehicle position is defined by its x and y coordinates (x_E, y_E) in a global coordinate system. The orientation or heading angle of the ego vehicle ϕ_E is the angle between the ego vehicle's longitudinal axis and the coordinate system's neutral axis.

First the center point and orientation of each cluster is determined. For radar points $j = \{1, 2, \cdots, C\}$ in cluster c, the center point (x_c, y_c) in a global coordinate system is calculated as

$$x_c = \sum_{j=1}^{C} \frac{x_j}{C} \quad y_c = \sum_{j=1}^{C} \frac{y_j}{C}. \tag{4.6}$$

The orientation ϕ_c of each cluster is determined by calculating a first-order trend line using least-squares polynomial fitting and taking the slope of that line in radials. The lateral distance, the distance perpendicular to the orientation of the ego vehicle, of the cluster center to the ego vehicle is calculated as

$$d_{c,lat} = \left| \frac{y_c - y_E - \tan(\phi_E)(x_c - x_E)}{\sqrt{\tan(\phi_E)^2 + 1}} \right|. \tag{4.7}$$

A cluster is considered a boundary cluster when the condition

$$d_{c,\text{lat}} < d_{\text{lat, thres}} \wedge |\phi_c - \phi_E| < \Delta\phi_{\text{thres}} \wedge \#p_{c,static} > \#p_{c,dynamic} \tag{4.8}$$

is satisfied. The parameters $d_{\text{lat, thres}}$ and $\Delta\phi_{\text{thres}}$ are user-defined hyperparameters of CURBE and have to be tuned. As a starting point, the threshold $d_{\text{lat, thres}}$ is typically approximately the width of the road that the ego vehicle drives on and $\Delta\phi_{\text{thres}}$ is between 0 and $\pi/2$ radians. $\#p_{c,static}$ and $\#p_{c,dynamic}$ are the number of static and dynamic points in cluster c. To define a radar point as static or dynamic we use the tags given to each point by the radar sensor, which will be further explained in Section 5.1. Alternatively, the ego motion compensated radial velocity can be used, where a velocity close to zero indicates a static radar point.

We expect the four-point spline method of estimating the road boundary, as used in [36], to not be robust enough since this only leads to an accurate representation when the four points are well chosen. Simply choosing the closest four points, as is done in [36] will not suffice. Instead, we use the fitted first-order polynomial least-squares polynomial from the previously described cluster selection step as an estimation of the road boundary, which we call a road boundary line. Under the assumption that the ego vehicle is driving on a straight road, this is a good estimator of the road boundary. On curved roads, it is less accurate as a road boundary estimator but is expected to be still feasible with wider corners. Sharp corners are not accurately modeled by a straight line, but due to the low resolution of radar data, these are not expected to be correctly represented by clusters. A more complex representation of a road boundary that can model sharp corners is not expected to lead to improved performance.

The boundary lines are translated to point labels in a process that we call estimation label assignment. When a radar point p is within a distance r_{assign} of a boundary estimation line it gets assigned the *boundary* label, $\text{label}_p = 1$. Because the radar point cloud is less sparse as the distance to the ego vehicle increases, we expect the road boundary estimation to get less accurate with increasing distance to the ego vehicle. Therefore, there is a second constraint for the maximum distance to the ego vehicle d_{max}. A radar point p is assigned as *boundary* by the function

$$\text{label}_p = \begin{cases} 1 & \text{if } d(p, \hat{e}_{\text{closest}}) < r_{\text{assign}} \wedge d(p, E) < d_{\text{max}} \\ 0 & \text{else} \end{cases}. \tag{4.9}$$

Where $\text{label}_p = 1$ indicates a boundary point label, $d(p, \hat{e}_{\text{closest}})$ is the distance from point p to the closest road boundary estimation line and $d(p, E)$ is the distance from point p to the ego vehicle.

CURBE has a total of six hyperparameters that have to be tuned for the algorithms to perform optimally. The clustering sub-process has two, as described in Section 3.1.1. The minimum number of points to form a cluster n_{min} and the maximum distance between points to be considered part of a cluster ε. Two more hyperparameters rise from the cluster selection described in Section 4.2. The maximum lateral distance from the ego vehicle $d_{\text{lat, thres}}$ and the accepted difference in the direction of the cluster and ego vehicle $\Delta\phi_{\text{thres}}$. The final two hyperparameters come from the translation step of road boundary estimation lines to radar point estimation labels as described in Section 4.2 as well. These are the point assign radius r_{assign}. And the maximum distance to the ego vehicle d_{max}.

4.3. Static points variation of CURBE

Since for RBE, only the static points are of interest, a variation of CURBE can be made where all non-static points are filtered out in the pre-processing step: Static CURBE or S-CURBE. This in turn means that the radial velocity does not have to be taken into account during clustering, since the radial velocity of all radar points will be close to zero. Additionally, since only x and y coordinates are used in this variation, there is no need to normalize these. This has the benefit that it conserves the original shape of the radar point cloud.

This variation affects only the pre-processing and boundary cluster selection procedures and is illustrated in Algorithm 2. There is no need to adjust the pseudo-code for the clustering procedure. By not including the radial velocity in radar point cloud P the clustering procedure only uses the x and y coordinates.

5

Road boundary detection benchmark

In this chapter we propose a novel benchmark to evaluate radar base RBE algorithms. This benchmark is used in the next chapter to evaluate CURBE's performance. First, from the publicly available data sets as described in Section 3.3 the nuScenes data set is expected to be the most fruitful and is therefore selected for experimental use. The choice for nuScenes is substantiated in Section 5.1 and the data set is described in detail. In Section 5.2 the ground truth labeling for this benchmark is illustrated. In Section 5.3 the other pre-processing steps taken are described. In Section 5.4 we propose two types of subsets of the nuScenes data set to evaluate RBE algorithms on. In Section 5.5 we describe which performance metrics. And, finally, in Section 5.6 the baseline performance to compare RBE algorithms to is introduced.

5.1. nuScenes data set in detail

The nuScenes data set was chosen as the main data set to test the road boundary estimation algorithms. It scores best on almost all criteria. Most importantly, it is the largest annotated data set and includes information of various sensors which are all synchronized in time and location. It has 360-degree radar, LIDAR, and camera coverage. It includes information about the ego vehicle orientation, direction, velocity. In addition, the data set includes a detailed road map and is easy to use due to its software development kit.

The data of the nuScenes data set is recorded in four different locations: the seaport area of Boston, USA, and the One North, Queenstown, and Holland Village districts of Singapore. Driving situations include urban, residential, nature, and industrial areas and are varied in weather conditions and time of day.

The sensor suite includes six cameras, five radar sensors, and one LIDAR sensor. The location and orientation of these sensors are illustrated in Figure 5.1. As radar sensors the Continental ARS-408-21 are used, which are long-range, 77 GHz, Frequency Modulated Continuous Wave (FMCW) sensors. They have a capture frequency of 13 Hz, a range up to 250 m, and measure velocity with an accuracy of ± 0.1 km/h. The camera and LIDAR sensors have a capture frequency of 12 Hz and 20Hz respectively. All these sensors have their intrinsics and extrinsics calibrated. The extrinsic coordinates are described relative to the midpoint of the rear axle of the ego vehicle. The vehicles are also equipped with a GPS and inertial measurement unit (IMU) sensor. To accurately describe the ego vehicle location the LIDAR and odometry information from the IMU is used in a Monte Carlo Localization scheme as described in [8]. This results in accurate localization with errors of ≤ 10 km.

The nuScenes data is stored in a relational data set, as illustrated in Figure 5.2. It consists of a set of elements that describe a certain part of the data. For example, the *ego_pose* element describes the location and orientation of the ego vehicle at a certain point in time. Each element has a unique key, or pointer, that can be used to retrieve the data element from the database.

The *sample_data* element holds the actual data of a single radar, camera, or LIDAR sensor at a certain point in time. Each *sample_data* has references to an *ego_pose* element, that described the location and orientation of the vehicle at roughly the same time, and a *calibrated_sensor* element, that holds all the information of the related sensor.

A *sample* is an annotated keyframe. Each *sample* has references to *sample_data* for all radar, camera, and LIDAR sensors. In addition, a *sample* contains references to all annotated objects in the keyframe, who

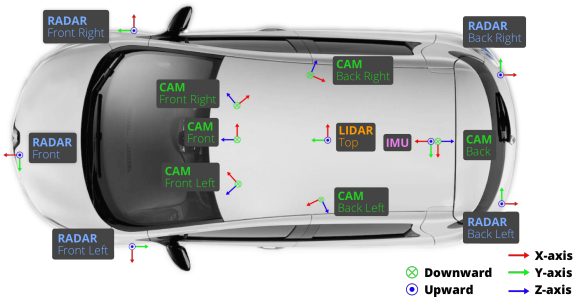


Figure 5.1: The location and orientation of the sensors on the vehicles used for the nuScenes data collection. Source: [42].

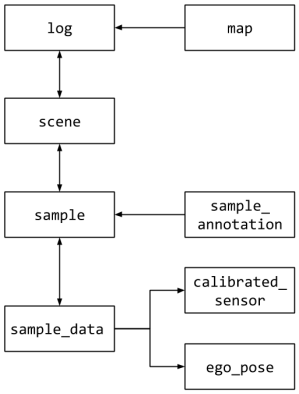


Figure 5.2: A simplified overview of the data structure in the relational nuScenes data set. Each block describes an element in the data base that holds a certain part of the data. An arrow indicates that the element includes a reference to another element. The most important elements are the *scene*, which is a twenty-second long recording, the *sample*, that describes a keyframe in the *scene*, and the *sample_data*, that holds the data of the radar, camera, and LIDAR sensors.

are stored in *sample_annotation* elements. A *scene* is a recording of twenty seconds that contains roughly 40 annotated keyframes. Each *scene* has a short description of the driving situation. The *scenes* recorded in a single drive are bundled together as *logs*. The *log* element also has a reference to the *map* element, which holds the information about the location and layout of the map.

Summarizing, the most important elements for this thesis are the *scene*, which is a twenty-second long recording, the *sample*, that describes an annotated keyframe in that recording, and the *sample_data* that holds the data of the radar, camera, and LIDAR sensors.

The full data set of 1000 scenes is split into an annotated training set of 850 scenes and a testing set of 150 scenes that is not annotated. Only the 850 annotated scenes will be used for the benchmark. The scenes are annotated with a frequency of 2 Hz. That means each scene has around forty samples, synchronized annotated frames, and the complete data set. The complete data set contains 34150 samples. For more numerical information about the data set, see Table 5.1.

Table 5.1: Statistics of the nuScenes data base. The size is expressed in scenes (recordings), samples (annotated key frames), radar sweeps (a single frame of radar data) or radar points. Boundary points are radar points lying on a road boundary and are annotated in a preprocessing step described in Section 5.2.

Data base size	850 annotated scenes	150 non annotated scenes
Samples	34,150 samples	
Radar sweeps	170,000 annotated	1,105,000 non-annotated
Radar points	73,090,000 total points	11,841,000 boundary points

Objects in nuScenes are annotated and given one of 23 different labels. Examples of labels include dif-

ferent types of pedestrians, different types of vehicles, and static objects such as traffic cones, debris, and barriers. Road boundary-defining objects such as fences and guard rails are not annotated in the data set. The closest is the barrier category, which is described in the nuScenes database as: “temporary road barrier placed in the scene in order to redirect traffic. Commonly used at construction sites. This includes concrete barrier, metal barrier, and water barrier. No fences." These are often clearly detectable with radar and might have a significant influence on the performance of the RBE algorithm.

Included in the data set is a binary road map of each area where data was recorded. The map is human-annotated and has a resolution of 10px/m. For each pixel, it states if it is a road or not. We use this map as the ground truth for the road boundary locations.

The radar sensor gives a wide range of information. Besides the radar point coordinates, Doppler velocity, and RCS, additional information is added that is the result of data pre-processing in the radar sensor itself [11]. Three important fields are the invalid state, the ambiguity state, and the dynamic property. The invalid state describes if the radar point is valid or some form of a misdetection. The fields can have eighteen different labels, which are: invalid because of reason x, valid but debatable because of reason x, or valid. By default, nuScenes only includes radar points with the label valid, but this can be changed. The ambiguity state describes the state of the Doppler velocity ambiguity solution. There are five different labels: *unambiguous*, *ambiguous*, and three special cases. By default, nuScenes only includes points with the label *unambiguous*. Points with the labels of *stationary candidate* would be interesting to include for this research, but there seem to be no points with this label in the nuScenes data set, so this can be neglected. The dynamic property field describes if the radar reflection is from a stationary or moving point. The field can have eight different labels. It describes different states of stationary with the possible labels of *stationary*, *stationary crossing*, or *stationary candidate*, and describes different forms of moving. This field can be used as a preprocessing filter since road boundaries are stationary entities. A table describing the invalid state, ambiguity state, and dynamic property fields and their labels can be seen in Table B.1.

Besides the annotated radar sweeps, the non-annotated sweeps are included in the nuScenes data set. The total number of radar sweeps in the data set is 6.5 times the number of annotated radar sweeps: a total of around 220,000 sweeps for each of the five radar sensors. On average, there are between four and six non-annotated radar sweeps in between each annotated keyframe. These radar sweeps do have an *ego_pose* element associated with them, so an accurate position is known.

5.2. Ground truth labeling

The radar points have to be annotated, so a ground truth is established that can be used to evaluate the performance of CURBE. The annotations available in nuScenes are 3D bounding boxes. Individual radar points are not given an annotation label, so these have to be added in a separate step. The nuScenes annotation categories do not include road boundaries. Objects annotated with the *barrier* category often indicate a road boundary, but most road boundaries do not have *barrier* objects and remain unlabeled in the nuScenes database. The detailed top-down binary road map, however, can be used as ground truth to annotate each radar point as a boundary or non-boundary point.

First, the binary road map is processed such that only the road/non-road edges are shown with the border following algorithm from [56]. The result is a one-pixel wide contour of the binary road map that can be used as the road boundary ground truth, the so-called *contour map*.

To annotate a radar point p three criteria have to be satisfied: First, a radar point p has to be close enough to a road boundary e. More formally

$$d(p, e_{\text{closest}}) < r_{\text{annotate}}, \tag{5.1}$$

where e_{closest} is the boundary edge on the contour map closest to point p, $d(p, e_{\text{closest}})$ is the shortest distance from point p to e_{closest} and r_{annotate} is the annotation radius parameter.

Second, radar point p has to be stationary. More formally

$$s_{\text{dynprop}}(p) \in S_{\text{stationary}}, \tag{5.2}$$

where $s_{\text{dynprop}}(p)$ is the dynamic property field of point p and $S_{\text{stationary}}$ is the set of dynamic property labels that indicate a stationary point. As described in Section 5.1 the dynamic property field is information the radar sensor adds to each radar reflection to indicate if the object of that reflection is moving or not.

Third, radar point p cannot be part of an area that is annotated in nuScenes as a non-road boundary object. More formally

$$p \notin \bigcup_{c \in C, i \in I} b_{\text{box}}(c,i), I = \{1,2,...,N_c\}, \tag{5.3}$$

where $b_{\text{box}}(c,i)$ is the i'th bounding box of annotation category c, N_c is the number of bounding boxes in category c and C is the set of annotation categories that do not describe road boundaries. For the nuScenes data set C will include all categories except for *movable_object.barrier*.

Radar point p is annotated as a boundary point if and only if all three conditions 5.1, 5.2, 5.3 are satisfied. Conceptually, a radar point p is labeled as a boundary point when it is within radius r_{annotate} of a boundary edge, is stationary and does not lie within a non-road boundary annotation bounding box,

5.3. Other preprocessing steps

To prepare the data set for road boundary estimation various preprocessing steps have to be taken. The desired ambiguity state and invalid state fields have to be selected and the point clouds have to be transformed to the same frame of reference. Recent non-annotated sweeps can be merged to annotated ones, to get a denser point cloud. To make sure these preprocessing steps have to be performed only once, the radar sweeps are required to be stored afterward.

5.3.1. General preprocessing: point selection and transformation

Only pre-processed data can be used in the experiments, so the preprocessing steps have to be executed on the complete data set. That way, in the experiments there is the free choice of which scenes are used. To accommodate this, the filters that select which radar points are included when loading the raw radar point clouds from the nuScenes database are set to include all feasible points. This means that all points are included when it has a label according to the following criteria:

1. The invalid state field is labeled with some form of valid and
2. the ambiguity state field has the label *unambiguous* or *stationary candidate*.

Also, points from non-annotated radar sweeps are included, as will be described in Section 5.3.2.

Radar points are stored in the nuScenes database in the coordinate frame of the respective radar sensor. To be able to relate these radar points to the binary road map they have to be transformed to the same coordinate frame the binary road map uses. For each sensor, the translation and rotation with respect to the ego vehicle are known and the translation and rotation of the ego vehicle with respect to the map are stored as well. Each radar point p is transformed as

$$p_m = T_{v \to m} T_{s \to v} p_s. \tag{5.4}$$

Where p_s and p_m are the radar point in respectively the sensor and map coordinate system, $T_{s \to v}$ the transformation matrix from the sensor to the ego vehicle coordinate system and $T_{v \to m}$ the transformation matrix from the ego vehicle to the map coordinate system. All following steps will be performed on these transformed radar points p_m.

Radar points in the nuScenes data set have x, y, z coordinates where z is the height. Since the radar sensor only measures in the xy-plane, the z coordinate does not hold relevant information for this research and is neglected.

5.3.2. Combining multiple radar sweeps

With a radar capture frequency of 13 Hz and an annotation frequency of 2 Hz, there are approximately eleven non-annotated radar sweeps per second. In practice, between each annotated radar sweep there are four to six non-annotated sweeps. Combining the sweeps leads to a denser point cloud which can improve the RBE performance of CURBE. This can help prevent radar reflections of objects far away from the ego vehicle from being discarded as noise points. With typical velocities of 10 to 15 m/s of the ego vehicle, the position of the ego vehicle can differ up to 7.5 m between sweeps. But since the location at each radar sweep is known, we can correct for the translation.

To perform the boundary point annotation as described in the previous section, the nuScenes annotations are required. The annotations can be approximated by performing linear interpolation between the

annotation boxes of the two closest annotated sweeps. Per scene, there are no radar sweeps before the first or after the last annotated sweep, therefore the linear interpolation can always be performed.

When combining multiple sweeps the number of sweeps n_{sweeps} is defined. When the value of $n_{\text{sweeps}} = 1$ only the keyframe radar sweep is returned. For each number above that an extra radar sweep is merged. For example, with $n_{\text{sweeps}} = 4$ a keyframe radar sweep and the three most recent sweeps are combined. Before merging the sweeps, each sweep is transformed to the map coordinate frame and boundary points are labeled using the real or approximate annotation bounding boxes depending on whether or not the sweep is a keyframe sweep.

5.3.3. Data storage

After all previously described preprocessing steps the radar point clouds are saved to disk. Besides the coordinates of each point, other important information is stored as well. The *ambiguity state* and *invalid state* fields and the radar sweep number are stored such that only radar points with desired properties are loaded later on. Furthermore, the absolute velocity, the *dynamic property* field, and the boundary/non-boundary label for each point are stored.

5.4. Data set subsets

Due to the limited available computing power, the benchmark uses a subset of the complete nuScenes data set. We propose two ways of selecting a subset of the data set for this benchmark.

Random subset A subset of N randomly selected scenes.

Barrier subset A subset of N scenes with the highest number of *road barrier* annotations.

The random subset represents the general nuScenes data set and can be used in experiments to show how RBE algorithms perform in general urban situations.

The barrier subset consists of scenes that tend to have more clearly defined road boundary structures. In nuScenes, Road boundaries themselves are not annotated or classified for detectability by radar sensors. For this reason the *road barrier* annotation is used as an proxy for scenes with clearly detectable road boundaries. As described in Section 5.1, the *barrier* category consists of movable concrete, metal, or plastic barriers placed to redirect traffic for situations such as construction work. Although fences are explicitly excluded, these are the annotated objects that closest represent the road boundary separation measures that are used for the segregated road of automated shuttles. RBE algorithms are expected to perform better on the barrier subset.

The number of scenes in a subset N can be varied depending on the available computing power. A larger N will generally lead to more accurate results. However, when N becomes too large compared to the maximum of 850 the usage of the random or barrier subset will no longer be meaningful. We use in this research 40 to 60 scenes per subset.

The holdout method is being used to split the data [4]. This means for our application that the subset is split into two equal-sized parts: one is being used as the optimization set and one as the testing set. Since only a subset of the complete data set is used, more advanced optimization-testing set splits do not at any benefits. An example of such an advanced method is the popular k-fold cross-validation that is ideal when data is limited because the complete data set can be used for training and validation at the same time.

First, the hyperparameters of CURBE will be tuned using the optimization partition[1]. Once the hyperparameters are optimized, CURBE using the best performing hyperparameter combination will be evaluated on a separate testing partition.

5.5. Performance metrics

To measure the performance of the RBE algorithm, the problem is described as a binary classification problem. Each radar point is assigned a ground truth label *boundary* or *non-boundary* as described in Section 5.2. By comparing the estimated point labels to the ground truth labels the performance can be expressed.

[1] In many machine learning applications the model parameters are trained using a training set and that performance is validated on a separate data partition: the validation set. Following that convention, the *optimization partition* should be called a *validation partition*. However, in CURBE there is no model parameter training, which makes the term *validation partition* confusing, so *optimization partition* is used instead.

With the ground truth annotations and the road estimation algorithm labels for each radar point, we distinguish the four possible situations: the estimation label can be a true positive (TP), false positive (FP), true negative (TN), or false negative (FN). A positive means the point is estimated to be a boundary point, a negative means the point is estimated to be a non-boundary point. For example, when a point has the estimation label *boundary* and the ground truth annotation for the point is *boundary* as well, CURBE has correctly classified it as a boundary point and it is a TP. When the ground truth annotation for the point would be *non-boundary* it would be an FP. This happens in the same way for TN and FN when the estimation label is *non-boundary*.

There is a significant class imbalance in the data: as shown in Section 5.1 around 84% of the points is a non-boundary point . Because of this, using the accuracy, the ratio of correctly labeled points, as a metric is not a good metric: by simply labeling all points as non-boundary points an accuracy of 84% would be achieved. More meaningful metrics are precision and recall. Recall, also called sensitivity sometimes, is the percentage of all true boundary points that were classified as boundary point and is defined as

$$recall = \frac{\#TP}{\#TP + \#FN}, \tag{5.5}$$

where $\#TP$ and $\#FN$ describe the number of true positives and false negatives in an experiment. Intuitively, it measures the ability of CURBE to find all the boundary points. The precision is the percentage of all classified boundary points that are truly boundary points and is defined as

$$precision = \frac{\#TP}{\#TP + \#FP}, \tag{5.6}$$

where $\#TP$ and $\#FP$ describe the number of true positives and false positives in an experiment. The precision is intuitively the ability of CURBE not to label a non-boundary point as a boundary point. When the recall is high, the precision is generally lower and the other way around. For example, assuming ten percent of the radar point is a boundary point, when classifying all radar points as boundary points, the recall will be 1, but the precision will be 0.1. The precision-recall curve will be given for each experiment to see the relation between the two metrics.

A third metric, the F_1-score, will be used as the main optimization parameter. It is defined as

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}} = \frac{\#TP}{\#TP + \frac{1}{2}(\#FP + \#FN)} \tag{5.7}$$

and can be interpreted as the harmonic mean of the recall and precision [4].

5.6. Baseline

The metrics are compared to a hypothetical baseline where all radar points are estimated to be boundary points. This is the most naive way of estimating the road boundaries and all RBE algorithms should perform better than this baseline. From Equation 5.5, the $recall = 1$ in that situation, since all of the boundary points are correctly classified as such. From Equation 5.6, the $precision$ will be the ratio of boundary points to all the radar points in the optimization set, since $\#TP + \#FP$ describes all radar points.

6

Experiments

To measure the performance of CURBE as described in Chapter 4 the algorithm is executed on the benchmark that was introduced in Chapter 5. In Section 6.1 the setup of the different experiments will be discussed and how they relate to the research questions. In Section 6.2 and 6.3, the results of these experiments are presented and discussed. Finally, in Section 6.4 the experiment results on a few representative nuScenes samples are investigated in a qualitative analysis of the results.

6.1. Description of experiments

Experiments are performed by executing CURBE on the benchmark that was introduced in Chapter 5.

In total seven hyperparameters require tuning. In Section 4.2 the clustering parameters are introduced: n_{min} and ε and the boundary cluster selection parameters: $d_{\text{lat, thres}}$ and $\Delta\phi_{\text{thres}}$. Furthermore, the two point label assignment parameters introduced in Section 5.5: r_{assign} and d_{max}. Finally, the number of radar sweeps used as input n_{sweeps} as described in Section 5.3.2. n_{sweeps} is not a model parameter in the sense that it does not affect the way CURBE evaluates the data presented, but we group it with the hyperparameters in order to investigate the effect of different numbers of included sweeps. The hyperparameters and their description are shown in Table 6.1.

Table 6.1: All hyperparameters that have to be tuned for CURBE, the sub-process they belong to, their description and the section where they were introduced.

Sub-process	Parameter	Description	Section
Data input	n_{sweeps}	Number of included radar sweeps as input	5.3.2
Clustering	n_{min}	Minimum number of points for a cluster to be formed	3.1.1
	ε	Maximum distance between points to be considered part of the same cluster	3.1.1
Boundary cluster selection	$d_{\text{lat, thres}}$	Maximum lateral distance that a boundary cluster can be from the ego vehicle	4.2
	$\Delta\phi_{\text{thres}}$	Maximum allowed angle difference between the ego vehicle orientation and a boundary cluster direction	4.2
Estimation label assignment	r_{assign}	Boundary points have to be within this radius of an boundary estimation line	5.5
	d_{max}	Boundary points can be at most this distance from the ego vehicle	5.5

The hyperparameters are tuned using a grid-search algorithm on the optimization set. For each hyperparameter, a set of reasonable values is be determined. Then CURBE is ran using all combinations of these hyperparameter values. Afterward, using the hyperparameter combination that maximizes the F_1-score CURBE is executed on the testing set to assess the performance on independent data.

CURBE is tested in three different experiments, which are each designed to lead to an answer to one of the research questions as presented in Section 1.1. Remembering, the research questions were:

1. Is clustering-based RBE using radar data reliable enough to be used for autonomous shuttles?
2. How does the existence of clear road boundary structures affect the performance of CURBE
3. Does the inclusion of ego-motion compensated radial velocity as a dimension in the clustering process lead to better RBE performance compared to filtering out non-static points before clustering?

The experiments differ in which variant of CURBE is used or which subset of the nuScenes data set is used to evaluate the algorithm. An experiment can make use of the main CURBE algorithm as described in Algorithm 1 or the S-CURBE variation that only uses static radar points as described by Algorithm 2. These algorithms are executed on one of two subsets of the data set that were described in Section 5.4: the *random* subset that consists of 40 randomly selected scenes or the *barrier* subset that consists of the 60 scenes with the highest number of *road barrier* annotations. The experiments have a name, built up as Experiment <*CURBE variation*>-<*subset of data*> and are described in short as:

Experiment CURBE-random Using CURBE on the random subset with 40 scenes.

Experiment CURBE-barrier Using CURBE on the barrier subset with 60 scenes,

Experiment S-CURBE-random Using S-CURBE on the same random subset as used in Experiment CURBE-random.

The stated number of scenes for each experiment consists of the optimization and testing set combined. Both optimization and testing sets have half the number of scenes stated.

Experiment CURBE-random is running CURBE on a random subset of the data. The goal of the experiment is to investigate how CURBE performs on the general nuScenes data and to obtain results to compare with the following experiments. 20 scenes are used for parameter optimization and 20 different scenes are used to evaluate the best parameter combination. The selected values for each of the hyperparameters in the grid search optimization are shown in Table 6.2.

The goal of Experiment CURBE-barrier is to answer research question 2. In the experiment, CURBE is applied to scenes that have the highest number of annotated in the *barrier* category. This experiment and Experiment CURBE-random use the same CURBE variant but are applied to different subsets of the data set. By comparing the results of the two experiments the effect of the barriers on the RBE performance can be investigated. The barriers are expected to be well detectable by radar sensors, especially those made out of metal, and therefore lead to increased performance. The same grid-search parameter combination as in Experiment CURBE-random is used, see Table 6.2.

Experiment S-CURBE-random will investigate research question 3. In the experiment, a variation of CURBE is used that first filters out all non-static points and does not use the radial velocity in the clustering process. Because the distance metric for DBSCAN only uses the x- and y-coordinates of each radar point measured in meters, normalizing the spatial dimensions is not required. The optimization and test partitions are identical to the partitions in Experiment CURBE-random. This way the RBE performance of the algorithm variants can be compared to find an answer to research question 3.

Because the radar point coordinates are not normalized before the clustering step, different values of clustering parameter ε are needed. The selected grid-search parameters are shown in Table 6.2.

The hyperparameter combination as stated in Table 6.2 show that in Experiment CURBE-random and CURBE-barrier a total of 8,100 combinations and for Experiment S-CURBE-random 11,340 combinations are computed.

6.2. Results of hyperparameter optimization

For the hyperparameter optimization, CURBE is executed on the respective optimization set of each experiment. This is the most extensive part of each experiment since CURBE is executed on the optimization partition for each combination of the hyperparameters, as stated in Table 6.2. Note that the only difference in hyperparameter values between the CURBE and S-CURBE experiments is in the hyperparameter ε. This is because in CURBE before clustering, the standard score of the x and y coordinates and the radial velocity is taken, whereas in S-CURBE the unmodified x and y coordinates are used.

Table 6.2: Selected hyperparameter values for the grid search optimization of each experiment. The experiments CURBE-random and CURBE-barrier are executed with the same CURBE variant and thus have identical hyperparameter values.

experiment(s)	ϵ	n_{min}	$\Delta\phi_{\text{thres}}$	$d_{\text{lat, thres}}$	r_{assign}	d_{max}	n_{sweeps}
CURBE-random and	0.025	2	0.1745	6.0	0.5	50.0	1
CURBE-barrier	0.0375	3	0.3491	8.0	1.0	75.0	3
	0.05	4	0.5236	10.0	2.0	100.0	5
	0.0625				3.0	125.0	
	0.075				4.0		
S-CURBE-random	0.25	2	0.1745	6.0	0.5	50.0	1
	0.5	3	0.3491	8.0	1.0	75.0	3
	1	4	0.5236	10.0	2.0	100.0	5
	1.5				3.0	125.0	
	2				4.0		
	2.5						
	3						

The $recall$, $precision$, and F_1-score resulting from executing CURBE for each of the hyperparameter combinations will be referred to as the grid-search results.

Three different figures are included to illustrate the results of the grid-search optimization and will be explained in the following paragraphs. The results for the three experiments will be shown in subplots in each figure.

6.2.1. Presentation of results

The absolute values of $recall$, $precision$, and F_1-score are relevant here as a measure of RBE performance since this only shows how well CURBE performs when it is optimized for the same data set it is tested on. However, the distribution of the metrics can give insight into the nature of CURBE. Histograms showing the distributions of each of the metrics are shown in Figure 6.1.
The metrics are compared to the baseline that is described in Section 5.6 where all radar points are estimated to be boundary points. The baseline for the $recall$ is not shown in the plots, since it will be 1 in every situation and therefore does not portray any relevant information.

The influence of each hyperparameter on the F_1-score is illustrated in a series of box plots in Figure 6.2. Each box is representing a set of CURBE executions where the value of one of the hyperparameters is fixed and the other hyperparameters appear in all possible combinations. In other words, each box represents the results of a grid-search with six of the seven hyperparameters, where the seventh hyperparameter assumes a fixed value. The box represents the distribution of this subset of the results by showing the median, the 25th, and 75th percentile and the minimum and maximum values of the F_1-scores.

While the median, and the 25th and 75th percentile give insight into the distribution of F_1-scores, the maximum F_1-score is the most important, since the hyperparameter combination that led to that F_1-score is the one that will be used in the end. It can be noted that the maximum F_1-score in a box plot, i.e. the highest F_1-score of all boxes in a box plot combined, will be the same for each box plot in a single experiment since the hyperparameter combinations from all boxes in a plot together describe the complete grid-search.

The same baseline as used for Figure 6.1 is shown in the box plots. As described in the previous paragraph, it is the hypothetical classifier that labels all radar points as boundary points.

In Appendix C the 2D extension of the box plots in Figure 6.2 is shown in Figures C.1, C.2 and C.3. In these plots, the maximum F_1-score is shown for each pair of hyperparameter values. It is a visual representation of a table, where each cell shows the maximum F_1-score of the subset of grid-search results where two hyperparameters are kept at fixed values and the remaining hyperparameters appear in all combinations. The maximum F_1-score is chosen to be shown for each subset of grid-search results since this represents a combination of hyperparameters that will be used in the subsequent experiments.

The table is square and symmetric along the diagonal. The cells for pairs of the same hyperparameter are empty since a hyperparameter can only have one value.

Table 6.3: For each experiment, the total number of radar points and number of ground truth boundary points is shown. The numbers are split for the optimization and test set and for each number of included radar sweeps.

experiment	type	$n_textsweeps$	total points	boundary points	boundary points ratio
CURBE-random	optimize	1	161,774	27,992	0.173
		3	476,921	82,600	0.173
		5	791,689	137,32	0.173
	test	1	160,861	25,651	0.159
		3	474,351	75,589	0.159
		5	787,977	125,442	0.159
CURBE-barrier	optimize	1	242,058	36,981	0.153
		3	713,642	109,185	0.153
		5	1,185,181	181,494	0.153
	test	1	257,388	42,991	0.167
		3	759,119	126,991	0.167
		5	1,260,512	210,509	0.167
S-CURBE-random	optimize	1	139,406	27,992	0.201
		3	410,696	82,600	0.201
		5	681,611	137,32	0.201
	test	1	139,782	25,651	0.184
		3	411,961	75,589	0.183
		5	684,354	125,442	0.183

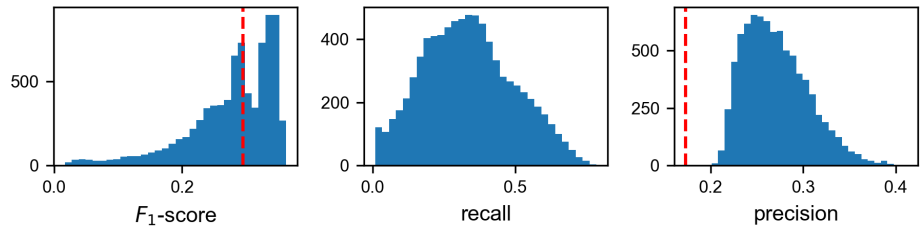
6.2.2. Discussion of results

When inspecting the metric histograms of Figure 6.1 the first thing that stands out is that the $precision$ and F_1-score are quite low. The maximum $precision$ of all experiments is around 0.5, meaning that for that combination of hyperparameters 50% of the radar points classified as boundary point are truly a boundary point. The recall varies between 0 and 0.8, meaning that there are hyperparameter combinations for which 80% of the boundary points get classified as such. However, because the precision is relatively low, in those situations there seem to be a lot of false positives as well.

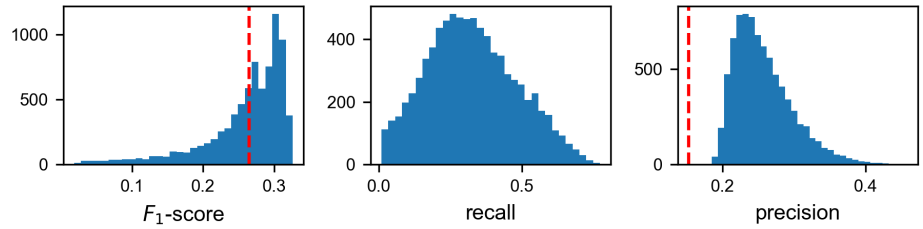
When comparing the distributions of Experiment CURBE-random and Experiment CURBE-barrier, they are very similar. However, in the metric distributions for Experiment S-CURBE-random, there is an abnormal number of hyperparameter combinations that have an $F1$-score and recall close to zero. The precision has a peak at zero as well, but much lower. The F_1-score is determined by the recall and precision, so those two metrics are the most important. The recall has the largest peak. Remembering from Equation 5.5 and 5.6 that the recall is defined as $\frac{\#TP}{\#TP+\#FN}$ and precision as $\frac{\#TP}{\#TP+\#FP}$. The recall can be zero when $\#TP$ is zero. However, this can not be the cause since the precision should be zero as well in that situation. Thus, the only explanation can be that there is a large number of false negatives compared to true positives. So, a lot of boundary points are not getting classified as such for a relatively large number of hyperparameter combinations.

An investigation into executions from the S-CURBE-random experiment with low recall confirms that this is because of large numbers of false negatives. When looking at the hyperparameter values, almost all low recall executions have an $n_{\text{sweeps}} = 1$ and $\varepsilon \in \{0.25, 0.5, 1\}$. This shows that the peak at near-zero recall is caused by simply a bad combination of parameters. Radar point clouds with $n_{\text{sweeps}} = 1$ are sparse and with $\varepsilon \in \{0.25, 0.5, 1\}$ points need to be very close to each other to form a cluster. This means only a very small number of clusters form, resulting in a very small number of boundary clusters, resulting in a very high number of radar points that are estimated to be *non-boundary* points, resulting in a very high number of false negatives. This can also been seen in Figure 6.2 for experiment S-CURBE-random where the distributions of $n_{\text{sweeps}} = 1$ and $\varepsilon \in \{0.25, 0.5, 1\}$ show a large number of low values.

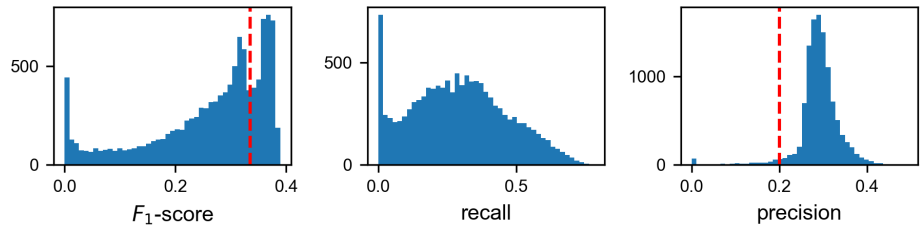
The influence of each hyperparameter value on the F_1-score is illustrated in Figure 6.2. Looking at the plots from Experiment CURBE-random and CURBE-barrier it shows a similar view to the metric histograms: most F_1-scores lie above the baseline. In Experiment CURBE-barrier the results show that n_{sweeps} is a strong



(a) Experiment CURBE-random.



(b) Experiment CURBE-barrier.



(c) Experiment S-CURBE-random.

Figure 6.1: Distribution of metrics after performing a grid search optimization of CURBE. The vertical axis shows the number of different hyperparameter combinations that led to the results in each of the bins of the histogram. The striped red line is the baseline which is the hypothetical situation where all radar points are classified as boundary points. The baseline for the $recall$ is not shown since it is always 1, regardless of the class balance, and thus uninteresting.

predictor of good results. With $n_{\text{sweeps}} = 5$ the complete distribution is above the baseline although the maximum F_1-score is the lower than for the other values of n_{sweeps}.

Regarding Experiment S-CURBE-random, for lower values of ε the distribution has a very high variance and is mostly below the baseline. As described earlier, this is because lower values of ε in combination with $n_{\text{sweeps}} = 1$ results in a very small number of clusters being formed, which has a big negative influence on the recall and F_1-score.

6.3. Results of CURBE run on the testing data partition

CURBE using the hyperparameter combination that performed best for each of the experiments is executed on an independent test set. The results from that are presented and discussed in this section.

6.3.1. Presentation of results

The best performing hyperparameter combinations and their corresponding F_1-score for each of the experiments are shown in Table 6.4. In Figure 6.3 the precision-recall curves are presented, illustrating the trade-off between the two metrics. These curves are constructed by changing the boundary point labeling threshold r_{annotate}. A radar point p gets classified by CURBE with Equation 4.9. To construct the precision-recall curve d_{max} is kept at the value as stated in Table 6.4 and r_{annotate} is changed from the minimum to the maximum value of $d(p, \hat{e}_{\text{closest}})$ for all radar points in the test set. Then, for each r_{annotate} the precision and recall are calculated and plotted as a data point in the precision-recall curve.

In precision-recall curves, the recall often goes from 0 to 1, but since there is a secondary condition for

boundary point label assignment, not all radar points will get labeled boundary point, even as $r_{\text{annotate}} \to \infty$.

Figure 6.4 shows the distribution of the F_1-score per nuScenes sample in the test set. The F_1-score for the complete test set for the respective experiment is shown as a purple, dashed line.

Table 6.4: The best performing parameter combinations on the optimization set of each experiment. The F_1-score on the optimization and testing sets are presented as well.

	n_{sweeps}	ϵ	n_{min}	$\Delta\phi_{\text{thres}}$	$d_{\text{lat, thres}}$	r_{assign}	d_{max}	optimis. F_1-score	test F_1-score
CURBE-random	5	0.075	3	0.1745	10	3	125	0.363	0.304
CURBE-barrier	3	0.075	2	0.1745	8	4	125	0.3263	0.3412
S-CURBE-random	5	2.5	2	0.1745	10	4	125	0.3893	0.3411

6.3.2. Discussion of results

Analyzing the performance as indicated in Table 6.3, the most outstanding feature is that the overall F_1-score is relatively low, implying that the current CURBE is not very reliable in detecting boundary points. Remember that the F_1-score is the harmonic mean of recall and precision, as shown in Equation 5.7: both recall and precision are required to obtain high values in order for the F_1-score to obtain high values. When inspecting the precision-recall curves in Figure 6.3 the cause of this is the low precision, meaning there are a lot of false positives compared to true positives. We discuss this phenomenon further in Section 6.4.

Comparing the final, test F_1-score for Experiment CURBE-random and CURBE-barrier, running the algorithm on scenes with a high number of barrier annotations results in a significant improvement in results. During the hyperparameter optimization, Experiment CURBE-random had better results, but these did not generalize well to the testing set. The results of Experiment CURBE-barrier generalize well, even improving the results on the testing set compared to the optimization set.

S-CURBE, as used in Experiment S-CURBE-random, shows a significantly higher F_1-score as well, suggesting that this should be used instead of CURBE.

The F_1-score distribution amongst nuScenes samples as shown in Figure 6.4 shows a similar pattern in all experiments: most nuScenes samples have an F_1-score of approximately 0.3.

6.4. Qualitative analysis of results

In this section, the performance of CURBE on the test set is inspected. This means the hyperparameter values from Table 6.3 are used. From a few representative nuScenes samples, including ones with high and low F_1-scores, the plots of intermediate results are analyzed to see in which situations affect CURBE's performance.

Three Figures are used to help with the qualitative analysis: Figure 6.5, 6.6, and 6.7. Each figure shows the view from all six camera sensors for reference and the intermediate steps of CURBE in four plots. The information in the plots should be interpreted as follows:

- **Top left**: the starting point. The radar point cloud with the ground truth boundary points is shown as well as which points are dynamic.
- **Top right**: clustering. Each cluster is highlighted together with the cluster direction line.
- **Bottom left**: boundary cluster selection. The clusters which are selected as boundary clusters are highlighted with the resulting RBE lines.
- **Bottom right**: boundary label assignment. The results of the boundary label assignment by CURBE are shown. For each radar point, it is shown whether it is a true or false positive or negative.

In each of the plots, the location of the ego vehicle is shown as a red cross and the orientation of the ego vehicle is indicated by a red vector.

As illustrated by the figures in this section there seem to be four main factors that affect CURBE's performance:

Presence of radar detections: A trend amongst the worst performing samples, such as the one shown in Figure 6.7 is the low number of radar detections. Because of this, there are only a small number of viable clusters, and often very few or none of them get selected as boundary clusters, which in turn means very few to none estimated boundary points.

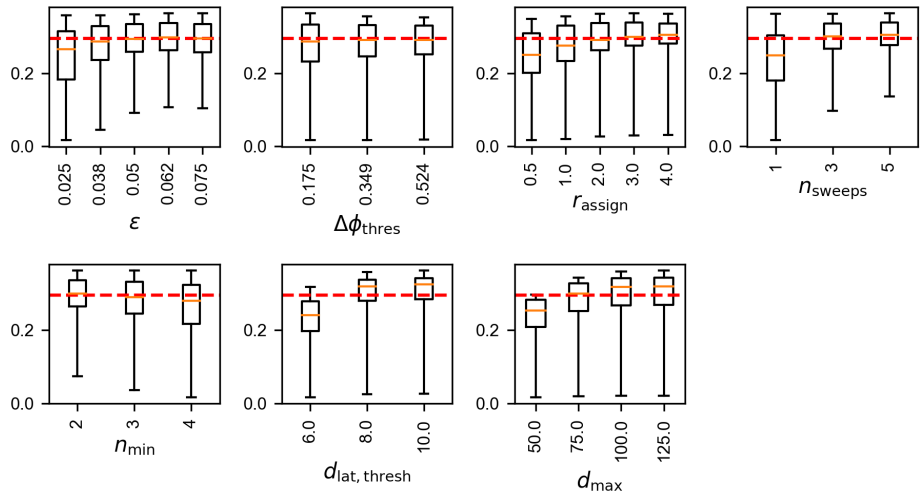
Road separation objects: The presence of clearly detectable road boundary objects is very important. When road boundary objects are present these show in a lot of detections in the radar point cloud as can be seen for the fences and guard rails in Figures 6.5 and 6.6. Other objects that are detectable by radar are parked vehicles and buildings. In the nuScenes data set, most scenes do not have these detectable road boundary objects. Instead, curbs are the most common. But as can be seen in Figure 6.7, the radar sensors do not seem to be capable of detecting curbs. The radar sensor measures points at a fixed height and curbs are often too low to the ground to get picked up.

Road geometry: The shape of the road is another important factor for CURBE's performance. Because the road boundaries are estimated by a straight line and the direction of that line has to be similar to the ego vehicle orientation, CURBE does not perform well in situations with complex road geometry, as can be seen in Figure 6.7. Curved roads give a problem for the straight-line approximation as well. Although, sometimes it still results in a good estimation as in Figure 6.6. Figure 6.6 also shows, however, that only the parts of the road boundary that line up the ego vehicle direction get approximated correctly. The part of the road that curves left in front of the ego vehicle is not modeled correctly. Similar situations occur when the ego vehicle is at an intersection and takes a right or left turn. In that case, the orientation of the ego vehicle also does not line up with the road boundaries anymore and the estimations fail.

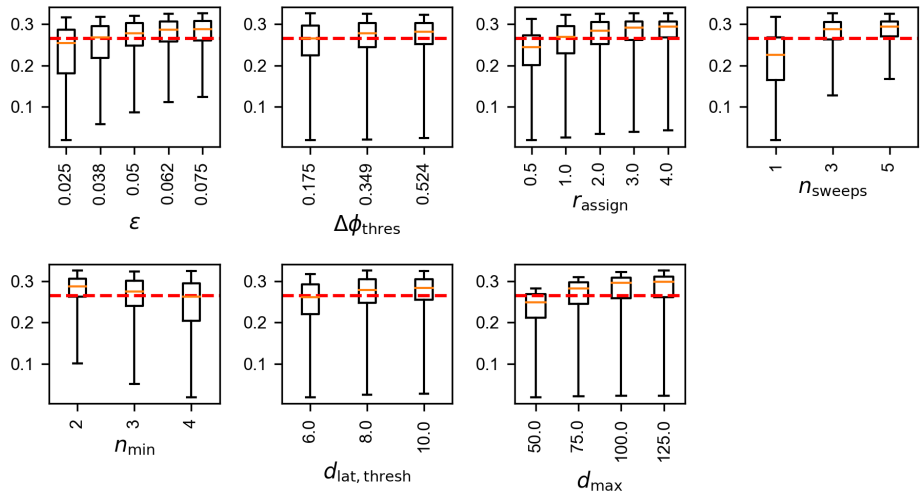
Another problem with intersections, as illustrated in Figure 6.7, is that it introduces boundary points of intersection roads. These, however, do not get detected CURBE and are thus a source of false negatives.

The number of ground truth boundary points: Not all radar points that a human annotator would expect ground truth boundary points are in fact boundary points. When there are clearly detectable road separators, such as the fence in Figure 6.5 and the guard rails in Figure 6.6, part of the radar points do not have the ground truth label of boundary point. As stated in Section 5.2 a radar point is annotated as boundary point when it is within r_{annotate} of a road boundary. The reason some radar points on the fence are not annotated as boundary point is that they are more than r_{annotate} away from the road boundary. Fences and guard rails are often not placed on the road boundary as indicated by the nuScenes binary road map, 0.5 to 1.5 meters away from it. This leads to the radar sensor detecting a boundary object and CURBE estimating it as a road boundary, which results in a set of false-positive points.

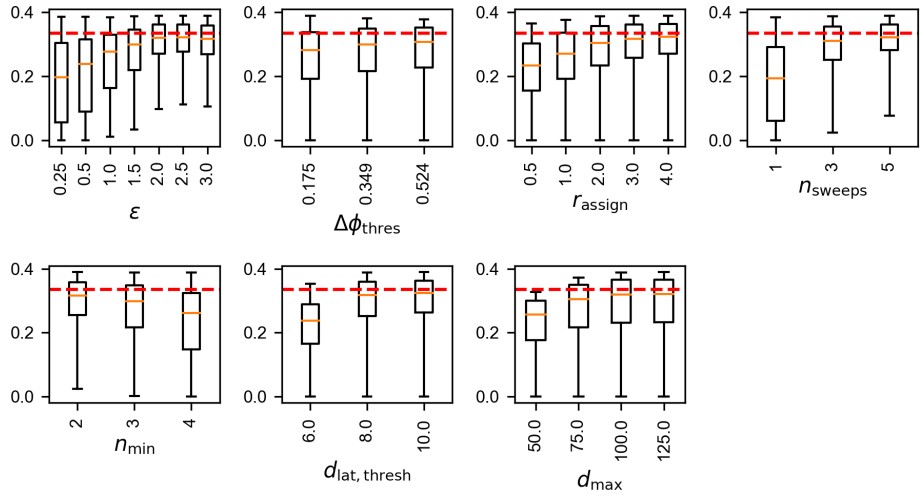
The same occurs in an urban situation where buildings are very close to the road. The buildings are detected by the radar and CURBE will estimate the road boundary is at the buildings, but because there is some space in between the road and the buildings, for example, a sidewalk, these road boundary estimations will be false positives.



(a) Experiment CURBE-random.



(b) Experiment CURBE-barrier.



(c) Experiment S-CURBE-random.

Figure 6.2: The F_1-score distribution of a subset of grid-search results where one of the hyperparameters is fixed and the other six appear in all possible combinations. The striped red line is the baseline which is the hypothetical situation where all radar points are classified as boundary points. Three sets of seven hyperparameter plots are shown.

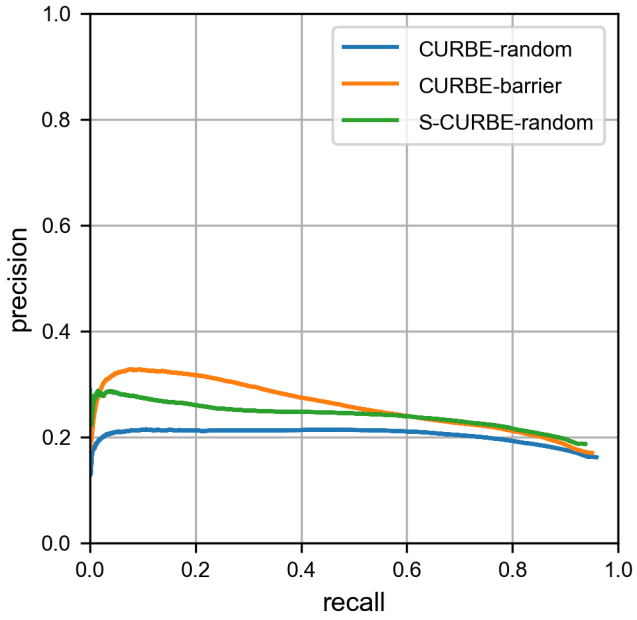


Figure 6.3: The precision-recall curves when varying the r_{assign} is shown for CURBE and S-CURBE using the hyperparameter combinations that performed best on the optimization partition, see Table 6.4.

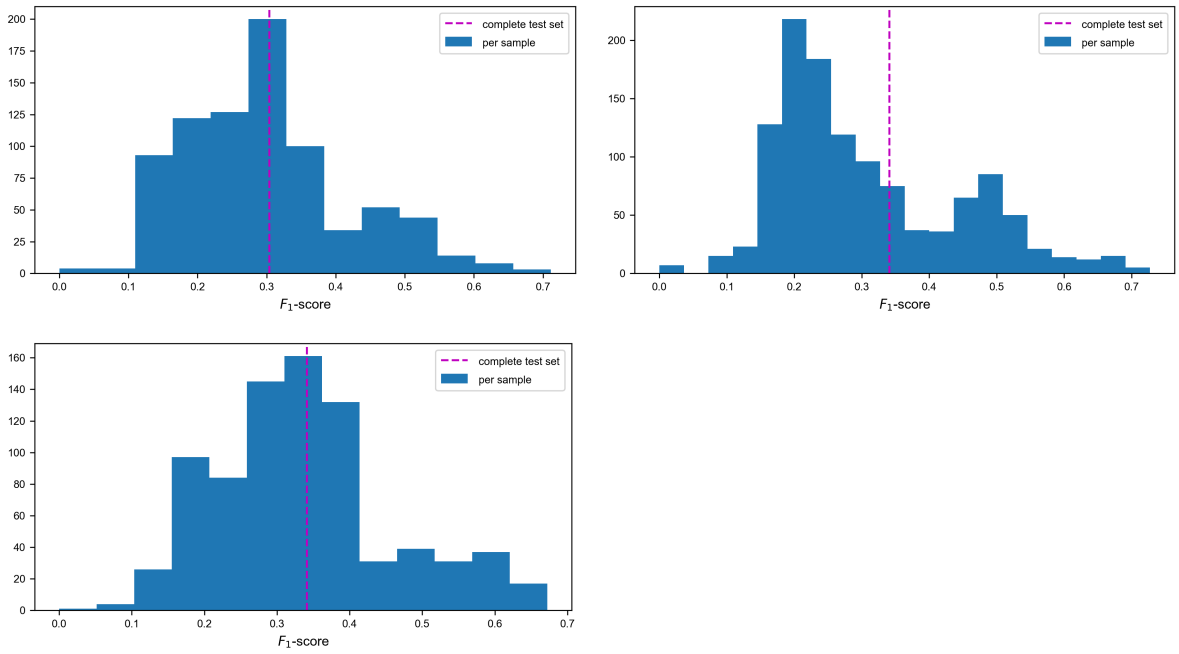
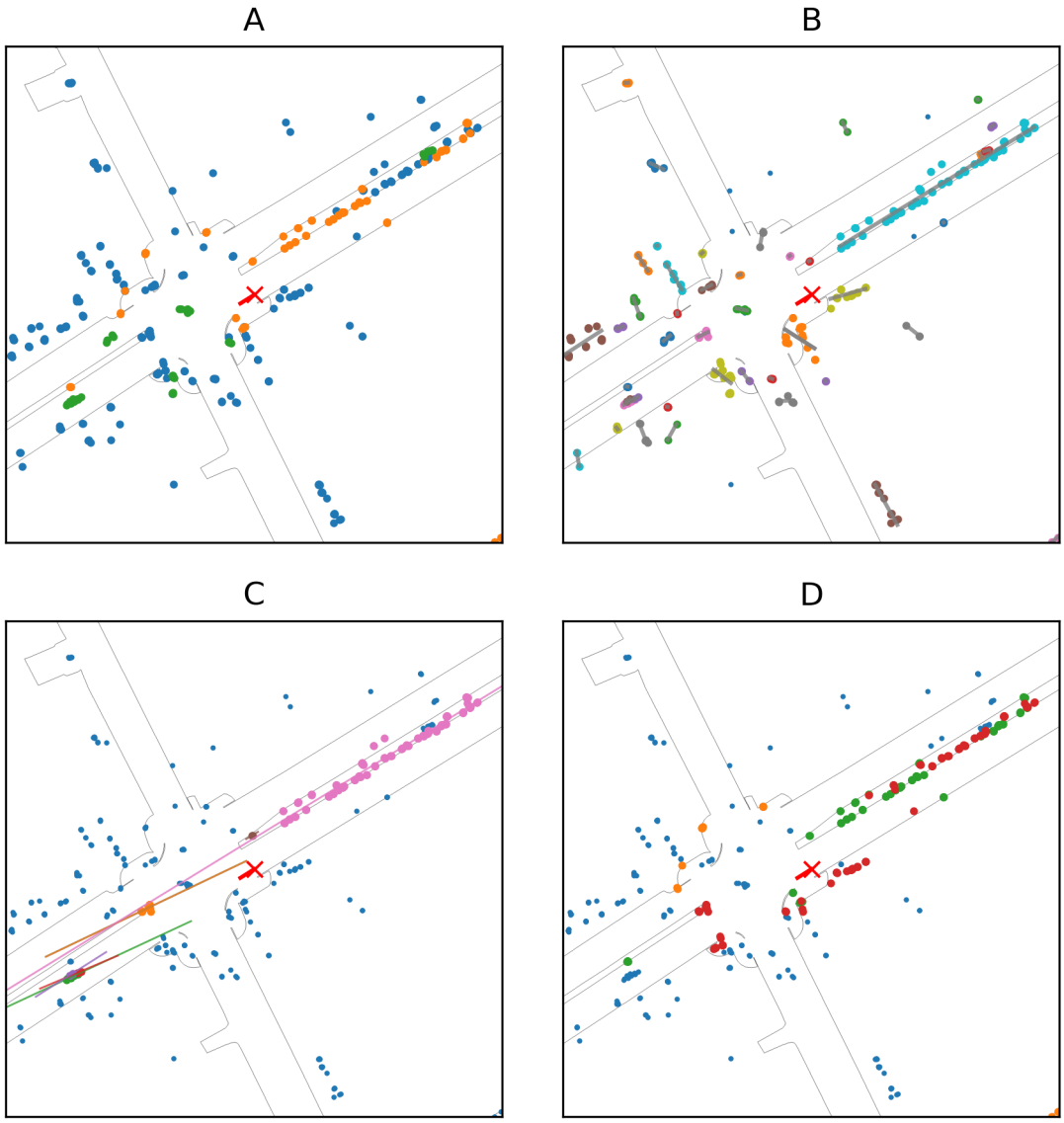


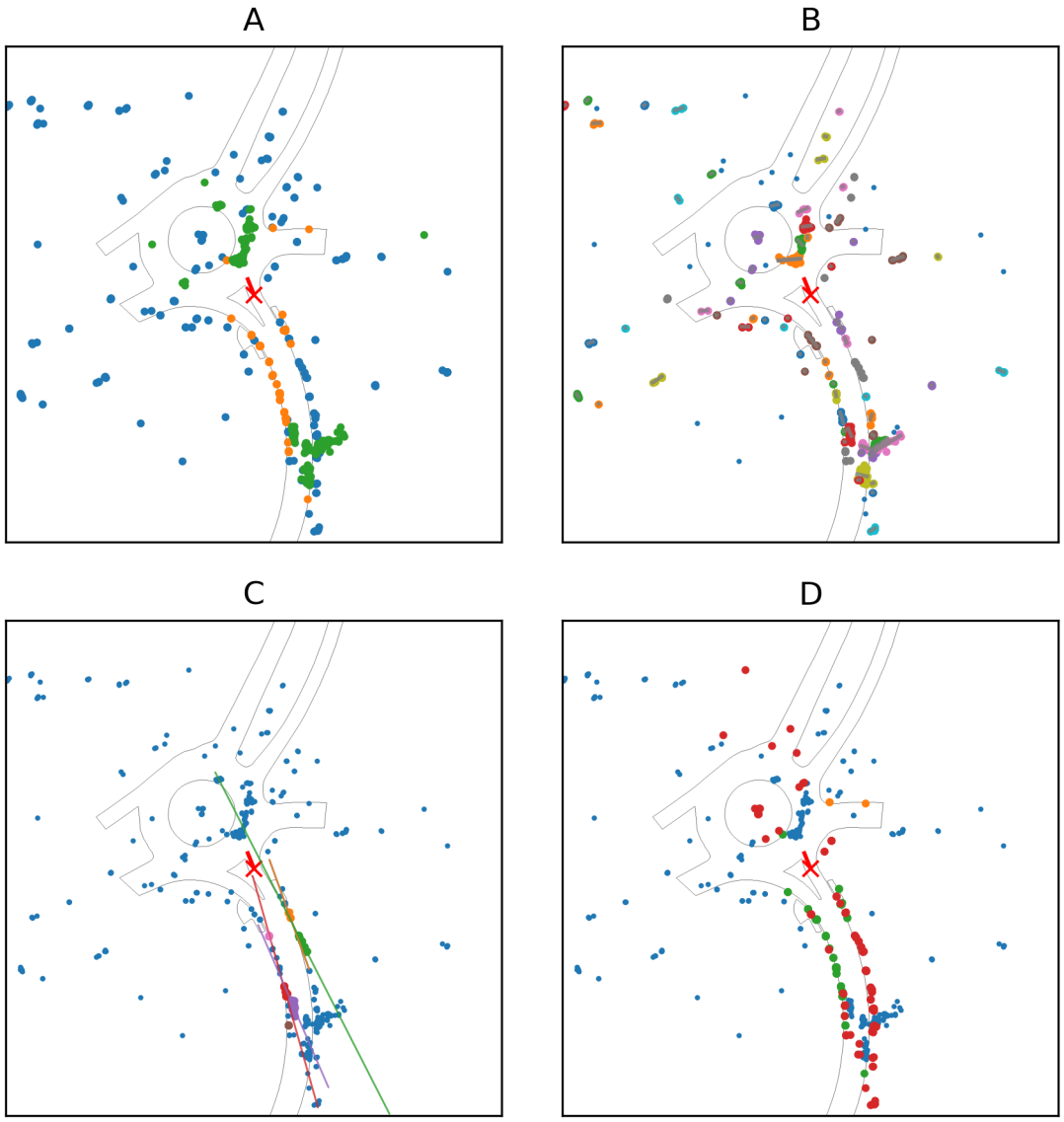
Figure 6.4: The distribution of F_1-score per nuScenes sample is shown for the hyperparameter combination that performed best on the optimization partition for each experiment, see Table 6.4. The vertical axis shows the number of scenes that had a certain result. The dashed purple line is the average F_1-score for the complete test set.
Top left: Experiment CURBE-random, **top right:** Experiment CURBE-barrier, **bottom left:** Experiment S-CURBE-random.



(a) Four plots showing different stages of CURBE. In all plots, the ego vehicle location is shown in the center as a red x and the ego vehicle orientation as a red vector. The different plots are: **A:** the boundary point ground truth. Points in orange are labeled as boundary point, points in blue as non-boundary point, and points in green are from dynamic objects. **B:** the clusters and the fitted cluster direction lines. Each cluster is a group of colored points. The smaller points in blue are outliers. **C:** the selected boundary clusters and boundary estimation lines resulting from it. The smaller blue points belong to non-boundary clusters or are outlier points. **D:** the final results of CURBE. True positives are shown in green, false positives in red, false negatives in orange and true negatives in blue.

(b) Image data from the same situation. Data is shown from: **A:** front camera. **B:** rear camera.

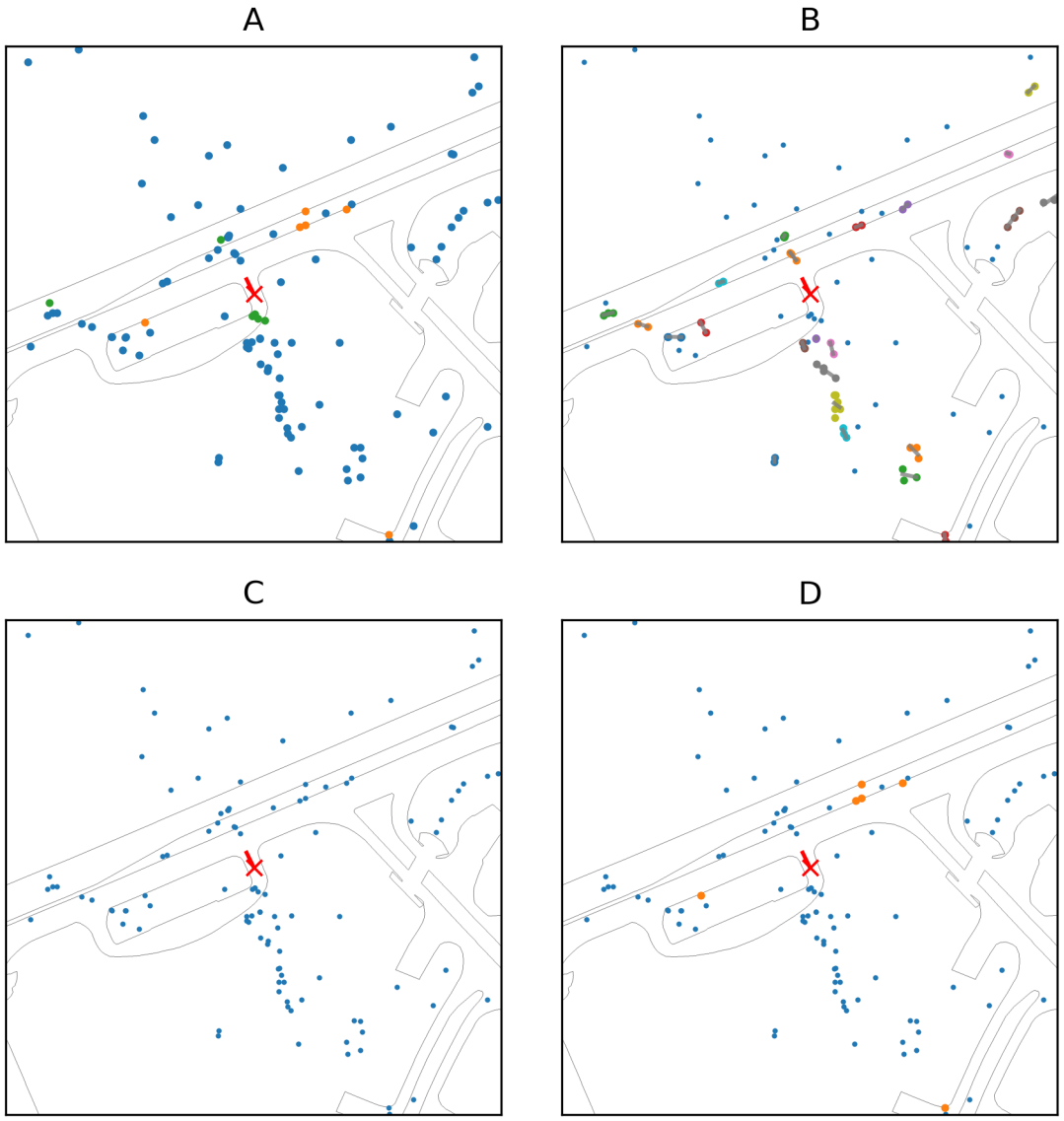
Figure 6.5: Intermediate steps of CURBE shown on a sample from Experiment CURBE-random.



(a) Four plots showing different stages of CURBE. In all plots, the ego vehicle location is shown in the center as a red x and the ego vehicle orientation as a red vector. The different plots are: **A:** the boundary point ground truth. Points in orange are labeled as boundary point, points in blue as non-boundary point, and points in green are from dynamic objects. **B:** the clusters and the fitted cluster direction lines. Each cluster is a group of colored points. The smaller points in blue are outliers. **C:** the selected boundary clusters and boundary estimation lines resulting from it. The smaller blue points belong to non-boundary clusters or are outlier points. **D:** the final results of CURBE. True positives are shown in green, false positives in red, false negatives in orange and true negatives in blue.

(b) Image data from the same situation. Data is shown from: **A:** front camera. **B:** rear camera.

Figure 6.6: Intermediate steps of CURBE shown on a sample from Experiment S-CURBE-random.



(a) Four plots showing different stages of CURBE. In all plots, the ego vehicle location is shown in the center as a red x and the ego vehicle orientation as a red vector. The different plots are: **A:** the boundary point ground truth. Points in orange are labeled as boundary point, points in blue as non-boundary point, and points in green are from dynamic objects. **B:** the clusters and the fitted cluster direction lines. Each cluster is a group of colored points. The smaller points in blue are outliers. **C:** the selected boundary clusters and boundary estimation lines resulting from it. The smaller blue points belong to non-boundary clusters or are outlier points. **D:** the final results of CURBE. True positives are shown in green, false positives in red, false negatives in orange and true negatives in blue.

(b) Image data from the same situation. Data is shown from: **A:** front camera. **B:** rear camera.

Figure 6.7: Intermediate steps of CURBE shown on a sample from Experiment CURBE-barrier.

7

Conclusion and discussion

In this thesis research, a method for clustering-based Road Boundary Estimation (RBE) using radar data was studied. The goal was to find a suitable RBE method that can be used for autonomous shuttles. A new method was introduced and tested on a novel benchmark for RBE techniques.

Results show that CURBE gives mediocre performance: CURBE reached an F_1 score of 30.4% and S-CURBE was the best performing variation with an F_1-score of 34.1%.

In idealized situations with straight roads, road boundary objects that are clearly detectable by radar, such as fences and guard rails, CURBE performs decently and can reach F_1-scores up to 70%. Most situations, however, are not as idealized. CURBE will perform significantly worse in situations where there are no clearly detectable road obstacles, there is a lot of road curvature, complex road geometry, or when there is much clutter in the form of parked cars or bikes, or other irregular metal structures.

Another important observation is a flaw in chosen method of measuring the performance of CURBE. By phrasing the RBE as a binary classification problem it makes it easy and fast to generate quantitative results, which is a big advantage, but it introduces some complications as well. The ground truth annotation introduces an extra step in between the real ground truth, the binary road map, and the RBE algorithm. This introduces an extra source for error.

More importantly, the performance of CURBE is only evaluated for road boundaries that have close-by radar points. A road boundary without radar points that is not detected by CURBE will not affect the recall or precision since it does not increase the number of false negatives. This means an RBE method can reach

Without taking road boundaries with no radar points into account, the benchmark can still adequately compare different RBE techniques since it will still measure the performance on the road boundaries that do have radar points. However, approaches that manage to estimate the road boundary in the absence of radar points, for example by using nearby objects that can be detected, will not see an increase in F_1-score. A method to measure performance directly from the binary road map would lead to results that more accurately describe the actual performance of CURBE and other methods.

7.1. Answering the research questions

To come back to the research questions for this thesis study, starting with the sub-questions and ending with the main research question. We restate each of the research questions and the corresponding hypotheses, after which we present a conclusion.

How does the existence of clear road boundary markings affect the performance of the RBE algorithm?

Hypothesis: Building on the hypothesis from the previous research question, the expectation is that more clear road boundary markers such as fences, guards rails or potentially even bushes will show a significant increase in performance.

Conclusion: This hypothesis has been shown to be correct. Using CURBE on a random subset of scenes in the nuScenes data set led to an F_1-score of 30.9% while using the same algorithm on scenes with a high number of *barrier* annotations (which indicates more fences and guard rails) led to an F_1-score of 34.1%

Does the inclusion of ego-motion compensated radial velocity as a dimension in the clustering process lead to better RBE performance compared to filtering out non-static points before clustering?

Hypothesis: Even though including the ego-motion compensated radial velocity as an extra clustering dimension will give a more complete representation of the surroundings, it will not improve results. Since only static radar points are of interest for the final results, including non-static points will not give direct benefits while introducing an extra source of potential misdetections. For example, imagine a situation where a class of school children is walking in a row on the sidewalk. The radar sensor will detect a set of points close together approximately on a straight line, which will be clustered together. Then, in later processing steps of the RBE algorithm, it might be concluded incorrectly that this is a road boundary.

Inclusion of the ego-motion compensated radial velocity compared to ignoring it will show a significant increase in performance. However, the specific approach used to include it will have a marginal effect since both approaches should be capable of finding similar clusters of static points.

Conclusion: It was hypothesized that whether the radial velocity is used or not would have a limited effect on the results. However, it was shown in Section 6.3 that when comparing CURBE to S-CURBE the F_1-score increased from 30.4% to 34.1%. This is a significant improvement, so this hypothesis is incorrect.

Is clustering-based RBE using radar data reliable enough to be used for autonomous shuttles?

Hypothesis: Since this approach tries to detect the road boundaries, as opposed to the absence of obstacles in the occupancy grid approach, these road boundaries have to be detectable by the radar sensors. The method will work well on roads with clearly defined road boundaries that are detectable by radar sensors. Straight roads or roads with only light curvature will make for more accurate road boundary estimations.

Conclusion: This was the main research question for this thesis study. For an RBE algorithm to be considered reliable the F_1-score should be at least higher than 50%. It was hypothesized that the method would show good performance on roads with clearly defined road boundary obstacles. Although an increase in performance was shown for roads with these boundary obstacles, this was not enough to be considered a good performance. With the current RBE algorithm and with the current method of measuring performance there are not enough grounds to conclude this method as a reliable way of road boundary estimation. But further development of the method has the potential to change this.

7.2. Future work

Based on the findings, we can see various directions of future work.

An important subject for further research should be how CURBE performs when different metrics are used. An interesting approach would be to take ground truth road boundary lines as a starting point. For example, the contour map as constructed in Section 5.2 could be used. A metric based on the similarity between these ground truth boundary lines and the estimated boundary lines could improve the current approach. This would decrease the performance of RBE algorithms that do not estimate road boundaries in absence of radar points. Additionally, this would take away the need for the ground truth point labeling step with the r_{annotate} parameter as described in Section 5.2 and the boundary label assignment step of CURBE and the hyperparameter r_{assign} as described in Section 4.2.

Afterward, implementing a new performance metric, running the most promising RBE algorithms as described in Section 3.2 on the newly introduced benchmark in this study, would be interesting. To see how the approaches perform on a different data set and how they compare to each other can give important insights in new directions of research in the field of radar-based RBE.

Another topic of interest is to explore what effects different clustering methods have. Currently, we use the basic DBSCAN in CURBE, but as shown in Section 3.1.3 there are clustering methods specifically designed for radar data. For CURBE, as a method that has clustering as a core technique for its functionality, we expect this to significantly increase the performance.

As an extension of this, more advanced techniques could be used to improve CURBE. For example clustering the road boundary estimations, as was done in Kim and Song [27], so there is a single road estimation line

on one side of the road could improve accuracy. Another example is to detect other road users with methods like Palffy et al. [44] and track their paths along with different frames or even predict it for future frames to help the detection of road or curb surfaces.

When the performance is significantly increased to at least an F_1-score of over 50%, CURBE should be adjusted such that it is able to deal with curved roads and intersections. To accomplish this a road boundary representation other than a straight line has to be implemented.

A

CURBE pseudo code

Pseudo code for the complete algorithm for CURBE and S-CURBE are shown here in Algorithm 1 and Algorithm 2.

Algorithm 1 Main road boundary estimation algorithm CURBE.

$pointcloud \leftarrow \{p_{\text{raw},0}, p_{\text{raw},1}, ..., p_{\text{raw},N}\}$ ▷ Raw data of each radar point
$E \leftarrow \{x_E, y_E, v_E, \phi_E\}$ ▷ Ego vehicle data

procedure PREPROCESSING($pointcloud, E$)
  $P \leftarrow \{\}$ ▷ Set of pre-processed radar points
  for $i := 0$ **to** N **do**
    $p_{\text{raw}} \leftarrow pointcloud(i)$
    $p.x \leftarrow p_{\text{raw}}.distance \cdot \sin(p_{\text{raw}}.azimuth)$
    $p.y \leftarrow p_{\text{raw}}.distance \cdot \cos(p_{\text{raw}}.azimuth)$
    $p.v \leftarrow p_{\text{raw}}.v - v_E \cdot \cos(p_{\text{raw}}.azimuth)$
    $P(i) \leftarrow p$
  end for
  for $i := 0$ **to** N **do**
    $P(i) \leftarrow$ standard_score($P(i)$) ▷ Normalized with standard score as in Equation 4.3
  end for
  return P
end procedure

procedure CLUSTERING($P, \varepsilon, n_{\min}$)
  $clusters$ = DBSCAN($P, \varepsilon, n_{\min}$) ▷ Applied as in [16] using Euclidean distance
  return $clusters$
end procedure

procedure BOUNDARYCLUSTERSELECTION($E, clusters, d_{\text{lat, thres}}, \Delta\phi_{\text{thres}}$)
  $boundary\_clusters \leftarrow \{\}$
  for c **in** $clusters$ **do** ▷ Each cluster c is a set of N_c radar points
    $x_c, y_c \leftarrow$ average x and y coordinates of cluster c
    $d_{c,lat} \leftarrow$ lateral distance between (x_E, y_E) and (x_c, y_c) ▷ See Equation 4.7
    $f_c(x) = a_c \cdot x + b_c \leftarrow$ least_squares_fitting(c)
    $\phi_c \leftarrow \arctan(a_c)$
    num(c_{static}) $\leftarrow$ number of static points in c
    num($c_{dynamic}$) $\leftarrow$ number of dynamic points in c
    if $d_{c,lat} < d_{\text{lat, thres}} \wedge |\phi_c - \phi_E| < \Delta\phi_{\text{thres}} \wedge \text{num}(c_{static}) > \text{num}(c_{dynamic})$ **then**
      append c **to** $boundary\_clusters$
    end if
  end for
  return $boundary\_clusters$
end procedure

procedure BOUNDARYLABELASSIGNMENT($P, E, boundary\_clusters, r_{\text{annotate}}, d_{\max}$)
  $estimation\_lines \leftarrow \{\}$
  for b **in** $boundary\_clusters$ **do**
    $\hat{e} = a_b \cdot x + b_b \leftarrow$ least_squares_fitting(b)
    append $\hat{e}$ **to** $estimation\_lines$
  end for
  labels $\leftarrow \{\}$
  for p **in** P **do**
    $d(p, \hat{e}_{\text{closest}}) \leftarrow \min(\{\text{shortest distances from } (x_p, y_p) \text{ to each line } \hat{e} \text{ in } estimation\_lines\})$
    $d(p, E) \leftarrow$ distance between (x_p, y_p) and (x_E, y_E)
    if $d(p, \hat{e}_{\text{closest}}) < r_{\text{annotate}} \wedge d(p, E) < d_{\max}$ **then**
      append 1 **to** labels ▷ Estimated boundary point
    else
      append 0 **to** labels ▷ Estimated non-boundary point
    end if
  end for
  return labels
end procedure

Algorithm 2 S-CURBE: The variation of CURBE, as described in Algorithm 1, that only uses static points. Only the pre-processing and boundary cluster selection procedures are changed compared to Algorithm 1.

$pointcloud \leftarrow \{p_{\text{raw},0}, p_{\text{raw},1}, ..., p_{\text{raw},N}\}$ ▷ Raw data of each radar point

procedure PREPROCESSING($pointcloud$)
  $P \leftarrow \{\}$ ▷ Set of pre-processed radar points
  for $i := 0$ **to** N **do**
    $p_{\text{raw}} \leftarrow pointcloud(i)$
    if p_{raw} is a dynamic point **then**
      continue
    else
      $p.x \leftarrow p_{\text{raw}}.distance \cdot \sin(p_{\text{raw}}.azimuth)$
      $p.y \leftarrow p_{\text{raw}}.distance \cdot \cos(p_{\text{raw}}.azimuth)$
      $P(i) \leftarrow p$
    end if
  end for
  return P
end procedure

procedure BOUNDARYCLUSTERSELECTION($E, clusters, d_{\text{lat, thres}}, \Delta\phi_{\text{thres}}$)
  $boundary\_clusters \leftarrow \{\}$
  for c **in** $clusters$ **do** ▷ Each cluster c is a set of N_c radar points
    $x_c, y_c \leftarrow$ average x and y coordinates of cluster c
    $d_{c,lat} \leftarrow$ lateral distance between (x_E, y_E) and (x_c, y_c) ▷ See Equation 4.7
    $f_c(x) = a_c \cdot x + b_c \leftarrow$ least_squares_fitting(c)
    $\phi_c \leftarrow \arctan(a_c)$
    if $d_{c,lat} < d_{\text{lat, thres}} \wedge |\phi_c - \phi_E| < \Delta\phi_{\text{thres}}$ **then**
      append c **to** $boundary\_clusters$
    end if
  end for
  return $boundary\_clusters$
end procedure

B

Radar sensor additional data

The Table B.1 shows the different point state fields that the Continental ARS 408-21 radar sensor includes.

Table B.1: Description of different point state information that the Continental ARS 408-21 radar sensor sends [11]

Signal	Description	Values
Cluster_InvalidState	State of Cluster validity state	0x00: Valid 0x01: Invalid due to low RCS 0x02: Invalid due to near-field artefact 0x03: Invalid far range Cluster because not confirmed in near range 0x04: Valid Cluster with low RCS 0x05: reserved 0x06: Invalid Cluster due to high mirror probability 0x07: Invalid because outside sensor field of view 0x08: Valid Cluster with azimuth correction due to elevation 0x09: Valid Cluster with high child probability 0x0A: Valid Cluster with high probability of being a 50 deg artefact 0x0B: Valid Cluster but no local maximum 0x0C: Valid Cluster with high artefact probability 0x0D: reserved 0x0E: Invalid Cluster because it is a harmonics 0x0F: Valid Cluster above 95 m in near range 0x10: Valid Cluster with high multi-target probability 0x11: Valid Cluster with suspicious angle
Cluster_AmbigState	State of Doppler (radial velocity) ambiguity solution	0x00: invalid 0x01: ambiguous 0x02: staggered ramp 0x03: unambiguous 0x04: stationary candidates
Cluster_DynProp	Dynamic property of cluster to indicate if is moving or not	0x00: moving 0x01: stationary 0x02: oncoming 0x03: stationary candidate 0x04: unknown 0x05: crossing stationary 0x06: crossing moving 0x07: stopped

C

Additional results

This appendix shows additional visualisations of the grid-search in Chapter 6. In each matrix two of the hyperparameters have fixed values and of the remaining hyperparameter combinations the maximum F_1-score is shown. This is similar of plots shown in Figure 6.2 where one of the hyperparameters is fixed. Note that the matrices are mirrored along the diagonal axis.

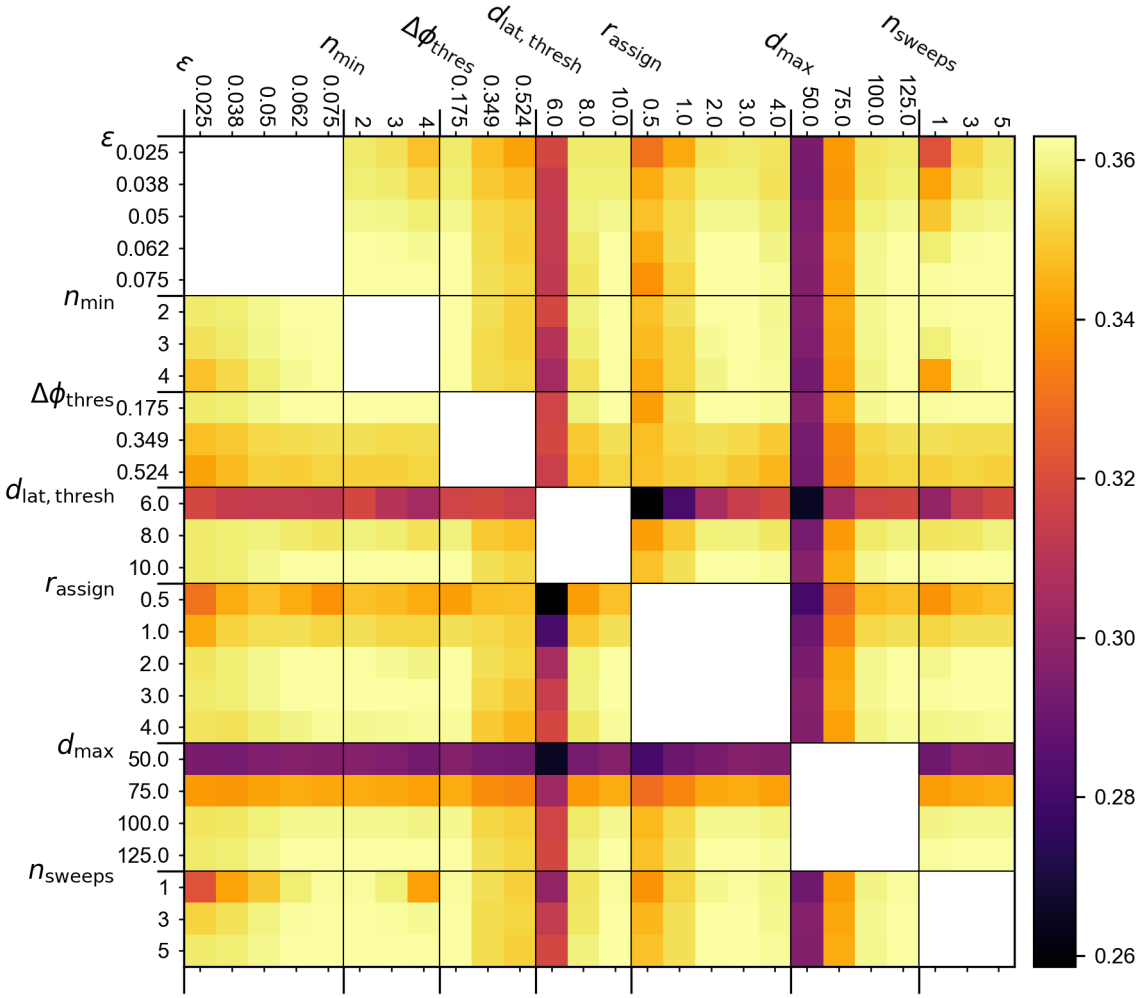


Figure C.1: Visual representation of a table of maximum F_1-scores for different hyperparameter combinations in the grid-search optimization of Experiment CURBE-random. Each cell represents the maximum F_1-score of a subset of the grid-search results where two of the hyperparameters have a fixed value and the remaining hyperparameters have all combinations. Note that the colors in the different plots do not represent the same values, they are meant to illustrate the different outcomes of the grid-search optimization per experiment.

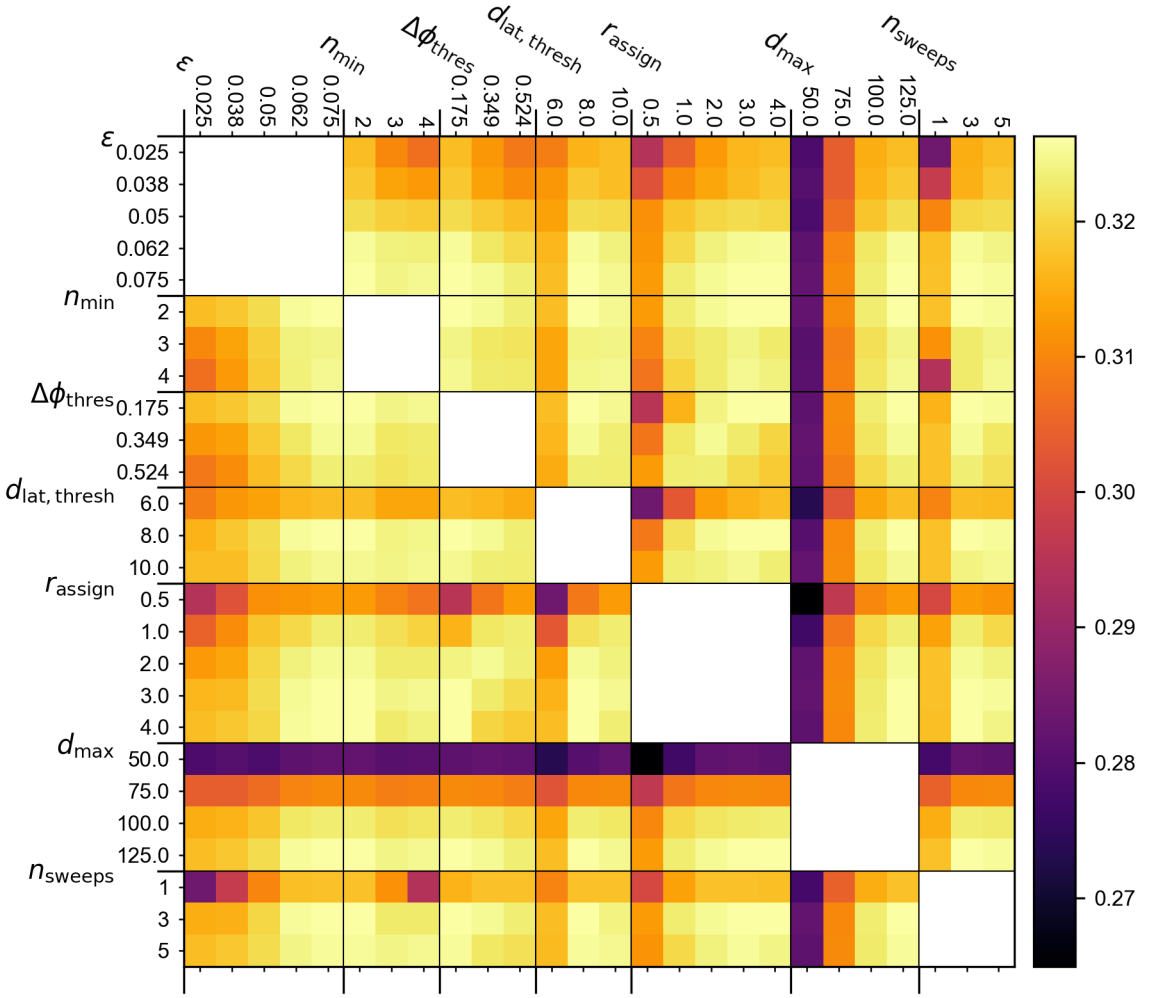


Figure C.2: Visual representation of a table of maximum F_1-scores for different hyperparameter combinations in the grid-search optimization of Experiment CURBE-barrier. Each cell represents the maximum F_1-score of a subset of the grid-search results where two of the hyperparameters have a fixed value and the remaining hyperparameters have all combinations. Note that the colors in the different plots do not represent the same values, they are meant to illustrate the different outcomes of the grid-search optimization per experiment.

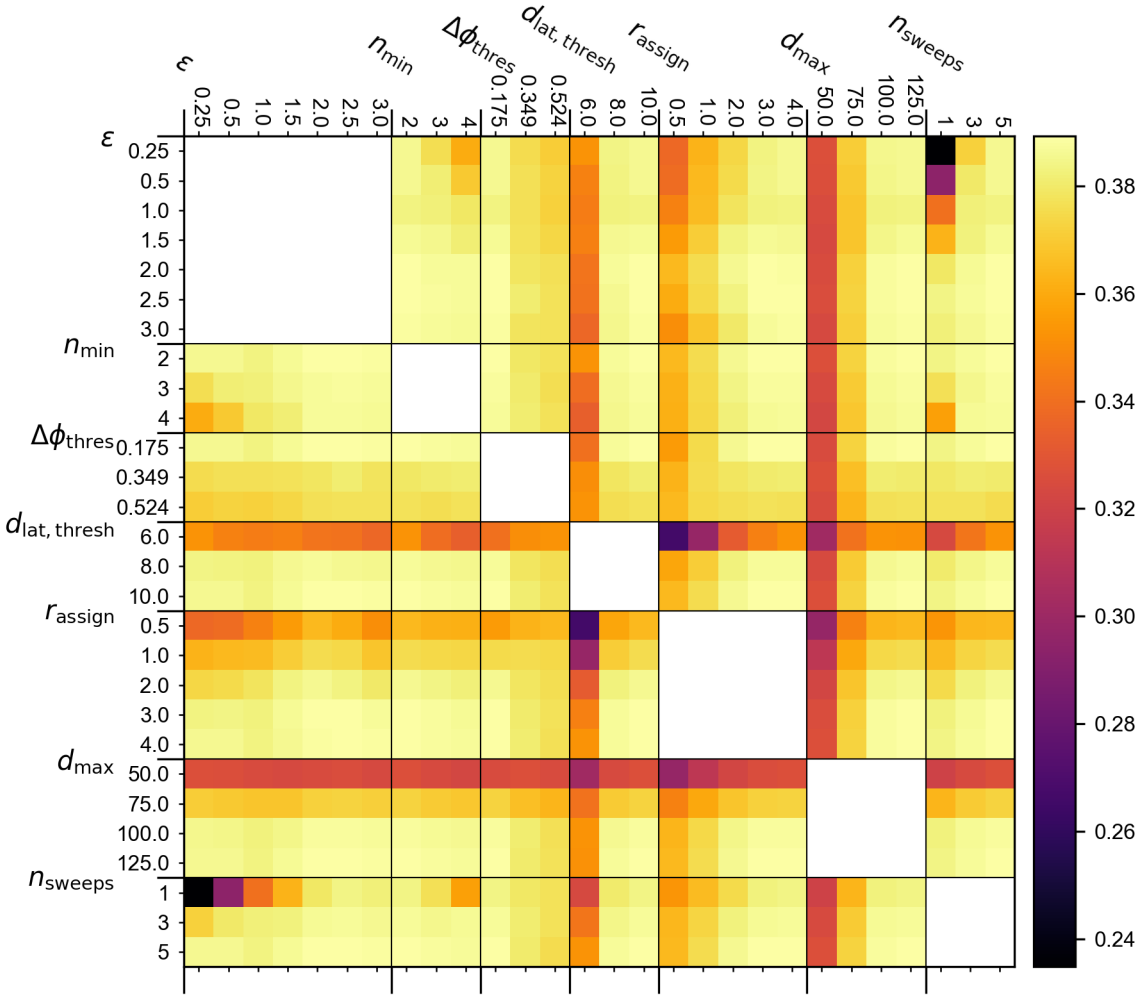


Figure C.3: Visual representation of a table of maximum F_1-scores for different hyperparameter combinations in the grid-search optimization of Experiment S-CURBE-random. Each cell represents the maximum F_1-score of a subset of the grid-search results where two of the hyperparameters have a fixed value and the remaining hyperparameters have all combinations. Note that the colors in the different plots do not represent the same values, they are meant to illustrate the different outcomes of the grid-search optimization per experiment.

Bibliography

[1] Ryota Aihara and Yasutaka Fujimoto. Free-Space Estimation for Self-Driving System Using Millimeter Wave Radar and Convolutional Neural Network. In *Proceedings - 2019 IEEE International Conference on Mechatronics, ICM 2019*, pages 467–470. Institute of Electrical and Electronics Engineers Inc., 5 2019. ISBN 9781538669594. doi: 10.1109/ICMECH.2019.8722937.

[2] Mihael Ankerst, Markus M Breunig, Hans-peter Kriegel, and Jörg Sander. OPTICS: Ordering Points To Identify the Clustering Structure. *ACM Sigmod record*, 28(2):49–60, 1999.

[3] Dan Barnes, Matthew Gadd, Paul Murcutt, Paul Newman, and Ingmar Posner. The Oxford Radar RobotCar Dataset: A Radar Extension to the Oxford RobotCar Dataset. *arXiv preprint arXiv:1909.01300*, 2019. URL http://arxiv.org/abs/1909.01300.

[4] Tomas Borovicka, Marcel Jirina, Pavel Kordik, and Marcel Jiri. Selecting Representative Data Sets. In *Advances in Data Mining Knowledge Discovery and Applications*. InTech, 9 2012. doi: 10.5772/50787. URL http://dx.doi.org/10.5772/50787.

[5] Bosch mobility solutions. Multi purpose camera, 2019. URL https://www.bosch-mobility-solutions.com/en/products-and-services/passenger-cars-and-light-commercial-vehicles/driver-assistance-systems/lane-departure-warning/multi-purpose-camera/.

[6] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. URL http://arxiv.org/abs/1903.11027.

[7] B. D. Carlson, E. D. Evans, S. L. Wilson, and M. I.T. Lincoln. Search radar detection and track with the hough transform. *IEEE Transactions on Aerospace and Electronic Systems*, 30(1):102–108, 1994. ISSN 00189251. doi: 10.1109/7.250410.

[8] Z. J. Chong, B. Qin, T. Bandyopadhyay, M. H. Ang, E. Frazzoli, and D. Rus. Synthetic 2D LIDAR for precise vehicle localization in 3D urban environment. In *IEEE International Conference on Robotics and Automation*, pages 1554–1559, 2013. ISBN 9781467356411. doi: 10.1109/ICRA.2013.6630777.

[9] Charles K. Chui and Guanrong Chen. *Kalman filtering: With real-time applications, fifth edition*. Springer International Publishing, 1 2017. ISBN 9783319476124. doi: 10.1007/978-3-319-47612-4.

[10] Connexxion. Parkshuttle Rivium, 2020. URL https://www.connexxion.nl/nl/onze-routes/vervoersgebieden/parkshuttle-rivium.

[11] Continental Engineering Services GmbH. Standardized ARS Interface Technical Documentation. Technical report, Continental AG, 2017.

[12] Continental Engineering Services GmbH. Continental ARS 408-21 datasheet. Technical report, Continental AG, 2017. URL www.continental-industrial-sensors.com.

[13] Juergen Dickmann, Jens Klappstein, Markus Hahn, Nils Appenrodt, Hans Ludwig Bloecher, Klaudius Werber, and Alfons Sailer. Automotive radar the key technology for autonomous driving: From detection and ranging to environmental understanding. In *2016 IEEE Radar Conference, RadarConf 2016*, pages 1–6. IEEE, 2016. ISBN 9781509008636. doi: 10.1109/RADAR.2016.7485214.

[14] Richard O. Duda and Peter E. Hart. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Communications of the ACM*, 15(1):11–15, 1 1972. ISSN 15577317. doi: 10.1145/361237.361242. URL https://dl.acm.org/doi/10.1145/361237.361242.

[15] Alberto Elfes. Sonar-Based Real-World Mapping and Navigation. *IEEE Journal on Robotics and Automation*, 3(3):249–265, 1987. ISSN 08824967. doi: 10.1109/JRA.1987.1087096.

[16] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[17] Junhao Gan and Yufei Tao. DBSCAN revisited: Mis-claim, un-fixability, and approximation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 519–530, 2015. ISBN 9781450327589. doi: 10.1145/2723372.2737792.

[18] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012. ISBN 9781467312264. doi: 10.1109/CVPR.2012.6248074. URL `www.cvlibs.net/datasets/kitti`.

[19] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 9 2013. ISSN 02783649. doi: 10.1177/0278364913491297. URL `http://www.cvlibs.net/datasets/kitti`.

[20] Tilmann Giese, Jens Klappstein, Jurgen Dickmann, and Christian Wohler. Road course estimation using deep learning on radar data. In *Proceedings International Radar Symposium*. IEEE Computer Society, 8 2017. ISBN 9783736993433. doi: 10.23919/IRS.2017.8008125.

[21] Junyao Guo, Unmesh Kurup, and Mohak Shah. Is it Safe to Drive? An Overview of Factors, Challenges, and Datasets for Driveability Assessment in Autonomous Driving. *arXiv preprint arXiv:1811.11277*, 11 2018. URL `http://arxiv.org/abs/1811.11277`.

[22] Kun Yi Guo, Edward G. Hoare, Donya Jasteh, Xin Qing Sheng, and Marina Gashinova. Road Edge Recognition Using the Stripe Hough Transform From Millimeter-Wave Radar Images. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):825–833, 4 2015. ISSN 15249050. doi: 10.1109/TITS.2014.2342875.

[23] Hsieh S. Hou and Harry C. Andrews. Cubic Splines for Image Interpolation and Digital Filtering. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(6):508–517, 1978. ISSN 00963518. doi: 10.1109/TASSP.1978.1163154.

[24] Dominik Kellner, Jens Klappstein, and Klaus Dietmayer. Grid-based DBSCAN for clustering extended objects in radar data. In *2012 IEEE Intelligent Vehicles Symposium*, pages 365–370. IEEE, 2012. ISBN 9781467321198. doi: 10.1109/IVS.2012.6232167.

[25] Dominik Kellner, Michael Barjenbruch, Klaus Dietmayer, Jens Klappstein, and Jurgen Dickmann. Instantaneous lateral velocity estimation of a vehicle using Doppler radar. In *Proceedings of the 16th International Conference on Information Fusion, FUSION 2013*, pages 877–884. ISIF (Intl Society of Information Fusi, 2013. ISBN 9786058631113.

[26] Motaz Khader and Cherian Samir. An Introduction to Automotive LIDAR. Technical report, Texas Instruments, 2020.

[27] Taeryun Kim and Bongsob Song. Detection and Tracking of Road Barrier Based on Radar and Vision Sensor Fusion. *Journal of Sensors*, 2016, 2016. ISSN 16877268. doi: 10.1155/2016/1963450.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105, 2012. URL `http://code.google.com/p/cuda-convnet/`.

[29] Naveen Kumar and S. Sivasathya. Density-Based Spatial Clustering with Noise and Obstacles : A Survey. *International Journal of Computer Science and Mobile Computing*, 3(3):1004–1011, 2014.

[30] Tae Yun Lee, Vladimir Skvortsov, Myung Sik Kim, Seung Hoon Han, and Min Ho Ka. Application of W-Band FMCW Radar for Road Curvature Estimation in Poor Visibility Conditions. *IEEE Sensors Journal*, 18(13):5300–5312, 7 2018. ISSN 1530437X. doi: 10.1109/JSEN.2018.2837875.

[31] Nadav Levanon and Eli Mozeson. *Radar Signals*. John Wiley & Sons, Inc., Hoboken, 2004.

[32] Mingkang Li, Zhaofei Feng, Martin Stolz, Martin Kunert, Roman Henze, and Ferit Küçükay. High resolution radar-based occupancy grid mapping and free space detection. *VEHITS 2018 - Proceedings of the 4th International Conference on Vehicle Technology and Intelligent Transport Systems*, 2018-March (March):70–81, 2018. doi: 10.5220/0006667300700081.

[33] Mingkang Li, Martin Stolz, Zhaofei Feng, Martin Kunert, Roman Henze, and Ferit Kucukay. An Adaptive 3D Grid-Based Clustering Algorithm for Automotive High Resolution Radar Sensor. In *2018 IEEE International Conference on Vehicular Electronics and Safety, (ICVES)*, pages 1–7. IEEE, 2018. ISBN 9781538635438. doi: 10.1109/ICVES.2018.8519483.

[34] Mingkang Li, Zhaofei Feng, Martin Stolz, Martin Kunert, Roman Henze, and Ferit Küçükay. *Static environment perception based on high-resolution automotive radars*, volume 992. Springer Verlag, 2019. ISBN 9783030266325. doi: 10.1007/978-3-030-26633-2{_}10.

[35] Lifeng He, Yuyan Chao, and Kenji Suzuki. A Run-Based Two-Scan Labeling Algorithm. *IEEE Transactions on Image Processing*, 17(5):749–756, 2008. ISSN 1057-7149. doi: 10.1109/TIP.2008.919369. URL `http://ieeexplore.ieee.org/document/4472694/`.

[36] Sohee Lim, Seongwook Lee, and Seong Cheol Kim. Clustering of detected targets using DBSCAN in automotive radar systems. In *Proceedings International Radar Symposium*, pages 1–7. German Institute of Navigation - DGON, 8 2018. ISBN 9783736995451. doi: 10.23919/IRS.2018.8448228.

[37] Adrian Macaveiu and Andrei Campeanu. Automotive radar target tracking by Kalman filtering. In *2013 11th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services, TELSIKS 2013*, volume 2, pages 553–556, 2013. ISBN 9781479909025. doi: 10.1109/TELSKS.2013.6704439.

[38] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The Oxford RobotCar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 1 2017. ISSN 0278-3649. doi: 10.1177/0278364916679498. URL `http://journals.sagepub.com/doi/10.1177/0278364916679498`.

[39] Holger H. Meinel and Juergen Dickmann. Automotive radar: From its origins to future directions. *Microwave Journal*, 56(9):24–40, 2013. ISSN 01926225.

[40] Urban Meis, Wladimir Klein, and Christoph Wiedemann. A new method for robust far-distance road course estimation in advanced driver assistance systems. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 6(1):1357–1362, 2010. doi: 10.1109/ITSC.2010.5625239.

[41] Michael Meyer and Georg Kuschk. Automotive Radar Dataset for Deep Learning Based 3D Object Detection. In *2019 16th European Radar Conference (EuRAD)*, pages 129–132, 2019. ISBN 9782874870576. URL `www.astyx.net`.

[42] Motional Inc. NuScenes website general overview, 2020. URL `https://www.nuscenes.org/nuscenes`.

[43] Stephen Nellis. Velodyne aims to price new self-driving car sensor below $500 | Reuters, 2020. URL `https://www.reuters.com/article/velodyne-lidar-tech/velodyne-aims-to-price-new-self-driving-car-sensor-below-500-idINKBN27U03K`.

[44] Andras Palffy, Jiaao Dong, Julian F.P. Kooij, and Dariu M Gavrila. CNN based Road User Detection using the 3D Radar Cube. *IEEE Robotics and Automation Letters (RAL)*, 5(2):1263–1270, 2020. URL `https://github.com/tudelft-iv/RTCnet`.

[45] Michael Parker. Automotive Radar - With contributions by Ben Esposito. In Michael Parker, editor, *Digital Signal Processing 101*, chapter 20, pages 253–276. Elsevier, 2 edition, 2017. doi: 10.1016/b978-0-12-811453-7.00020-2.

[46] Mark A. Richards, James A. Scheer, and William A. Holm. *Principles of Modern Radar - Volume I - Basic Principles*. SciTech Publishing, 2010. ISBN 978-1-891121-52-4.

[47] SAE International. SAE International Releases Updated Visual Chart for Its “Levels of Driving Automation” Standard for Self-Driving Vehicles, 2021. URL http://tinyurl.com/4nayj7p7.

[48] SAE on-road Automated Vehicle Standards Committee. J3016 - Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles, 2018.

[49] Nicolas Scheiner, Nils Appenrodt, Jürgen Jurgen Dickmann, and Bernhard Sick. A Multi-Stage Clustering Framework for Automotive Radar Data. In *2019 IEEE Intelligent Transportation Systems Conference*, pages 2060–2067. Institute of Electrical and Electronics Engineers Inc., 10 2019. ISBN 9781538670248. doi: 10.1109/ITSC.2019.8916873.

[50] Matthias Schreier and Volker Willert. Robust free space detection in occupancy grid maps by methods of image analysis and dynamic B-spline contour tracking. In *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 514–521, 2012. ISBN 9781467330640. doi: 10.1109/ITSC.2012.6338636.

[51] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter H.-P Hans-Peter Kriegel, Xiaowei Xu, and Hans Peter H.-P Hans-Peter Kriegel. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Transactions on Database Systems*, 42(3), 7 2017. ISSN 15574644. doi: 10.1145/3068335. URL https://doi.org/10.1145/3068335.

[52] Ole Schumann, Markus Hahn, Jürgen Dickmann, and Christian Wöhler. Semantic Segmentation on Radar Point Clouds. *2018 21st International Conference on Information Fusion, FUSION 2018*, pages 2179–2186, 2018. doi: 10.23919/ICIF.2018.8455344.

[53] Pierre Soille. *Morphological Image Analysis*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-642-07696-1. doi: 10.1007/978-3-662-05088-0. URL http://link.springer.com/10.1007/978-3-662-05088-0.

[54] Jan Erik Stellet, Fabian Straub, Jan Schumacher, Wolfgang Branz, and J. Marius Zollner. Estimating the Process Noise Variance for Vehicle Motion Models. In *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, volume 2015-Octob, pages 1512–1519. Institute of Electrical and Electronics Engineers Inc., 10 2015. ISBN 9781467365956. doi: 10.1109/ITSC.2015.212.

[55] Kenji Suzuki. *Artificial Neural Networks - Architectures and Applications*. InTech, 1 2013. doi: 10.5772/3409.

[56] Satoshi Suzuki and Keiichi A. be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics and Image Processing*, 30(1):32–46, 4 1985. ISSN 0734189X. doi: 10.1016/0734-189X(85)90016-7.

[57] Technical University Delft. WEpod, 2020. URL https://www.tudelft.nl/en/ceg/research/stories-of-science/wepod/.

[58] Velodyne. Velodyne lidar HDL-32E datasheet. Technical report, Velodyne, 2016. URL www.velodynelidar.com.

[59] Fenglei Xu, Huan Wang, Bingwen Hu, and Mingwu Ren. Road Boundaries Detection based on Modified Occupancy Grid Map Using Millimeter-wave Radar. *Mobile Networks and Applications*, 2019. ISSN 15728153. doi: 10.1007/s11036-019-01378-5.

[60] Zhi Yan, Li Sun, Tomáš Krajník, and Yassine Ruichek. EU Long-term Dataset with Multiple Sensors for Autonomous Driving. *arXiv preprint arXiv:1909.03330*, 2019. URL https://epan-utbm.github.io/.

[61] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling. *arXiv preprint arXiv:1805.04687*, 5 2018. URL http://arxiv.org/abs/1805.04687.

[62] Yibo Zhang, Huadong Meng, Chenxi Hu, Yimin Liu, and Zicheng Du. Road and vehicle detection in highway scene for automotive FMCW antenna array radar. In *IET International Radar Conference 2015*, pages 1–5. Institution of Engineering and Technology, 2015. ISBN 978-1-78561-038-7. doi: 10.1049/cp.2015.1229. URL https://digital-library.theiet.org/content/conferences/10.1049/cp.2015.1229.