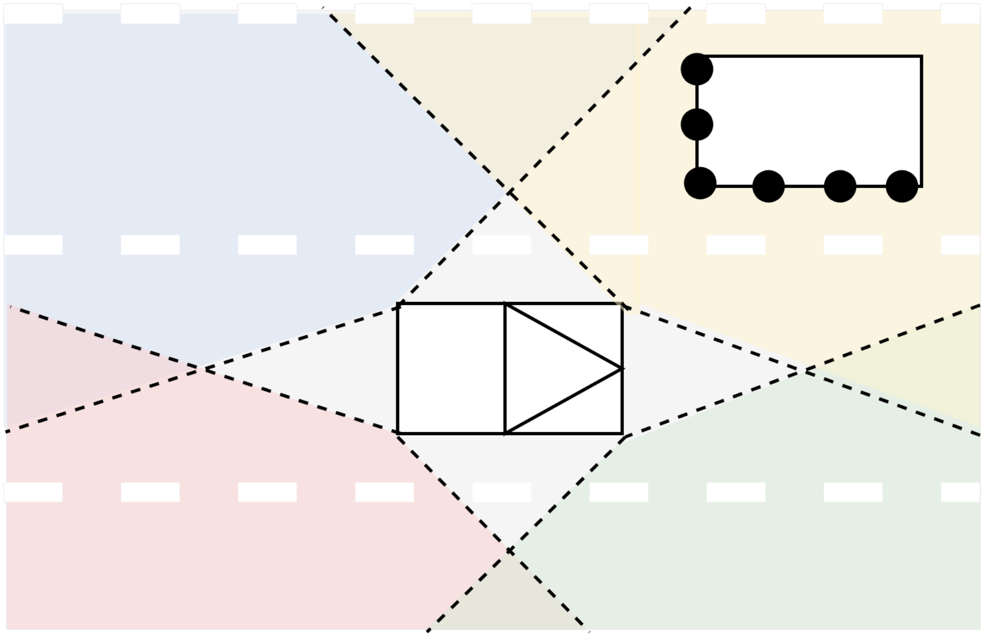
CHALMERS
UNIVERSITY OF TECHNOLOGY



Clustering, shape extraction and velocity estimation applied to radar detections

Extracting higher order information of detections from automotive radar for tracking applications

Master's thesis in Engineering Mathematics and Computational Science
Pia Lidman

Master's thesis in Systems, Control and Mechatronics
Susan Luu

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis 2018:EX024

Clustering, shape extraction and velocity estimation applied to radar detections

Extracting higher order information of detections from automotive radar for tracking applications

PIA LIDMAN
SUSAN LUU

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2018

Clustering, shape extraction and velocity estimation applied to radar detections
Extracting higher order information of detections from automotive radar for tracking applications
PIA LIDMAN, SUSAN LUU

Supervisors: Abu Sajana Rahmathullah, Zenuity AB
Daniel Svensson, Zenuity AB
Karl Granström, Department of Electrical Engineering
Examiner: Karl Granström, Department of Electrical Engineering

Master's Thesis 2018:EX024
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Visualization of four radars mounted on the corners of an ego vehicle, with the field of view for each radar marked. Detections from another vehicle are illustrated as black points.

Typeset in LaTeX
Gothenburg, Sweden 2018

Clustering, shape extraction and velocity estimation applied to radar detections
Extracting higher order information of detections from automotive radar for tracking applications
Pia Lidman
Susan Luu
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Autonomous vehicles is a topic subjected to extensive research in the automotive industry. A vital component to many features that are sought after in an autonomous vehicle is an accurate assessment of the surroundings, based on data observed from sensors mounted on the ego vehicle. If there is an object close by, it is of interest to know where that object is, what the object's velocity is, and if the object can be described by a plain geometric shape. Radar is one of the commonly used sensors in automotive applications, and it often returns multiple detections per target. This means that it is necessary to group together detections originating from the same target, i.e. to cluster the detections, in order to obtain information about the surroundings.

The objective of this master thesis is to develop an algorithm that clusters radar detections, assigns a shape to the points in each cluster, and computes a velocity estimate for each cluster. The data that were used were provided by Zenuity AB and a study of it resulted in the investigation of 12 datasets containing scenarios situated on the highway, and focused on overtakings. This report has the purpose of presenting the proposed algorithm which has two major methods – Density-based spatial clustering of applications with noise (DBSCAN), and Random sample consensus (RANSAC). The first method, DBSCAN, is for clustering detections and the second, RANSAC, is for estimating the velocity of the detections in a cluster and the corresponding geometrical shape. The estimated shapes can be a line or an L-shape to represent surrounding vehicles and guardrails.

The evaluation of the proposed algorithm was performed with manually annotated data. The results show a variation between the amount of correct clusters and over- or undersegmentation between the 12 datasets. The average sensitivity was 92.28%, which indicates that many clusters have been classified as over- or undersegmented due to 7.72% of the detections being clustered incorrectly. Regarding the velocity estimation, an average of 64.29%, of the estimated clusters received a velocity estimate where at least 80% of the detections constituting a cluster, could be described with the same velocity estimate.

Keywords: Radar, Clustering, DBSCAN, RANSAC, Velocity estimation

Acknowledgements

We would like to thank Abu Sajana Rahmathullah, Daniel Svensson, Maryam Fatemi and the rest of the Target tracking team at Zenuity for the possibility of doing this master thesis, and for all the help and supervision in the process. We would also like to thank Karl Granström at the Department of Electrical Engineering at Chalmers University of Technology for valuable input and guidance along the way.

Pia Lidman and Susan Luu, Gothenburg, June 2018

Contents

1 Introduction 1

1.1 Background and problem setup . . . 2

1.2 Thesis objective and limitations . . . 3

1.3 Outline . . . 4

2 Clustering, shape extraction and velocity estimation 5

2.1 Clustering algorithms . . . 5

2.1.1 DBSCAN . . . 6

2.1.2 Grid-based DBSCAN . . . 8

2.2 Shape extraction by model estimation . . . 10

2.3 Velocity estimation . . . 12

3 Algorithm overview and implementation 15

3.1 Algorithm overview . . . 15

3.2 Implementing RANSAC . . . 18

3.2.1 Using RANSAC to estimate horizontal lines . . . 19

3.2.2 Using RANSAC to estimate an L-shape . . . 21

3.2.3 Using RANSAC to estimate longitudinal and lateral velocity . 25

4 Data annotation and performance measures for evaluation 27

4.1 Data annotation . . . 27

4.2 Performance measures . . . 29

4.2.1 Sensitivity and precision . . . 30

4.2.2 Gaussian Wasserstein distance . . . 31

4.2.3 Oversegmentation and undersegmentation . . . 33

4.2.4 Evaluating shapes . . . 35

5 Results and Discussion 37

5.1 Tuning and parameter values . . . 37

5.1.1 Tuning parameters . . . 37

5.1.2 Parameter choices . . . 39

5.2 Evaluation results . . . 40

5.2.1 Sensitivity and precision . . . 41

5.2.2 Segmentation . . . 44

5.2.3 Shape evaluation . . . 50

5.3 Velocity estimation . . . 51

6 Future work **59**

7 Conclusion **61**

Bibliography **63**

A Tables **I**

1

Introduction

Autonomous vehicle research is a leading topic in the automotive industry. The goal with autonomous vehicles is for them to be better than manually operated ones in terms of safety and efficiency, but also in terms of environmental impact and sustainability.

Several arguments exists for why autonomous vehicles would be safer than a manually operated one. For instance, autonomous vehicles have the ability to drive routes that are non-stimulating to the driver, hence traffic related accidents due to drivers falling asleep, or becoming inattentive, could be avoided. Another benefit is further development of Intelligent Transportation Systems, which are vital to increase safety, and tackle emission and congestion problems [1]. This would help decrease fuel consumption and emissions polluting the air [2].

For an autonomous vehicle to be capable of perceiving its surroundings, it is equipped with sensors. Two key components within perception are localization and target tracking. The localization problem arises from the need for the ego vehicle to know its whereabouts, i.e., where it is located. It also needs to keep track of surrounding objects, i.e., target tracking. Different solutions regarding these problems are constantly being developed. For instance, [3] and [4] use multi-channel LiDAR with probabilistic planar surface maps and orthographic ground reflectivity map respectively, for self-localization of a vehicle in an urban environment. In [5], the authors focus on the target tracking problem and uses a geometric model-based method fused with a hypothesis trajectory model by an extended Kalman Filter.

The most common autonomous vehicle sensors providing data are camera, LiDAR and radar. A LiDAR uses laser light to measure distances and can generate a 3D map of a vehicle's surroundings. However, the algorithms can be computationally expensive since LiDAR generates large amounts of data. Camera sensors provide pictures that are processed by image recognition algorithms, and these sensors are good at classifying and distinguishing different objects. Nevertheless, it requires a large amount of processing power and it is also sensitive to conditions with poor light. Radar uses radio waves to determine range, angle and range rate to objects in an accurate manner [6]. A radar does not produce as much data as LiDAR, making radar data less computationally expensive. It is also robust against different weather conditions [6], making it more reliable than a camera.

As [6] states, radar is used in many safety systems, both on its own and with other sensors because of its properties. For a long time, radars have e.g., been an important part of airborne tracking applications. The difference between radars used in airborne systems and automotive systems respectively, is the range. In the air, distances are in terms of kilometers, while in automotive applications, they are at most a few hundreds of meters. A distance in terms of meters often means a finer radar resolution than the physical extent of a target. Because of this difference in resolution, targets in airborne applications yield at most a single detection whereas in automotive applications, multiple reflections can be registered from one object.

In this thesis, the sensor and target setup will be such that there are possibly multiple measurements per target, thus making it necessary to group together reflections by clustering. One of the limitations in the current system that we have at hand, is that no memory will be available, hence, the clustering must be done for each time instance independently. As [7] states, clustering only radar detections for one time instance, independently, is seldom done and is therefore interesting to investigate.

The following section introduces further details of radar data, and presents the problem setup of this thesis. Thereafter, the objective and limitations are presented followed by an outline of the remaining report.

1.1 Background and problem setup

As described in [8], a radar, short for Radio Detection and Ranging, is a detection system that transmits radio waves and then receives the reflection that occurs when the radio waves encounter an object. The reflected waves carry information, making it possible to determine, e.g., range, range rate, and direction (bearing angle) of the reflection. The range is obtained by examining the elapsed time between the transmission and reception of the wave, and the bearing angle is determined by estimating the angle at which the reflection was received.

This project was carried out in the target tracking team at Zenuity, a company developing software for autonomous vehicles. The radar data provided by Zenuity had been pre-processed into detections, comprising range, bearing angle and radial velocity (range rate) relative to the radar.

In total, four radars were mounted on the corners of the ego vehicle. Each radar operates at 25Hz where each radar scan, referred to as a time frame, yields a number of detections. The field of view for each radar is approximately 150°. Figure 1.1 illustrates one radar mounted at the front left corner of the ego vehicle with the field of view of the radar marked in grey. Moreover, a detection, marked in green, is illustrated with its position described by the range r and bearing angle θ. The bearing angle θ is measured from the radar's symmetry axis, shown as a dashed line. The figure also shows the coordinate system for the radar, where the origin is at the radar and the coordinate axes are labeled as x_R and y_R. The mounting angle θ^m of the radar is also illustrated. The mounting angle is the angle between x_R and the radar's symmetry axis.

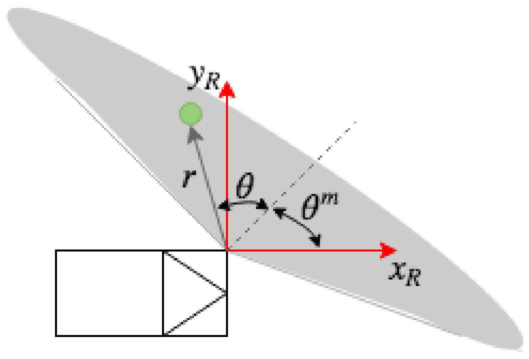


Figure 1.1: Ego vehicle with a radar mounted on the front left corner. The field of view for the radar is represented by the grey area. The green circle represents a detection with its position described by the range r and bearing angle θ, where θ is relative the radar's symmetry axis, the dashed line. The angle θ^m is the mounting angle of the radar. The coordinate axes of the radar's coordinate system are shown with the origin at the radar and the axes are labeled as x_R and y_R.

1.2 Thesis objective and limitations

The goal of this thesis was to develop an algorithm for clustering detections for one time frame at a time. Moreover, since it is important for an autonomous vehicle to accurately assess its surroundings, it is natural to want to extract as much information as possible. In this thesis, it is therefore also of interest to be able to describe clusters with shapes and to estimate their velocities.

Detections from, e.g., guardrails or nearby vehicles often form a line or an L-shape, therefore, it is of interest to be able to describe a cluster with these shapes. Figure 1.2 illustrates an example of detections forming a line and an L-shape. The estimated velocity of a cluster is relative to the radar and will be described by the velocity vector $(v_{x_R}, v_{y_R})^{\mathrm{T}}$. Here, v_{x_R} and v_{y_R} are the velocities in the longitudinal and lateral directions, respectively.

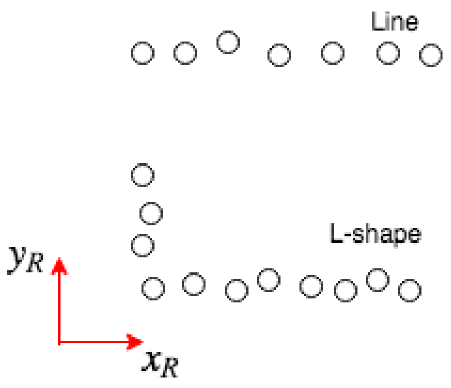


Figure 1.2: An illustration of how detections form a line and an L-shape.

The scope of this thesis is limited in the following ways. First, no memory is available, meaning that each time frame has to be treated separately and information from previous time frames are not to be considered. Second, the scenarios which will be investigated are situated on the highway around Gothenburg, and surrounding vehicles are assumed to drive in a reasonable manner, i.e., no narrow cut ins. Hence, there will be no urban environment with pedestrians or cyclists, i.e, the objects of interest are cars and trucks. Third, the four corner radars are not time synchronized and therefore, the data from each sensor have to be processed separately. Finally, there have been no focus on optimizing the algorithm.

1.3 Outline

The report is organized such that Chapter 2 gives an overview of the clustering algorithms, shape extraction and how the velocity of a cluster may be estimated. Chapter 3 summarizes the final algorithm used in this project, and describes the implementation of the algorithm, as well as, assumptions that have been made during the implementation. Chapter 4 describes the annotated data, and also the scenarios that have been examined and annotated, together with chosen evaluation methods. Chapter 5 presents obtained results from the evaluation and provides a discussion about the results. Finally, Chapter 6 talks about future work and Chapter 7 summarizes the findings in a conclusion.

2

Clustering, shape extraction and velocity estimation

This chapter describes the workings of two clustering algorithms, a method for shape extraction and a method to estimate the velocity of a cluster. The two clustering algorithms are called Density-based spatial clustering of applications with noise (DBSCAN), and Grid-based DBSCAN (GB-DBSCAN). Shape extraction is performed using an iterative method, called Random sample consensus (RANSAC), to fit a model to a set of data points. RANSAC is also used to estimate the velocity of a cluster.

2.1 Clustering algorithms

Cluster analysis aims to group together data, i.e., create clusters, according to some kind of similarity in one or more variables, e.g., distance; for more details, consult, e.g., [9]. Various methods and algorithms exist depending on the nature of the data – numerical or classes, supervised or unsupervised. In this thesis, two different unsupervised density based methods were tried, Density-based spatial clustering of applications with noise (DBSCAN) [10] and Grid-based DBSCAN (GB-DBSCAN) [11], a descendant of DBSCAN. Unlike clustering methods such as k-means [12] and k-medoids [13], DBSCAN does not require the user to guess or set a number for how many clusters that should be formed from a set of data points. Instead, DBSCAN forms a cluster if the density is high enough, and labels points as outliers otherwise.

DBSCAN has the following design parameters: a distance threshold ε, and a minimum number of points $minPts$. The two parameters together determine the density of the points in a cluster, i.e., if there are at least $minPts$ points within a distance ε from the point being evaluated, a cluster may be formed. The parameters will be presented in more detail in the following section.

An important aspect is that radar detections from a target close by, often results in many detections close to each other, while a target further away will result in fewer and more sparse detections. This is illustrated in Figure 2.1 as blue and red points, respectively. The figure shows how an ε that works well for a cluster close to the radar, does not necessarily work well for a cluster far away from the radar. This is because the detections are sparser further away, than closer to the radar. Hence, a fixed value of ε will not suffice for all ranges. Also, the number of detections further away could be fewer, meaning the number of detections needed to form a cluster close to the radar, could be too large for detections further away.

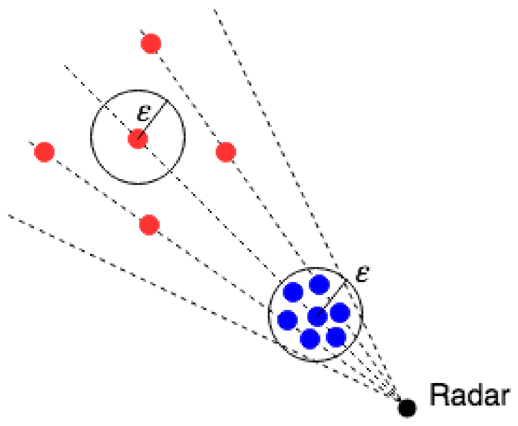


Figure 2.1: Illustration of the different density of clusters close by and far away from a radar (placed at the bottom of the cone). An ε that works well for a cluster close to the radar might not be sufficient for a cluster far away from the radar.

A clustering algorithm which takes the spread and number of detections into account, depending on their distance to the radar, is GB-DBSCAN. By considering a ratio between the range r and the bearing θ of a detection, the search area ε and the number of detections $minPts$, are adapted to the position of the detection.

2.1.1 DBSCAN

DBSCAN, first published in [10], is an unsupervised clustering algorithm which groups together data points if the density of the points is high enough. DBSCAN requires two parameters to determine the density. The first parameter is ε, describing the radial distance from a point p, that is being evaluated. The second parameter is $minPts$, which is the least number of detections that have to be within a distance ε from p, including p itself, to form a cluster. By choosing ε and $minPts$, it is then possible to decide the necessary density for a group of points to form a cluster.

To describe how DBSCAN forms clusters in more detail, the following definitions together with Figure 2.2 are required.

- A *core point* is a point with at least $minPts - 1$ points within distance ε, illustrated by the blue point in Figure 2.2.

- A point is *directly density-reachable* if it is within a distance ε from a core point. The green points in Figure 2.2 are directly density-reachable from the blue point.

- A point p_n is *density-reachable* from a point p_1 if there is a chain of points $p_1, ..., p_n$ where all points, except possibly p_n, are core points and where all points in the chain are directly density-reachable with the previous point (p_i is directly density-reachable from p_{i-1}).
- A *border point* is directly density-reachable from a core point but does not have enough points within distance ε itself, the green points in Figure 2.2.
- *Outliers* are all points that are not density-reachable from any other point, illustrated by the red point in Figure 2.2.
- The points p_1 and p_2 are *density-connected* if there exists a third point p_3 which is density-reachable from both p_1 and p_2. Looking at Figure 2.2, p_1, p_2 and p_3 corresponds to the green points and the blue point respectively.

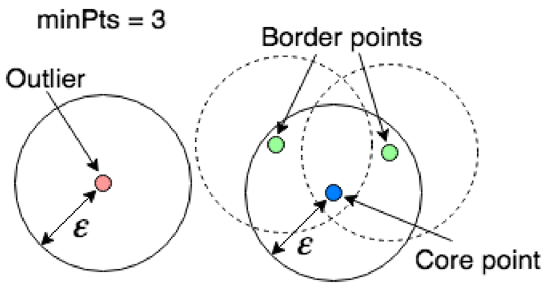


Figure 2.2: Illustration of the definitions for *minPts* = 3 and ε in the clustering algorithm DBSCAN. The blue point is a core point and forms a cluster together with the green border points. The red point is an outlier.

With these definitions, a cluster is defined as a group of points where all points are mutually density-connected and all points that are density-reachable from a point in the cluster, also belongs to the cluster. In other words, a core point outlines a cluster together with all points that are density-reachable from it. The points in a cluster that are not core points form the boundary of a cluster.

Figure 2.2 shows four points where the red point is an outlier, defined above. The blue point is the point being evaluated and because *minPts* = 3 and there are two green points falling within a distance ε of the blue point, the three points forms a cluster.

Pseudocode for DBSCAN is presented in Algorithm 1. Here, the step ”Investigate all neighbours” is done in the same manner as the investigation of the randomly chosen point p, but for all its neighbours.

```
Data: Set of data points
Result: Points divided into clusters or labeled as outliers
while There are points to be investigated do
    Choose a starting point $p$ from the set of data points;
    numberOfNeighbours = number of points inside circle with radius $\varepsilon$ around $p$;
    if numberOfNeighbours > minPts then
        Create a cluster with $p$ as core point and let the neighbours belong to the
          cluster;
        Investigate all neighbours recursively, label them as either core points or as
          border point;
        Remove all investigated points from set of points to be investigated;
    else
        Label $p$ as outlier.
    end
end
```

Algorithm 1: The algorithm for DBSCAN.

2.1.2 Grid-based DBSCAN

GB-DBSCAN, published in [11], works in the same manner as DBSCAN but does not have fixed parameters, ε and $minPts$. Instead, a polar grid is created according to the radial and angular resolution of the sensor, Δr and $\Delta\theta$ respectively. Detections in the polar coordinate system are then transformed to a corresponding cell in a Cartesian grid with the height Δr and width $\Delta\theta$. This is shown in Figure 2.3, where a detection is situated where the third radial and first angular grid lines intersects, and the corresponding cell, third in radial and first in angular direction, is marked.

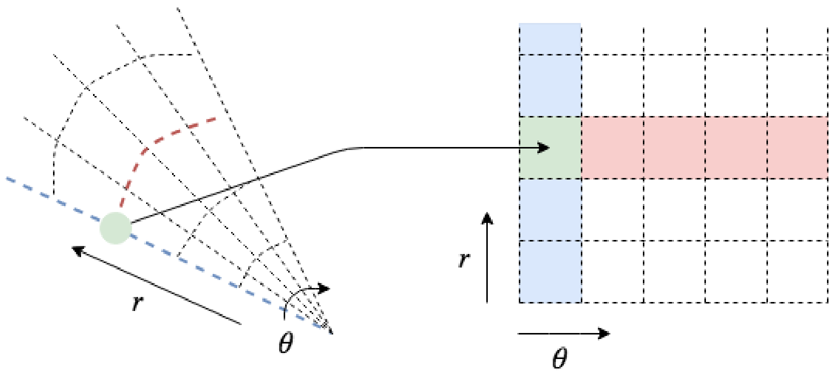


Figure 2.3: Illustration of a polar grid (left) transformed into a Cartesian grid (right). The height and width of each cell in the Cartesian grid is the radial resolution Δr and angular resolution $\Delta\theta$ respectively. A detection is situated where the third and first radial and angular grid lines intersects in the polar grid with its corresponding cell, third in radial and first in angular direction, is marked in the Cartesian grid.

Instead of looking at a circular search area with a fixed radius ε, GB-DBSCAN is able to use a more dynamic, elliptic, search area. The search area is a region calculated for each detection in which a certain number of detections ($minPts$) need to be located for a cluster to be created. In Figure 2.4, an example of an elliptic search area, with width w and height h, for a detection in the red cell, is displayed.

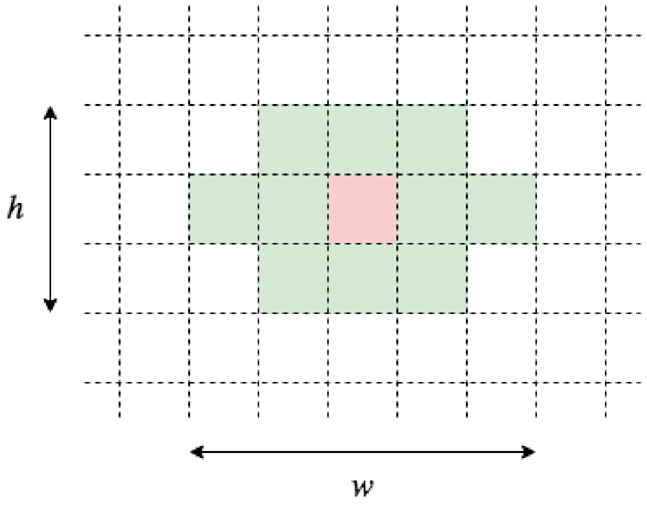


Figure 2.4: Example of an elliptic search area with width w and height h for a detection in the red box.

To determine the search area, each cell in the Cartesian grid has a value corresponding to the relationship between the range to a detection and the Cartesian grid. This value, called the c-value, is calculated as

$$c_{ij} = \frac{r_{ij}}{\Delta r} sin(\Delta\theta) \tag{2.1}$$

where i, j is the index of the cell in the Cartesian grid, r_{ij} is the radial distance to the detection that was transformed to reside in cell i, j.

The width, w, of the search area is then decided by c_{ij} and a tuning parameter g as

$$w_{ij} = \frac{g}{c_{ij}}, \tag{2.2}$$

and the height, h, of the search area is a tuning parameter.

Finally, the value of $minPts$ is decided by a suitable fraction, a tuning parameter, of the total number of cells in the search area. The algorithm is outlined as DBSCAN, see Algorithm 1.

2.2 Shape extraction by model estimation

In this project, we have used RANSAC to cluster and extract shapes in a single step. RANSAC is a framework in which it is possible to estimate a model of choice from experimental or simulated data. For a given model, e.g., a line or an L-shape, RANSAC returns each point that matches the model. This is an effective way of attaining a cluster of points that matches a line or an L-shape and at the same time label the cluster with one of those shapes, something DBSCAN can not do. The points that do not match the model are model outliers.

Previous studies have shown that RANSAC is a robust method with the potential of dealing with outlier contamination rates up to 50%, while being easy to implement [14]. It is therefore widely used, and many descendents have been developed with the aim to increase the robustness and/or modify it for different purposes [14]. One example is [15], where the authors incorporate the uncertainty of their model into RANSAC, in order to account for the effects of noise. Other descendents of RANSAC are Randomized RANSAC (RRANSAC), Maximum Likelihood Estimate Sample Consensus (MLESAC) and M-estimator Sample Consensus (MSAC) [14].

The framework of RANSAC is of the type hypothesize-and-verify [15]. A certain number of points, n, are randomly drawn from all input points. Together, they form a model, chosen by the user. This is the hypothesis step where the hypothesis is the model formed by the current n points. The verification step then lies within evaluating if remaining points support the hypothesis.

To aid the description of RANSAC in more detail, the following definitions are needed [16]:

- An *inlier* is a point that supports the hypothesis.
- An *outlier* is a point that does not support the hypothesis.
- The *error tolerance* is a design parameter that decides if a point is an inlier or outlier.
- The *threshold tolerance* is a design parameter that essentially decides if a hypothesis is good enough. A common threshold tolerance is to demand the percentage of inliers to be a certain amount.

To further aid the description, an example to estimate a straight line (our chosen model) will be examined while the hypothesize-and-verify steps are explained. The outline for this example of RANSAC is shown in Algorithm 2.

```
Data: Set of data points
Result: The line best fitted to the dataset
for m iterations do
    Draw two random points from the set of data points;
    Form a line between the two points;
    Set all points within orthogonal distance $d$ from the line to inliers;
    Calculate the percentage of inliers;
    if Percentage of inliers $>$ threshold tolerance then
        Save model and percentage of inliers;
    end
end
Return model with highest percentage of inliers;
```

Algorithm 2: The algorithm for RANSAC when estimating a straight line.

Consider the equation for a straight line,

$$y = kx + m,$$

where the slope k and the intersection point m should be decided upon to obtain a hypothesis. To obtain them, two points are needed, i.e., $n = 2$. Figure 2.5 illustrates two red points that have been randomly drawn. A blue line has formed between them, yielding the sought after k and m – the hypothesis step is finished.

To verify, an error tolerance is needed. The orthogonal distance d, from the line, is used for this purpose. The verification is then to calculate the orthogonal distance from all remaining points to the line. Those with a distance less than or equal to d are inliers, illustrated as green points, and those with a larger distance are outliers, illustrated as white points. Finally, the percentage of inliers are calculated to see if the hypothesis is good enough.

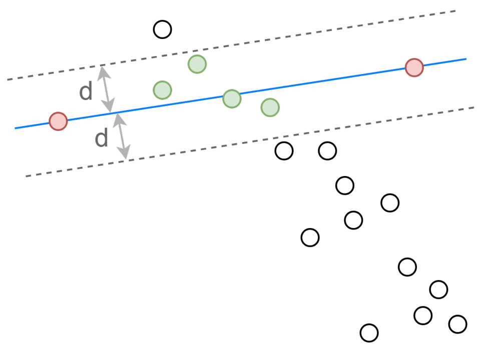


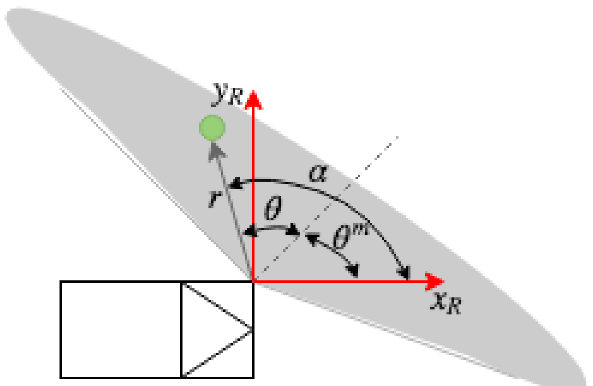
Figure 2.5: Example of RANSAC used to estimate a line to data points. The red points are two randomly drawn points, forming a line. The green points mark the inliers and remaining points are outliers.

The RANSAC algorithm is iterative: draw n points, form a hypothesis and verify it. The user sets a fixed number of iterations, and could either let all iterations run and then pick the best model (highest percentage of inliers), or terminate as soon as a model has got enough inliers. If an acceptable model has not been found after a predetermined number of iterations, either the model yielding the most inliers or no solution, may be returned [16]. Thus, RANSAC has three main parameters that needs to be set, the error threshold to separate inliers from outliers, a threshold tolerance for what is considered an acceptable model, and how many iterations to run [16].

2.3 Velocity estimation

One of the overall objectives in the autonomous vehicle industry is to gain as much information as possible about the surroundings of the ego vehicle. Estimation of the velocity of surrounding objects are therefore interesting. The theory and assumptions made to estimate the velocity vector $\mathbf{v} = (v_{x_R}, v_{y_R})^{\mathrm{T}}$ relative to the radar for a cluster, will be explained in this section. Here, v_{x_R} and v_{y_R} are the velocities in the longitudinal and lateral directions, respectively. In this thesis the motion of the ego vehicle has not been compensated for when estimating the cluster velocity.

In Figure 2.6, an angle α is introduced to be $\theta + \theta^m$, where θ is the bearing angle and θ^m is the mounting angle of the radar. Remaining notations (field of view, symmetry axis etc) are the same as in Figure 1.1.



.

Figure 2.6: Introducing the angle α to be the angle to be $\theta + \theta^m$. The coordinate axes and remaining notations used, (e.g. field of view, symmetry axis etc.) are the same as in Figure 1.1

As in [17], we have assumed that all detections in a cluster move according to a linear motion model. This means that the direction of the velocity vector $\mathbf{v} = (v_{x_R}, v_{y_R})^{\mathrm{T}}$ of a single detection, is the same for each detection in the cluster. For each of N detections, there is a range rate, $v_{r,i}$, which depends on the angle α_i, where the index

$i = 1, \dots, N$. The relationships between $v_{r,i}$ and α_i is described by the following system of equations,

$$\begin{bmatrix} v_{r,1} \\ v_{r,2} \\ \vdots \\ v_{r,N} \end{bmatrix} = \begin{bmatrix} \cos(\alpha_1) & \sin(\alpha_1) \\ \cos(\alpha_2) & \sin(\alpha_1) \\ \vdots & \vdots \\ \cos(\alpha_N) & \sin(\alpha_N) \end{bmatrix} \begin{bmatrix} v_{x_R} \\ v_{y_R} \end{bmatrix}. \tag{2.3}$$

The velocity profile (2.3) has two parameters: v_{x_R} and v_{y_R}. The profile is a linear system and it is over-determined for a cluster containing more than two detections. The velocity profile is of the type

$$Ax = b,$$

and when A is not a square matrix, it does not have an inverse. It is then possible to solve the system of equations by using least square,

$$x = (A^T A)^{-1} A^T b.$$

For Eq. (2.3), the matrix with cosines and sines correspond to A. Detections that are close to each other will have similar angles α, meaning that the rows of the matrix may be similar. This makes it difficult to invert $A^T A$, because it will be singular, or close to singular.

3 Algorithm overview and implementation

This chapter presents an overview of the proposed algorithm, followed by details of the implementation for the different models that have been estimated with RANSAC. Assumptions that have been made will be explained and motivated.

3.1 Algorithm overview

The proposed algorithm has three major blocks. The first and second blocks estimate guardrails and an L-shape, respectively, in specific search areas. The third block uses DBSCAN to obtain coarse clusters and also fits a line or an L-shape to each cluster under certain conditions. The final step in the algorithm is clustering with DBSCAN. Starting with all detections in a time frame, within each block, the approach of the proposed algorithm is to decrease the number of detections that are passed to the next block.

Figure 3.1 presents a flowchart of the different steps in the algorithm. The three different blocks are the three first rows in the figure, respectively. Let D bet the set with all detections in a time frame, and R a set of detections that belong to a certain cluster. The overall pattern that the model estimation follows with each block is that, a subset S is created, $S \subseteq D$. The subset is obtained according to a search area, described in Section 3.2.1 and Section 3.2.2, or by using DBSCAN. It is used as input to RANSAC and if RANSAC returns a model, a subset I is created, $I \subseteq S$. The subset I contains inliers to the estimated model and is used to update D and R by, $D = D \setminus I$ and $R = R \cup I$.

Figure 3.2 shows an illustration of an example of a time frame. The different clusters around the ego vehicle are illustrated with different colors where the blue detections form an L-shape and the green and yellow detections, each form a guardrail. Each numbered step (1-8) in Figure 3.1 will briefly be described below, using the example in Figure 3.2, before further implementation details are presented in Section 3.2.1 and 3.2.2.

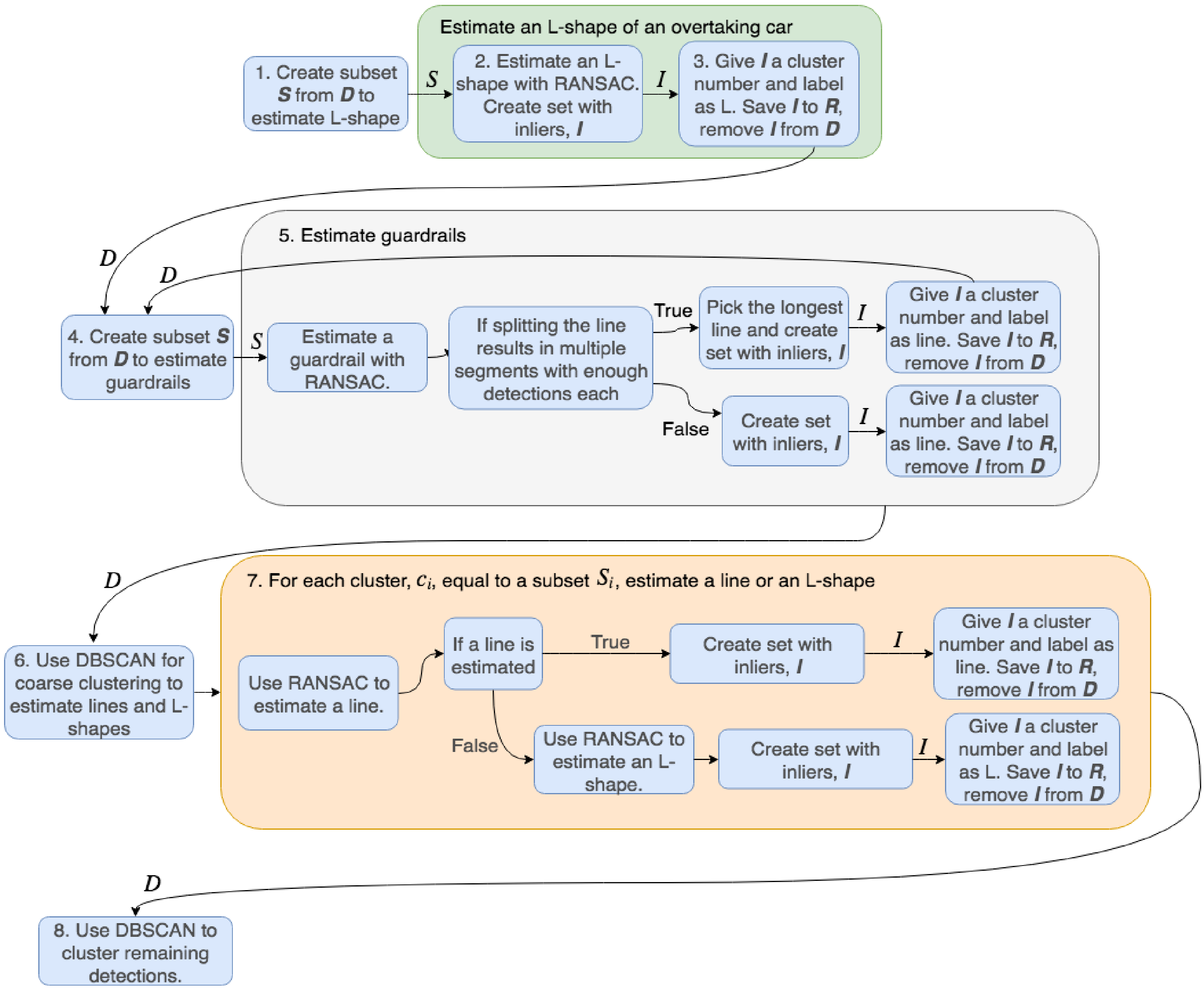

Estimate an L-shape of an overtaking car
1. Create subset S from D to estimate L-shape
S
2. Estimate an L-shape with RANSAC. Create set with inliers, I
I
3. Give I a cluster number and label as L. Save I to R, remove I from D
D
D
5. Estimate guardrails
4. Create subset S from D to estimate guardrails
S
Estimate a guardrail with RANSAC.
If splitting the line results in multiple segments with enough detections each
True
Pick the longest line and create set with inliers, I
I
Give I a cluster number and label as line. Save I to R, remove I from D
False
Create set with inliers, I
I
Give I a cluster number and label as line. Save I to R, remove I from D
D
7. For each cluster, c_i, equal to a subset S_i, estimate a line or an L-shape
6. Use DBSCAN for coarse clustering to estimate lines and L-shapes
Use RANSAC to estimate a line.
If a line is estimated
True
Create set with inliers, I
I
Give I a cluster number and label as line. Save I to R, remove I from D
False
Use RANSAC to estimate an L-shape.
Create set with inliers, I
I
Give I a cluster number and label as L. Save I to R, remove I from D
D
8. Use DBSCAN to cluster remaining detections.

Figure 3.1: An overview of the proposed algorithm. For the first step, given a set D with all detections in a time frame, and a set R with the clustering results, the steps 1-3, 4-5 and 6-7 follow the same pattern. Each block of the algorithm starts by creating a subset S, or subsets S_i in step 6, from D. Then, S is used as input to RANSAC to estimate a model. Another subset $I \subseteq S$ is then created, containing inliers to the model. Finally, D and R are updated by $D = D \setminus I$ and $R = R \cup I$. The final step of the algorithm is clustering of the remaining detections in D using DBSCAN.

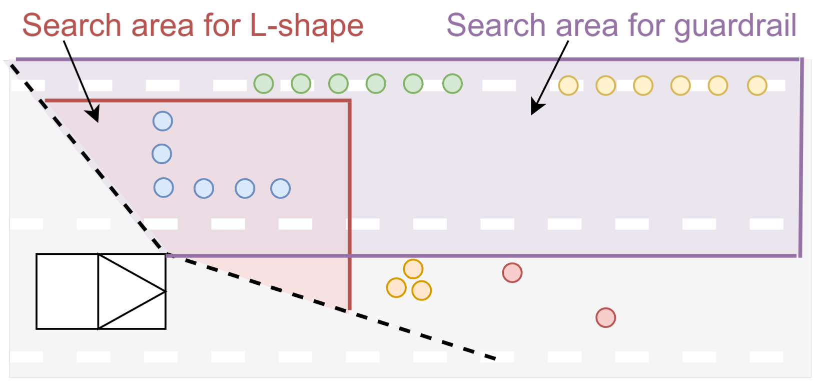


Figure 3.2: Illustration of an example of a time frame. The ego vehicle is illustrated in the lower left corner and four different clusters are illustrated as blue, green, yellow and orange detections, respectively. The red detections are outliers. The blue detections form an L-shape and the green and yellow detections form one guardrail each. The field of view is marked with black dashed lines, and the search area for L-shapes and for guardrails are marked with red and purple, respectively.

1. Extract detections to estimate an L-shape from an overtaking vehicle
Starting with D, create a subset $S \subseteq D$ containing detections within a certain search area, described in detail in Section 3.2.1. For the example in Figure 3.2, this step extracts the blue detections in the red search area. The subset S is used as input to RANSAC to estimate an L-shape.

2. Estimate L-shape
RANSAC is used to estimate an L-shape with the subset S obtained in step 1 as input. This is described in detail in Section 3.2.2.

3. Update D and R
If an L-shape was estimated in step 2, a subset $I \subset S$ is created, where I contains inliers to the L-shape. The update of D and R is made by

$$\begin{aligned} D &= D \setminus I \\ R &= R \cup I. \end{aligned} \tag{3.1}$$

For the example in Figure 3.2, all blue detections are inliers and removed from D.

4. Extract detections to estimate guardrails
Step 1 is repeated in order to obtain a subset to use as input to RANSAC to estimate a guardrail, but with a different search area. This step extracts the green and yellow detections in the purple search area in Figure 3.2.

5. Estimate guardrail
RANSAC is used to estimate a guardrail with the S obtained in step 4 as input. It is desirable to estimate lines where the detections form a line without any gaps; this will be explained further in Section 3.2.1. Therefore, an estimated line is split into

line segments if there are any gaps. The longest line segment is chosen, I is created from it and D and R are updated according to (3.1). If there are m number of other line segments, each with enough detections, step 4 and the estimation of a guardrail is run m times again.

For the example in Figure 3.2, the green and yellow detections are estimated as one line. The line is then split between the green and yellow detections, as there is a gap in the line. If the green detections are chosen, (i.e., they form the longest line) they constitute I, are removed from D and saved to R. The yellow detections are enough for a second estimation of a guardrail. After the estimation, they also constitute a subset I, are removed from D and saved to R.

6. Coarse clustering
Coarse clustering is done to obtain rough clusters, used for estimating lines and L-shapes. DBSCAN is used with D as input. Each cluster c_i, that is obtained from DBSCAN, is set to be a subset S_i. For the example in Figure 3.2, the orange cluster is a coarse cluster and the two red detections are outliers.

7. Estimate lines
For each S_i obtained from step 6, RANSAC is used to estimate a line. If a line is estimated, a subset of the inliers, $I \subseteq S_i$, is created. It is given a cluster number, labelled as line and D and R are updated according to (3.1). If no line estimate was obtained, RANSAC is used to estimate an L-shape. For an estimated L-shape, $I \subseteq S_i$ is created, given a cluster number, labelled as L-shape and D and R are updated according to (3.1). For the single coarse cluster, denoted by the orange detections, in Figure 3.2, neither a line nor an L-shape is estimated.

8. Last clustering step
The set D is used as input to DBSCAN for a last clustering step. This step is done in order to catch detections that, e.g., were a part of a cluster in the coarse clustering but might have become outliers to a line or an L-shape. For the example in Figure 3.2, the last clustering step returns the orange and red detections as a cluster and outliers, respectively.

3.2 Implementing RANSAC

This section will present how RANSAC is used to estimate lines, L-shapes and the velocity vector $(v_{x_R}, v_{y_R})^{\mathrm{T}}$ for a cluster. The implementation follows the framework laid out in Section 2.2. The general outline for each section is to describe the constraints and their associated parameters added in this thesis along with a motivation, followed by a pseudocode to illustrate where in the RANSAC algorithm, the constraints were used. The chosen parameter values will be presented in Chapter 5, Section 5.1.

3.2.1 Using RANSAC to estimate horizontal lines

The scope of this thesis covers scenarios on the highway. Apart from vehicles, such as trucks and cars, surrounding the ego vehicle, there are also guardrails. Detections from guardrails can be represented by a line, which was estimated using RANSAC. To increase the probability that an estimated line corresponds to a guardrail, studies of the data have been made in order to see what properties may be exploited. These will be presented and motivated in this section.

The following observations were made when studying the detections reflected from guardrails:

- It was not always clear that the detections forming a line were from a guardrail by looking at the length of the lines. Sometimes, they were clear long lines and sometimes not.
- Detections from a guardrail did not always form a clear straight line.
- Detections from guardrails could appear as the above mentioned types, but with gaps in between, i.e., the detections did not form a continuous line.
- Lines where it was clear that the detections came from a guardrail, were located at a certain distance from the ego vehicle. Detections located on the other side of this line could be from vehicles driving in the opposite direction.
- Detections forming a line were often aligned with the ego vehicle since we only consider highway scenarios.

Figure 3.3 illustrates three examples based on these observations. The first scenario shows blue and green detections forming lines that are longer and shorter than 20m, respectively. The second scenario shows two lines that are both longer than 20m. Finally, the third scenario shows how the blue detections form a sufficiently long line, while the green detections does not form a line. All scenarios presents a gap, as the third observation states.

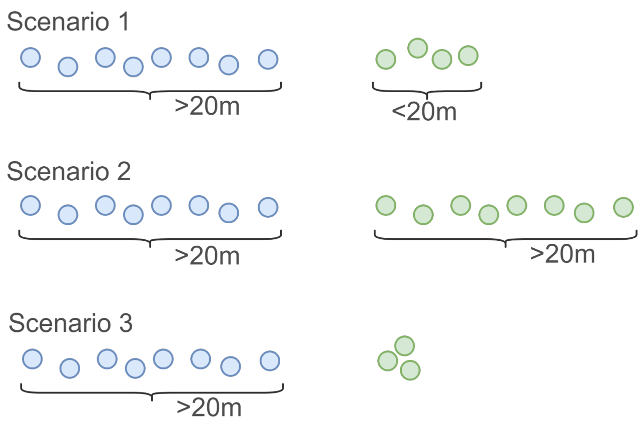


Figure 3.3: Illustration of three examples of how detections may appear from a guardrail. In the first scenario the blue and green detections form lines that are longer and shorter than 20m, respectively. The second scenario shows detections that form two lines that are longer than 20m. In the third scenario, the blue detections form a sufficiently long line, while the green detections does not form a line. There is a gap between the blue and green detections in all scenarios.

The above observations were characterized by a set of parameters, which are presented in Table 3.1. They will be motivated and explained below.

Constraint	**Parameter**
1	The search area for a guardrail was $(SA)_{guard}$ laterally in positive direction
2	Minimum length for a guardrail, l_{guard}
3	Minimum number of detections for RANSAC to find a guardrail, N_{guard}
4	Minimum number of detections for RANSAC to find shorter lines, N_{line}
5	Minimum length for lines shorter than a guardrail, l_{line}
6	Maximum allowed absolute value, k, of a line's slope
7	Maximum allowed longitudinal distance, T, between detections
8	Number of iterations, Num_{line}^{it}
9	Required percentage of inliers P_{line}

Table 3.1: Table of constraint numbers and their associated parameters that were used to estimate horizontal lines, and their respective values.

Constraint 1 comes from observing guardrails at a certain distance from the ego vehicle. The constraints number 2 and 3 says that if an estimated line is at least $l_{guard} = 20$m long, it is most likely a guardrail and that N_{guard} number of detections are required for RANSAC to estimate a guardrail. For estimation of lines that are shorter than l_{guard}, DBSCAN was used to obtain coarse clusters. Each cluster was used as input to RANSAC if they had at least N_{line} number of detections, constraint number 4, and were considered a line if the length was at least l_{line}, constraint number 5. Constraint number 6 constrains the absolute value of the slope of the line in order to assure that the line is horizontal. Constraint 7 prevents gaps by examining the median of the longitudinal distance between detections constituting a guardrail. The maximum allowed distance between detections is a threshold, T, times the median. Finally, constraints 8 and 9 are the remaining design parameters that were used; the number of iterations, Num_{line}^{it}, and the required percentage of inliers, P_{line}.

Algorithm 3 shows a pseudocode for estimating a guardrail or a line, where *preClustered* is a boolean that tells the algorithm if the input is a coarse cluster (step 6 in the algorithm overview, Figure 3.1).

```
Data: Set of data points
Result: The line best fitted to the dataset
for $Num_{line}^{it}$ iterations do
    Draw two random points from the set of data points;
    Form a line between the two points;
    Set all points within orthogonal distance $d$ from the line to inliers;
    if preClustered then
        Split line at gaps;
        Choose longest line and set those detections to inliers;
        If there are other line segments that could potentially be guardrails, save
          and let RANSAC return them;
        if abs(Slope of line) $\leq k$ and length of line $\geq l_{guard}$ then
            Calculate the number of inliers;
            Save model and number of inliers;
        end
    else
        Calculate the percentage of inliers;
        if abs(Slope of line) $\leq k$ and length of line $\geq l_{line}$ ...
          and percentage of inliers $\geq P_{line}$ then
            Save model and percentage of inliers;
        end
    end
end
Choose model with most inliers for guardrail or highest percentage of inliers for
  line;
```

Algorithm 3: The pseudocode for RANSAC to estimate guardrails and lines.

3.2.2 Using RANSAC to estimate an L-shape

When the target vehicles are close to the ego vehicle, within a range of approximately 13m, detections often form an L-shape. Such L-shapes are estimated with RANSAC. This section will present assumptions that have been made and what their corresponding constraints look like. The outline of the algorithm is shown in Algorithm 4.

In order to estimate an L-shape with RANSAC, three detections are drawn at random. These points are used to estimate the two straight lines that form the L. Figure 3.4 illustrates three drawn detections in red. They will here be referred to as the horizontal point (*hPoint*), the vertical point (*vPoint*) and the corner point (*cPoint*). The horizontal and vertical points form the horizontal and vertical lines of the L together with the corner point, respectively.

Further, the error threshold on the orthogonal distance from the horizontal and vertical lines was set as d. This threshold results in two boxes of thickness $2d$, with the *cPoint* in the overlapping region. The length of the boxes around the horizontal and vertical lines are denoted as *hLength* and *vLength*, respectively, Figure 3.4.

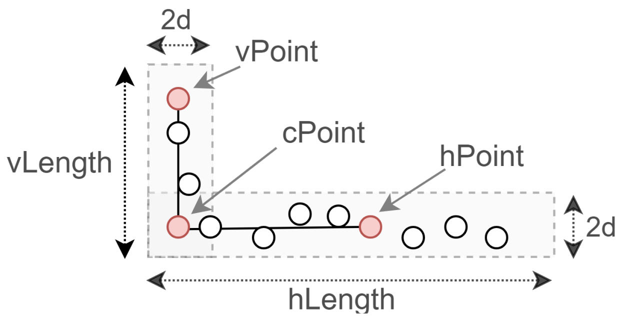


Figure 3.4: Illustration of detections forming an L-shape. The three red detections are randomly chosen to outline the L-shape. The detection at the corner of the L-shape is the corner point, denoted *cPoint*, and the detections at the vertical line and the horizontal line are denoted *vPoint* and *hPoint*, respectively. The length of the boxes around the horizontal and vertical lines are denoted as *hLength* and *vLength*, respectively. The width of the boxes is set to $2d$.

The constraints used to estimate L-shapes are listed in Table 3.2, and motivated and explained below.

Constraint	**Parameter**
1	The search area for an L-shape was $(SA)_L^{long}$ and $(SA)_L^{lat}$ in positive, longitudinal and lateral direction, respectively
2	Number of detections, N_L, required for RANSAC to find an L-shape
3	Maximum allowed absolute value, k, of the slope of the horizontal line
4	Minimum and maximum longitudinal distances, $hMin$ and $hMax$, respectively, between *cPoint* and *hPoint*
5	Minimum and maximum lateral distances, $vMin$ and $vMax$, respectively, between *cPoint* and *vPoint*
6	The allowed deviation of the angle between the horizontal and vertical line, $\Delta\varphi$
7	Number of iterations, Num_L^{it}

Table 3.2: Table of constraint numbers and their associated parameters that were used to estimate L-shapes.

Constraint 1 comes from observing where L-shapes appeared relative to the ego vehicle. This constraint is only used in step 1, in the proposed algorithm. The search area is illustrated in Figure 3.5.

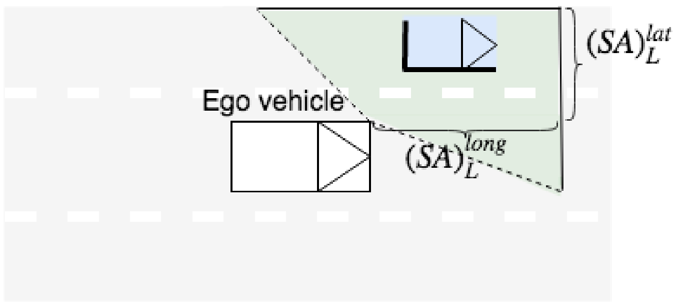


Figure 3.5: Illustration of the search area marked in green, when estimating an L-shape with RANSAC. The illustration is for the front left radar. The maximum, positive, longitudinal and lateral distance between the radar and a detection are $(SA)_L^{long}$ and $(SA)_L^{lat}$, respectively. The area is also cut off by the field of view of the radar. Only detections within this area are used as input to RANSAC.

RANSAC draws three detections to estimate an L-shape, therefore, constraint 2 is needed. L-shapes are assumed to appear like L's with their corner towards the ego vehicle, shown in Figure 3.6. This assumption is made because we are on a highway, for situations where there are no queues. As with the case of estimating a horizontal line with RANSAC, the slope of the horizontal line in the L-shape is also constrained by a maximum value k, constraint 3.

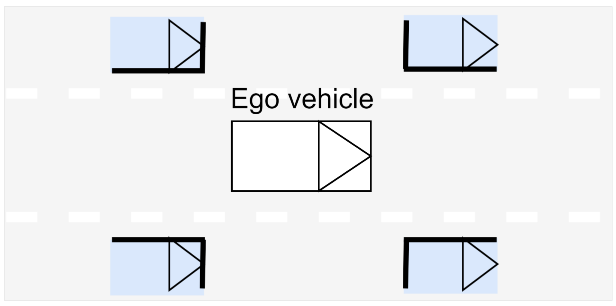


Figure 3.6: Possible appearances of L-shapes. The L-shapes are assumed to appear as L's with corners pointed towards the ego vehicle as illustrated by the black edges of the blue vehicles around the ego vehicle.

The lower limits, $hMin$ and $vMin$, in constraint 4 and 5 prevent RANSAC from estimating a corner where the length of the horizontal and vertical lines, respectively, are too short. If they are too short, undesired L-shapes as the one shown to the left in Figure 3.7 may be estimated. The upper limits, $hMax$ and $vMax$, in constraint 4 and 5 were chosen by considering the typical dimensions of cars and trucks. They also prevent RANSAC from estimating a corner where the lengths of the horizontal and vertical lines, respectively, are too long. This is illustrated to the left in Figure 3.7, for the vertical line.

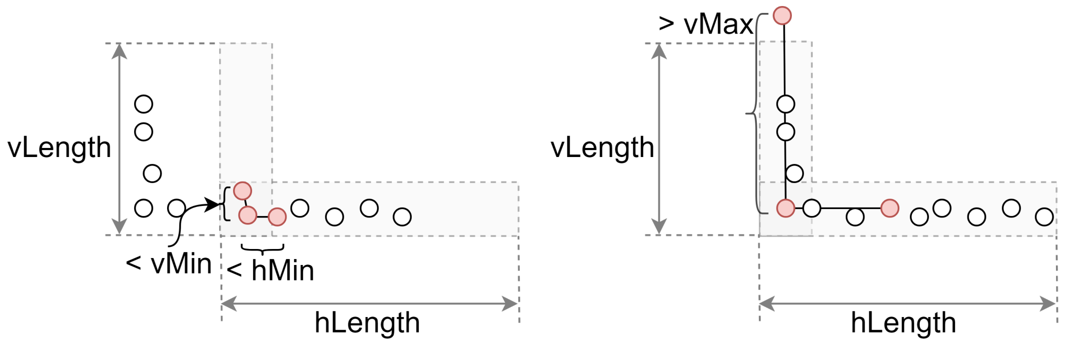


Figure 3.7: Illustration of two undesired scenarios. The left figure shows an incorrectly chosen corner of an L-shape where the longitudinal and lateral distance for the horizontal and vertical lines, respectively, are shorter than $hMin$ and $vMin$, respectively. The right figure shows a scenario where a detection that is too far away has been chosen as *vPoint*, violating the threshold *vLength*. The chosen *vPoint* should be an outlier.

For an L-shape, the horizontal and vertical lines have to be close to perpendicular. Constraint number 6 ensures that the two lines are almost perpendicular to each other by allowing a maximum deviation of the angle, $\Delta\varphi$. Finally, constraint number 7 is the remaining design parameter that was used, the number of iterations, Num_L^{it}.

```
Data: Set of data points
Result: The L best fitted to the dataset
for $Num_L^{it}$ iterations do
    Draw three random points from the set of data points;
    Set the three points to vPoint, corner point and hPoint depending on the
      location of the points;
    Form a vertical line between vPoint and corner point and a horizontal line
      between hPoint and corner point;
    if abs(Slope of the horizontal line) > $k$ then
        Continue to next iteration;
    end
    if abs($\varphi - 90°$) > $\Delta\varphi$ then
        Continue to next iteration;
    end
    if length(vertical line)< $vMin$ or length(vertical line)> $vMax$ or
      length(horizontal line)< $hMin$ or length(horizontal line)> $hMax$ then
        Continue to next iteration;
    end
    Set all points within orthogonal distance $d$ from either of the two lines to
      inliers;
    Calculate the number of inliers;
    Save model and number of inliers;
end
Choose model with most inliers;
```

Algorithm 4: The pseudocode for RANSAC to estimate an L-shape.

3.2.3 Using RANSAC to estimate longitudinal and lateral velocity

To estimate the velocity of a cluster, RANSAC is used with the model described by Eq. (2.3). The assumption that all detections travel in a linear motion, suits the considered highway scenarios well. The parameters to estimate are v_{x_R} and v_{y_R}. The number of randomly drawn detections is N_{vel} and the number of iterations is Num_{vel}^{it}.

Introducing the following notations for Eq. (2.3),

$$\underbrace{\begin{bmatrix} v_{r,1} \\ v_{r,2} \\ \vdots \\ v_{r,N} \end{bmatrix}}_{\mathbf{v}_r} = \underbrace{\begin{bmatrix} \cos(\alpha_1) & \sin(\alpha_1) \\ \cos(\alpha_2) & \sin(\alpha_1) \\ \vdots & \vdots \\ \cos(\alpha_N) & \sin(\alpha_N) \end{bmatrix}}_{A_{N_{vel}}} \underbrace{\begin{bmatrix} v_{x_R} \\ v_{y_R} \end{bmatrix}}_{\mathbf{v}},$$

where the subscript stands for the number of detections that is used. Using $N_{vel} > 2$ detections, the estimate $\hat{\mathbf{v}} = (\hat{v}_{x_R}, \hat{v}_{y_R})^{\mathrm{T}}$ is obtained by least square according to

$$\hat{\mathbf{v}} = (A_{N_{vel}}^{\mathrm{T}} A_{N_{vel}})^{-1} A_{N_{vel}}^{\mathrm{T}} \mathbf{v}_r. \tag{3.2}$$

By using this estimate, it is then possible to use the same equation to estimate the range rate for all detections in the cluster that is being considered, $\hat{\mathbf{v}}_r$. The error is calculated as

$$\frac{|\hat{\mathbf{v}}_r - \mathbf{v}_r|}{\hat{\mathbf{v}}_r}, \tag{3.3}$$

and the threshold for this error is denoted e_{vel}. The threshold tolerance that was used was the percentage of inliers, which was required to be P_{vel}^1. Algorithm 5 outlines this method.

If RANSAC terminated after Num_{vel}^{it} iterations without succeeding to estimate a velocity, it was run again but with a new, lower, threshold tolerance P_{vel}^2. Detections that were inliers in this case were saved along with the estimated velocity while the remaining detections were run once again with the initial threshold tolerance, P_{vel}^1. This procedure was done in order to catch situations where a cluster contained detections from wheel houses. For such clusters, the velocity from the wheel houses are higher than the detections originating from the body of the vehicle, and the cluster is considered to have two velocities. Clusters that obtain a velocity estimate on the first try, with only P_{vel}^1 as a threshold, are considered to have one velocity.

```
**Data:** Set of detections
**Result:** $(v_{x_R}, v_{y_R})^{\mathrm{T}}$ best fitted to the dataset
for $Num_{vel}^{it}$ iterations do
    Draw $N_{vel}$ number of detections at random from the set of detections;
    Solve for $(\hat{v}_{x_R}, \hat{v}_{y_R})^{\mathrm{T}}$ with least square, Eq. (3.2);
    Use $(\hat{v}_{x_R}, \hat{v}_{y_R})^{\mathrm{T}}$ in Eq. (2.3) and calculate $\hat{v}_r$ for all detections;
    Calculate the number of inliers where $|\hat{v}_r - v_r|/|\hat{v}_r| < e_{vel}$;
    if Percentage of inliers $\geq P_{vel}^{1}$ then
        Save $(\hat{v}_{x_R}, \hat{v}_{y_R})^{\mathrm{T}}$ and the percentage of inliers;
    end
end
Set $(v_{x_R}, v_{y_R})^{\mathrm{T}}$ to the $(\hat{v}_{x_R}, \hat{v}_{y_R})^{\mathrm{T}}$ with highest percentage of inliers;
```

Algorithm 5: The pseudocode for RANSAC used to estimate velocities.

4
Data annotation and performance measures for evaluation

This chapter presents how the data annotation was carried out, followed by the theory of four chosen performance measures: sensitivity, precision, oversegmentation and undersegmentation. A method for how to choose the cluster that corresponds to true positive (a measure needed in sensitivity and precision), using Gaussian Wasserstein distance is also presented, as well as the chosen method for shape evaluation. Regarding the velocity estimation, a ground truth was not available and an annotation has not been possible. Hence, no performance measure will be used for the obtained velocity estimates. The velocity estimates will instead be analyzed and discussed in Section 5.3.

4.1 Data annotation

The data used in this thesis had no ground truth. Therefore, in order to execute an evaluation of the proposed algorithm, it was necessary to annotate data to use as a reference.

Four log files were chosen for the annotation, each containing data from four radars, i.e., there were in total 16 radars. The log files enclose different driving scenarios. Time frames from 12 of the 16 radars were chosen by considering two radars from the first and third file respectively and four radars from the second and fourth file respectively. This was done to acquire a variation among the annotated data. Thereby, 12 different datasets were created with 472 time frames in total. In Table 4.1, the 12 datasets are described by which log file they came from, the number of time frames they contain, their mounting position on the ego vehicle and the driving scenarios they represent. One of the driving scenarios in the datasets included driving in a queue with other vehicles on both sides of the ego vehicle. This scenario was chosen to be able to look at data where surrounding vehicles cuts in behind or in front of the ego vehicle, changing lanes. Remaining scenarios captures overtaking by, and of, the ego vehicle, both with a guardrail close by and without.

Data-set	***Log file (1-4)***	***Number of time frames***	***Mounting position of radar***	***Scenario***
1	1	56	Rear left	Two cars in left lane overtaking ego vehicle. A guardrail to the left of the overtaking cars.
2	1	52	Front left	Two cars in left lane overtaking ego vehicle. A guardrail to the left of the overtaking cars.
3	2	31	Rear left	Two cars overtake ego vehicle.
4	2	25	Rear right	Ego vehicle overtakes two trucks.
5	2	25	Front left	Two cars overtake ego vehicle.
6	2	33	Front right	Ego vehicle overtakes two trucks.
7	3	80	Rear right	A car overtakes ego vehicle.
8	3	47	Front right	A car overtakes ego vehicle.
9	4	28	Rear left	Queue in three lanes. Ego vehicle in the middle overtakes cars in the left lane.
10	4	16	Rear right	Queue in three lanes. A truck in the right lane overtakes the ego vehicle.
11	4	28	Front left	Queue in three lanes, ego vehicle in the middle overtakes cars in the left lane.
12	4	64	Front right	Queue in three lanes. A car and then a truck in the right lane overtakes the ego vehicle.

Table 4.1: List of all datasets and their corresponding log file, the number of time frames they contain, their mounting position on the ego vehicle and the scenarios they covers.

To annotate, DBSCAN was first used, with $\varepsilon = 4.3$ and $minPts = 2$, to obtain rough clusters. The results were manually examined and adjusted if a cluster was deemed to be wrong, e.g., if one detection belonging to a cluster should be an outlier, or one cluster should be represented by two clusters. The examination was aided by videos where markers, obtained from a tracker developed by Zenuity, were shown along with the detections.

The focus when annotating was directed towards a couple of chosen vehicles surrounding the ego vehicle, and guardrails, for each dataset. Therefore, only time frames where the chosen vehicles have contributed with detections, have been annotated. Furthermore, this means that clusters that have not been connected to the chosen vehicles, or to guardrails, were not part of the data used for evaluation. These clusters include e.g. surroundings of the road and buildings next to it. The focus was to correctly cluster detections closer to the ego vehicle.

While adjusting clusters, labels were also assigned to clusters forming a line or an L-shape. The clusters were labeled as follows:

- Detections from a vehicle forming a clear L-shape, see Figure 1.2, were labelled ”L”.
- If a cluster representing a vehicle had a group of detections with higher density in one or two places compared to the rest of the cluster, see an example in

Figure 4.1, the label ”Wheel” was used and added to the shape label. This is because such detections are likely to come from the wheel houses.

- Guardrails and detections which formed shorter lines were marked as ”Line”.
- Clusters of no interest, i.e., the ones that were ignored in the evaluation, were labeled as ”Ignore”. Such clusters describes e.g. surroundings of the road.

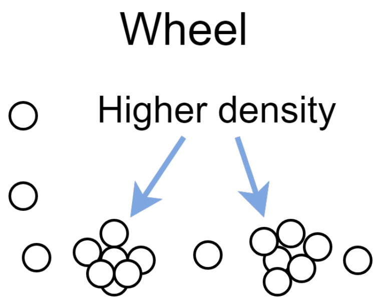


Figure 4.1: Example of a cluster with higher density at two places compared to the rest of the cluster, indicating wheel houses.

4.2 Performance measures

For the evaluation of clusters, four different performance measures were used: sensitivity, precision, oversegmentation and undersegmentation. The measures can be divided into two categories. Sensitivity and precision belongs to the first category which is a statistical measure for estimating the performance of the algorithm in terms of the amount of correctly classified points. The second category measures the segmentation performance, meaning the measure shows if a cluster has been oversegmented – the algorithm estimates multiple clusters that should really be one, or undersegmented – the algorithm estimates one cluster when there should really be multiple clusters. Further, a metric called Gaussian Wasserstein distance, was used to measure the distance between two clusters. Finally, an evaluation of the performance of the labelling of shapes was done.

This section will start by presenting the definitions for sensitivity and precision, followed by the Gaussian Wasserstein distance. It will then proceed by presenting definitions for over- and undersegmentation and finally explain how the shape evaluation was carried out. Henceforth, the annotated data will be referred to as the reference data and the result obtained from the clustering algorithm described in Section 3.1, will be referred to as the estimated data.

4.2.1 Sensitivity and precision

The first set of performance measures, sensitivity and precision, are used to evaluate the classification performance of a multiclass problem. In other words, to what extent does the proposed algorithm set the detections into correct clusters. This section uses the notation and follows the theory described in [18].

Before describing the measures, the following notations are introduced. Further, Figure 4.2 illustrates detections categorized by these notations. The black circles represents the reference data and the filled circles represents the estimated data where different coloring corresponds to different clusters.

- *True positive, TP,* is a detection that is correctly classified, meaning the reference and estimated data classify the point in the same way. Figure 4.2 illustrates true positives as green detections, encircled with black.
- *False negative, FN,* is a detection that belongs to the reference cluster under consideration, while the estimated data has that detection in another cluster or as an outlier. Figure 4.2 illustrates this as the red detections, encircled with black.
- *False positive, FP,* is a detection that is classified as belonging to the cluster by the estimate when the reference data says it should not belong to the cluster. Figure 4.2 illustrates this as the green detections without a black circle.

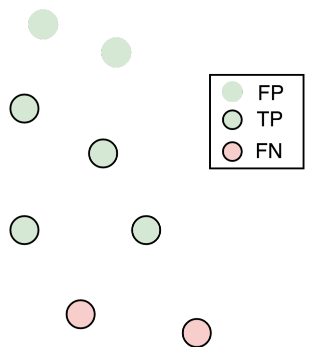


Figure 4.2: Illustration of a cluster according to reference and estimated data. The black circles forms the reference cluster and the detections filled with red and green are two different estimated clusters. Correctly clustered detections, true positives, TP, are the green detections encircled with black. The detections of the green estimated cluster outside the reference cluster are the false positives, FP. Finally, the red detections are wrongly clustered, false negatives, FN.

Sensitivity is the ratio between the number of true positives and the sum of true positives and false negatives, i.e., the part of the reference data that is correctly clustered by the proposed algorithm. This is calculated as

$$\text{sensitivity} \equiv \frac{\sum_{c\in C}(TP)_c}{\sum_{c\in C}(TP)_c + \sum_{c\in C}(FN)_c}, \tag{4.1}$$

where C is the set of all reference clusters, $(TP)_c$ is the number of true positives for cluster $c \in C$ and $(FP)_c$ the number of false negatives for cluster $c \in C$.

Precision is the ratio between the number of true positives and the sum of true positives and false positives, i.e., the part of the estimated data that is correctly clustered by the proposed algorithm. The definition of precision is

$$\text{precision} \equiv \frac{\sum_{c\in C}(TP)_c}{\sum_{c\in C}(TP)_c + \sum_{c\in C}(FP)_c},$$

where $(FP)_c$ is the number of the false positives for cluster $c \in C$ and remaining notations are the same as in Eq. 4.1.

Sensitivity and precision takes values between 0 and 1. If the proposed algorithm works perfectly and all detections are true positives, the sensitivity as well as the precision will be equal to 1. Both measures are equally important to obtain a qualitative evaluation. For instance, if all the detections in a time frame are clustered into one single cluster by the proposed algorithm, there will be no false negatives, whence, the sensitivity will be equal to 1. However, there will be a lot of false positives and the precision will therefore be very low. The opposite, sensitivity almost zero and precision equal to 1, would occur if there were just one reference cluster that had been separated into a number of estimated clusters by the proposed algorithm. Hence, these measures should be analyzed together and are not valid separately.

Since it is important to have high values for both sensitivity and precision, the measure average performance rate, is introduced. In this thesis, this measure is defined as the average of sensitivity and precision,

$$\text{Average performance rate} \equiv \frac{\text{sensitivity} + \text{precision}}{2}.$$

If the average performance rate is equal to 1, the estimated cluster corresponds exactly to the reference cluster, and the clustering is considered to be correct. The definition of a correct cluster will be presented in Section 4.2.3 and is illustrated in Figure 4.5.

4.2.2 Gaussian Wasserstein distance

When using sensitivity and precision, it is necessary to decide which detections correspond to true positives. For instance, if there is an oversegmented cluster such that there are three estimated clusters with parts of their detections corresponding to one reference cluster, which of the estimated clusters should be treated as the correct one? To solve this problem the Gaussian Wasserstein distance was used as a metric for how close clusters are. The theory in this section follows the theory in [19].

The idea behind the Gaussian Wasserstein distance is to approximate clusters with ellipses, see Figure 4.3, and find the estimated cluster whose ellipse is closest to the reference cluster's ellipse.

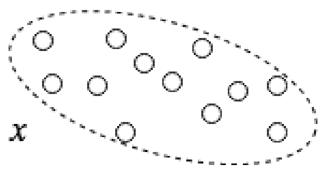


Figure 4.3: Example of cluster estimated by an ellipse x.

To calculate the Gaussian Wasserstein distance between two clusters the sample mean and the sample covariance of the clusters are used. The mean and covariance are used since an ellipse x can be interpreted by a Gaussian distribution,

$$\mathcal{N}_x = \mathcal{N}(m_x, s\Sigma_x),$$

where m_x is the mean of x, s is a scaling factor connected to the tolerance region (in this project, s is set to 1), and $\Sigma_x = \mathbf{RDR}^T$. Here, $\mathbf{R}$ is a rotation matrix and $\mathbf{D}$ is a diagonal matrix

$$\mathbf{R} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} (l_1)^2 & 0 \\ 0 & (l_2)^2 \end{bmatrix},$$

where ϕ gives the orientation, and l_1 and l_2 are the lengths of the major and minor semi-axes of the ellipse, respectively. Comparing this to a covariance matrix for a Gaussian distribution with two random variables, it can bee seen that $s\Sigma_x$ corresponds to a covariance matrix. The formula of the Gaussian Wasserstein distance between the ellipses x and $\hat{x}$, corresponding to two clusters, is

$$d_{gw}(\mathcal{N}_x, \mathcal{N}_{\hat{x}}) = ||m_x - m_{\hat{x}}||^2 + Tr\left(\Sigma_x + \Sigma_{\hat{x}} - 2\sqrt{\sqrt{\Sigma_x}\Sigma_{\hat{x}}\sqrt{\Sigma_x}}\right). \tag{4.2}$$

For each reference cluster, the Gaussian Wasserstein distance is calculated to each estimated cluster. The cluster with the least distance is chosen as the reference cluster's corresponding estimated one.

A problem that might appear is calculating the covariance of a cluster with only two detections. The covariance matrix will then be singular. This was avoided by approximating the cluster with a thin ellipse. To obtain this, the ellipse's minor semi-axis was perturbed with a small value μ. The covariance matrix for the ellipse is denoted $\hat{\Sigma}_x$, and calculated as

$$\hat{\Sigma}_x = \Sigma_x + \mathbf{R} \begin{bmatrix} 0 & 0 \\ 0 & \mu \end{bmatrix} \mathbf{R}^{\mathrm{T}}.$$

In Algorithm 6, the pseudocode for how sensitivity, precision and average performance rate were calculated with Gaussian Wasserstein distance is outlined. Here, C refers to the set of all reference clusters in the current time frame.

```
Data: Reference data and estimated data for all time frames in the 12 datasets
Result: Sensitivity, precision and average performance rate for each time frame
for Each time frame do
    for Each cluster $c$ in $C$ do
        Calculate the Gaussian Wasserstein distance to all estimated clusters with
          corresponding detections in $c$;
        Choose the estimated cluster with smallest distance to $c$;
        Calculate and save $TP$, $FP$ and $FN$ for $c$;
    end
    Calculate sensitivity, precision and average performance rate for the time frame
      with $TP$, $FP$ and $FN$ for each cluster;
end
```

Algorithm 6: The pseudocode for the evaluation of sensitivity, precision and average performance rate.

4.2.3 Oversegmentation and undersegmentation

The other category of performance measures is used to evaluate the segmentation performance of the proposed algorithm. Namely, if the algorithm estimates too many clusters, oversegmentation, or too few clusters, undersegmentation. These measures are connected to sensitivity and precision. However, while sensitivity and precision focuses on the amount of correctly clustered detections, over- and undersegmentation focuses on the composition of the clusters.

In the left illustration of Figure 4.4, oversegmentation is displayed. It is visible that the reference cluster, detections with black solid circles, has been estimated as two different small clusters, the green and the red, filled, circles. The illustration to the right shows undersegmentation. This is the opposite to oversegmentation. In this case, two different reference clusters, solid and dashed circles, have been estimated as one cluster, illustrated as green, filled, circles.

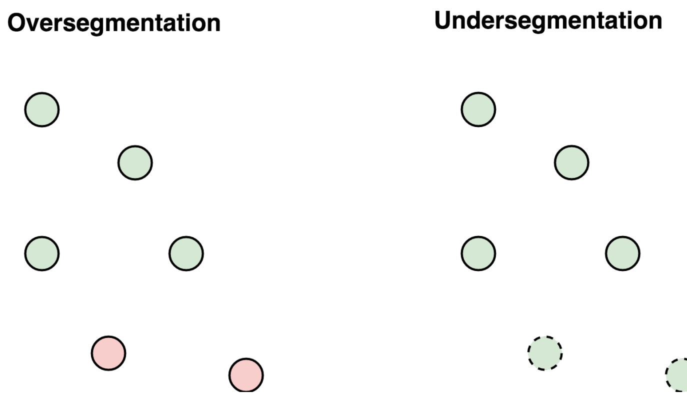


Figure 4.4: Illustration of over- and undersegmentation. The leftmost figure shows oversegmentation. The black circles corresponds to reference data and the green and red, filled, circles corresponds to two cluster estimates. The rightmost figure shows undersegmentation where solid and dashed circles correspond to reference data and the green, filled, circle is an estimated cluster.

In connection with over- and undersegmentation, three other measures were used. These were, the number of *correct clusters*, *false outliers* (reference clusters that were estimated to be outliers) and *false clusters* (outliers that were estimated to be a cluster). Figure 4.5 illustrates a correct cluster. The cluster is neither over- nor undersegmented and the number of true positives is equal to the number of detections in the reference cluster. In other words, the clustering is correct.

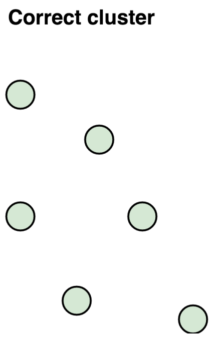


Figure 4.5: Illustration of a correct cluster. The black circles corresponds to the reference cluster and the green, filled, circles to the estimated data.

Figure 4.6 illustrates false outliers and a false cluster to the left and right respectively. To the left, the detections have been estimated to be outliers, red ”-1”, although they should form a cluster according to the reference, black circles. Hence, the detections are falsely estimated as outliers. To the right, the detections have been estimated to form a cluster, red filled circles, although they should be outliers according to the reference, black circles with ”-1”. Hence, the detections are falsely clustered.

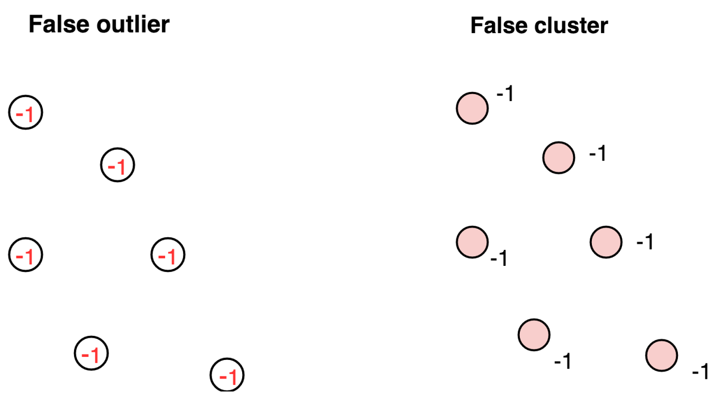


Figure 4.6: Illustration of false outliers to the left and a false cluster to the right. In both figures, the blacks circles corresponds to the reference data. The right figure shows that all detections have been estimated to be outliers, red "-1" when they should not – false outlier. The left figure shows that all detections have been estimated to form a cluster, red filled circles, when they should be outliers, "-1" – false cluster.

In Algorithm 7, an outline for the evaluation of the segmentation is presented. Here, C refers to the set of all reference clusters in the current time frame.

```
Data: All 12 datasets for both the reference data and the estimated data
Result: Evaluation of the segmentation for each dataset
for Dataset $i$ do
    for Each time frame in dataset $i$ do
        for Each clusters $c$ in $C$ do
            Check if $c$ is correct, oversegmented, undersegmented, false outliers or a
              false cluster;
        end
        Calculate the percentage of over- and undersegmentation;
    end
    Calculate the percentage of correct clusters, oversegmented clusters,
      undersegmented cluster, clusters of false outliers and false clusters, for dataset
      $i$;
end
```

Algorithm 7: The pseudocode for the evaluation of the segmentation.

4.2.4 Evaluating shapes

A straightforward measure was used to evaluate the performance of finding the correct labels for a line and an L-shape. All clusters from the reference data corresponding to a line or an L-shape were examined. If a corresponding detection in the estimated data had the same label as the reference, it was noted in a binary way. Then, the percentage of number of correctly labeled detections was calculated for each dataset, to get a measure of the performance. An outline of this procedure for the label ”L” is displayed in Algorithm 8. For the label ”Line” a similar evaluation was done.

```
Data: All 12 datasets
Result: Percentage of correctly labeled detections
Initiate empty list for number of correct labels;
for All detections in dataset i do
    if reference label is ”L” then
        if corresponding estimated label is ”L” then
            Add one to the list for number of correct labels;
        else
            Continue to next iteration;
        end
    else
        Continue to next iteration;
    end
    Calculate the percentage of correctly labeled detections;
end
```

Algorithm 8: The pseudocode for the shape evaluation.

5 Results and Discussion

In this chapter, tuning and choice of parameter values, evaluation results and an analysis of the velocity estimation, are presented. The tuning is made with 5-fold cross validation. This will be presented first. Then, the choice of parameter values for the parameters presented in Section 3.2.1, 3.2.2 and 3.2.3, are presented.

The next part of this chapter presents and discusses the results from applying the proposed algorithm to the reference data, using the performance measures of Chapter 4. Further, the estimated velocity for clusters have not been evaluated in the same manner since there is no ground truth for the velocity. The last section in this chapter therefore analyzes the obtained velocities and discusses them, rather than presenting an evaluation result.

5.1 Tuning and parameter values

All parameter values that have been used were obtained by tuning and observing the data. Tuning of parameter combinations have been made with cross validation, and some of the combinations will be presented here. Further, this section will also present the chosen values for the parameters presented in Table 3.1, 3.2 and the values presented in Section 3.2.3.

5.1.1 Tuning parameters

The tuning was done by considering different combinations of parameter choices. Table 5.1 shows the last five of these parameter combinations. In that table, ε_1 is the value used in DBSCAN for coarse clustering and ε_2 is the value used in DBSCAN as the final clustering step. Before testing these five different combinations, another tuning and some discussions were carried out in order to obtain initial values to start with. The obtained results when using these five different combinations are presented in Table 5.2, where the values are given in percent.

	Parameters	**Values**
1	ε_2 l_{guard} T	3 20 2.5
2	ε_2 l_{guard} T	4 20 2.5
3	$\varepsilon_{1,2}$ l_{guard} T	4.3 20 3
4	$\varepsilon_{1,2}$ l_{guard} T	4.3 22 3
5	ε_2 l_{guard} T	4 20 3

Table 5.1: Parameter combinations used for tuning.

	1	**2**	**3**	**4**	**5**
Average sensitivity	87.12	92.42	93.48	93.24	92.28
Median sensitivity	96.15	100.00	100.00	100.00	100.00
Average precision	85.34	85.39	84.60	83.98	85.06
Median precision	94.12	94.12	94.12	93.65	94.12
Average oversegmentation	31.43	38.35	39.45	40.18	38.55
Median oversegmentation	0.00	33.33	33.33	33.33	33.33
Average undersegmentation	25.37	22.18	21.44	20.91	23.36
Median undersegmentation	0.00	0.00	0.00	0.00	0.00

Table 5.2: Obtained performance measures from the five parameter combinations, shown in percent.

It can be seen that the tuning did not yield large differences in the performance measures between the folds. Hence, tuning was quite difficult since there is no clear parameter combination that gives the best performance measures. However, by examining the performance measures in Table 5.2, and the data, the last combination was chosen.

5.1.2 Parameter choices

The parameter values that was used to estimate horizontal lines are shown in Table 5.3. As previously stated, the values came from tuning and observing the data. In Section 3.2.1, these parameters were outlined.

Constraint	**Parameter**	**Value**
1	$(SA)_{guard}$	15m
2	l_{guard}	20m
3	N_{guard}	8 detections
4	N_{line}	4 detections
5	l_{line}	2m
6	k	0.02 ($\approx$ 1.15°)
7	T	3
8	Num^{it}_{line}	40
9	P_{line}	80%

Table 5.3: Parameter values for a horizontal line.

Table 5.4 shows the parameter values that were chosen to estimate L-shapes. In Section 3.2.2, these parameters were outlined. Taking the search area into account, constraint 1, and the fact that the scenario is situated on the highway, vehicles overtaking the ego vehicle are more likely to have an orientation aligned with the ego vehicle, previously shown in Figure 3.6. As stated in Section 1.2, it is assumed in this thesis that narrow cut ins do not occur, i.e., there will not be any cut ins within the distances of the search area. However, this assumption does not hold for traffic jams and will be discussed further in Chapter 6. Further, the values for constraints number 5-6 were obtained by considering typical dimensions of cars and trucks. For trucks, 8m is quite low since a typical truck is 7-12m, or, 16-24m with a trailer [20]. Increasing $hMax$ to better suit the dimensions of a truck, there is a possibility that cars obtain an L-shape estimate where the horizontal line is longer than the actual extension of the car. Using 8m, there is a possibility that the whole extension of a truck is not captured, but the corner of the L-shape may still be estimated. In this thesis, we chose to prioritize estimating the corner of an L-shape, rather than capturing the full extension of a truck. This limitation is something that should be revised and will be brought up in Chapter 6 regarding future work.

Constraint	**Parameter**	**Value**
1	$(SA)_L^{long}$ and $(SA)_L^{lat}$	13m and 8m, respectively
2	N_L	5 detections
3	N_{line}	4 detections
4	k	0.02 ($\approx 1.15°$)
5	$hMin$ and $hMax$	1m and 8m, respectively
6	$vMin$ and $vMax$	1m and 3m, respectively
7	Num_L^{it}	1500

Table 5.4: Parameter values for an L-shape.

Table 5.5 shows the parameter values that were used to estimate the velocity for a cluster. As previously stated, the values came from tuning and observing the data. In section 3.2.3, these parameters were outlined.

Parameter	**Value**
N_{vel}	3
e_{vel}	0.1
Num_{vel}^{it}	50
P_{vel}^1	80%
P_{vel}^2	50%

Table 5.5: Parameter values for estimating the velocity.

5.2 Evaluation results

This section will present the performance measures: sensitivity, precision, average performance rate, and segmentation, that were obtained from using the parameter values in Table 5.3 and Table 5.4. Using those values gave an average run time 0.3953sec, for one time frame, on a computer with an Intel Core 17-7700HQ processor. In Appendix A, Table A.1 shows the average run times, per time frame, for each datset.

The results for the performance measures are obtained through cross validation with five subsets of the 12 datasets. The results presented in this section are averaged over all time frames in all datasets. Table A.2 in Appendix A shows average and median sensitivity, precision and segmentation for each of the five test sets. In order to have a reference to compare the results of the proposed algorithm with, the same performance measures are presented for the case of only using DBSCAN to cluster for $\varepsilon = 4.3$, $minPts = 2$. Finally, the shape evaluation is presented.

5.2.1 Sensitivity and precision

The sensitivity and precision for each time frame is shown in Figure 5.1 and 5.2 for the proposed algorithm and DBSCAN, respectively. The figures also show the obtained average and median values over all time frames, also presented in Table 5.6, where the values are given in percent. In general, sensitivity takes higher values than precision in both figures, i.e., the number of false negatives is less than the number of false positives. This indicates that the algorithm estimates too large clusters, i.e., most clusters have extra detections that should not be there. This is connected to undersegmentation, see Section 5.2.2.

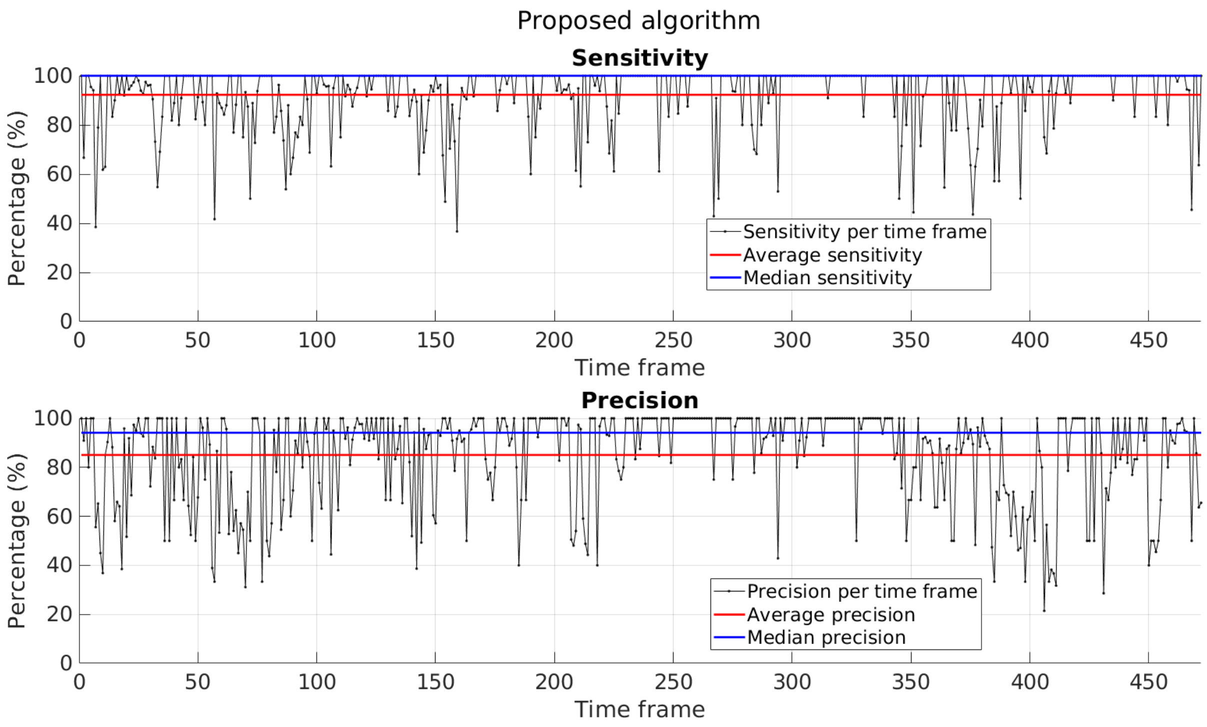


Figure 5.1: Sensitivity and precision for all time frames in the upper and lower plot respectively. The red and the blue lines corresponds to the average and median results over all time frames respectively.

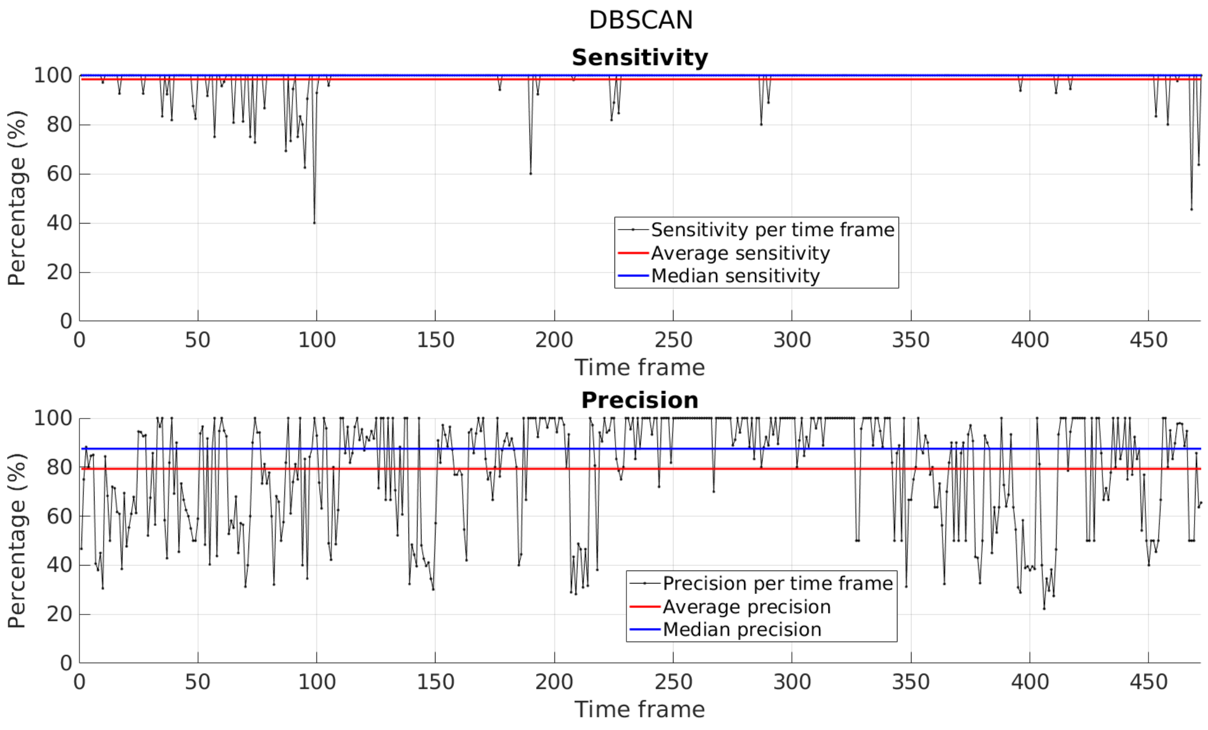


Figure 5.2: Sensitivity and precision for all time frames in the upper and lower plot respectively. The red and the blue lines corresponds to the average and median results over all time frames respectively.

	Average sensitivity	**Median sensitivity**	**Average precision**	**Median precision**
Proposed Algorithm	92.28	100.00	85.06	94.12
DBSCAN	98.40	100.00	79.33	87.50

Table 5.6: Average and median sensitivity and precision, given in percent, for the proposed algorithm and DBSCAN, respectively.

It can also be seen that sensitivity for only using DBSCAN is better, compared to the proposed algorithm. The values for sensitivity does not fluctuate as much, between time frames, in Figure 5.2, as for sensitivity in Figure 5.1. The difference in fluctuation is not as large when comparing the precision between the two figures.

The fluctuation is because the data sometimes differs much between time frames, and sometimes not, i.e., the reflective properties of the surroundings vary a lot. Furthermore, the proposed algorithm has been applied independently on each time frame. Hence, the performance could be improved if the clustering could be performed considering two or more time frames simultaneously. Something else to consider is that the fluctuation could be due to low robustness of the proposed algorithm, in combination with varying data.

Moreover, the median is higher than the average, which indicates the presence of some extreme values affecting the average but not the median, for Figure 5.1 and 5.2. It is also important to note that, since the reference data was obtained by manual annotation, some bad results are due to incorrect annotation rather than incorrect clustering.

Figure 5.3 and 5.4 shows the average performance rate for each time frame for the proposed algorithm and DBSCAN, respectively. The average and median are 88.67% and 92.86%, respectively, for the proposed algorithm and 88.86% and 92.86%, respectively, for DBSCAN. It is visible that the average over all time frames is lower than the median. Based on the results shown in Figure 5.1 and 5.2, together with Eq. (4.2.1), the outcome is reasonable.

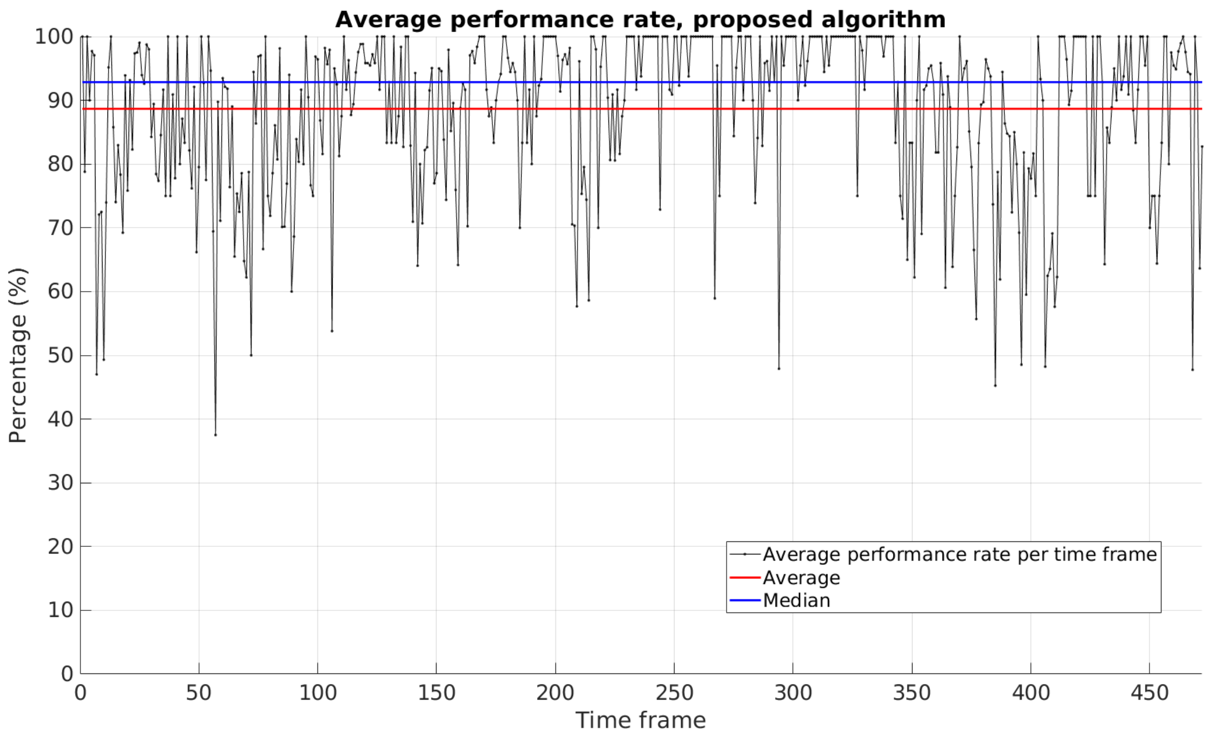


Figure 5.3: Average performance rate for all time frames. The red and the blue lines correspond to the average and median results, respectively, over all time frames.

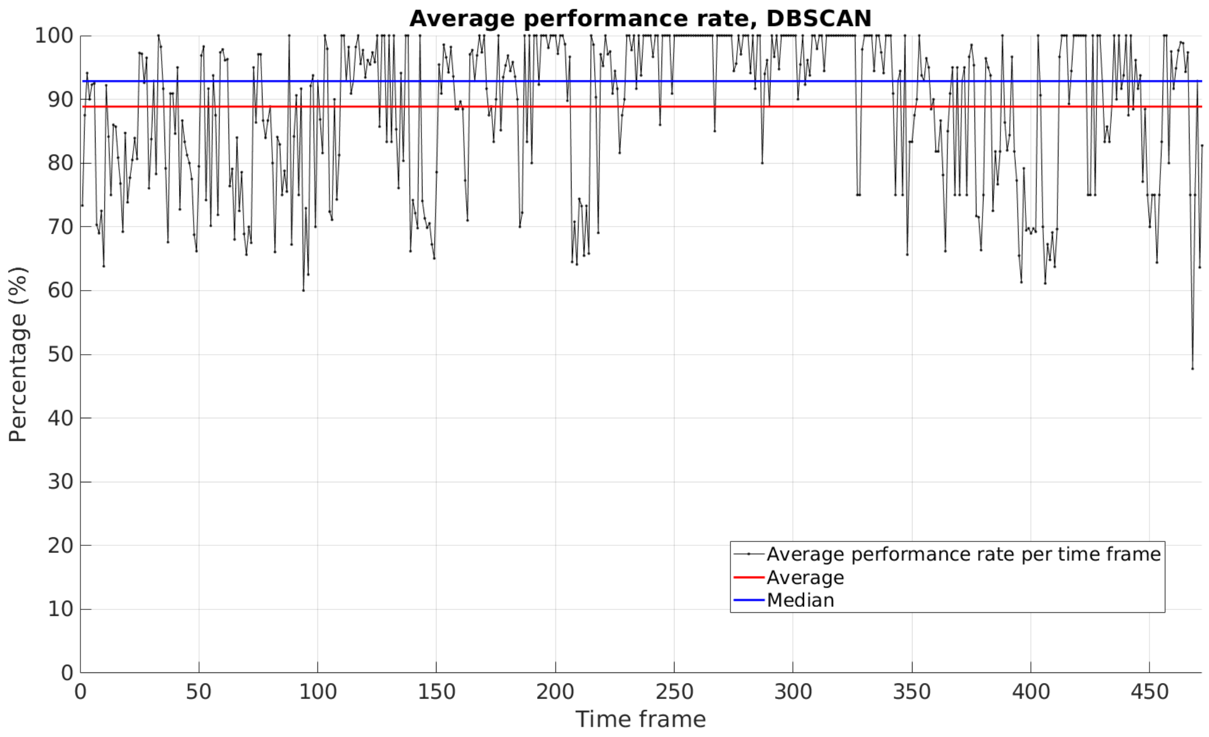


Figure 5.4: Average performance rate for all time frames. The red and the blue lines correspond to the average and median results, respectively, over all time frames.

5.2.2 Segmentation

Figure 5.5 and 5.6 shows the percentage of over- and undersegmented clusters for each time frame, for the proposed algorithm and DBSCAN, respectively. The desired percentage of over- and undersegmentation is for them to be as low as possible. As in the case with sensitivity and precision, the median value is better than the average for both figures. Further, the average and median values for both cases are shown in Table 5.7 where the values are given in percentage. From these figures, and the table, it is visible that, even though DBSCAN gives lower values for the undersegmentation than the proposed algorithm, it has much higher oversegmentation, which is undesired. Henceforth, this section will discuss the evaluation of the proposed algorithm, and the comparison of the proposed algorithm and DBSCAN will continue in the next section.

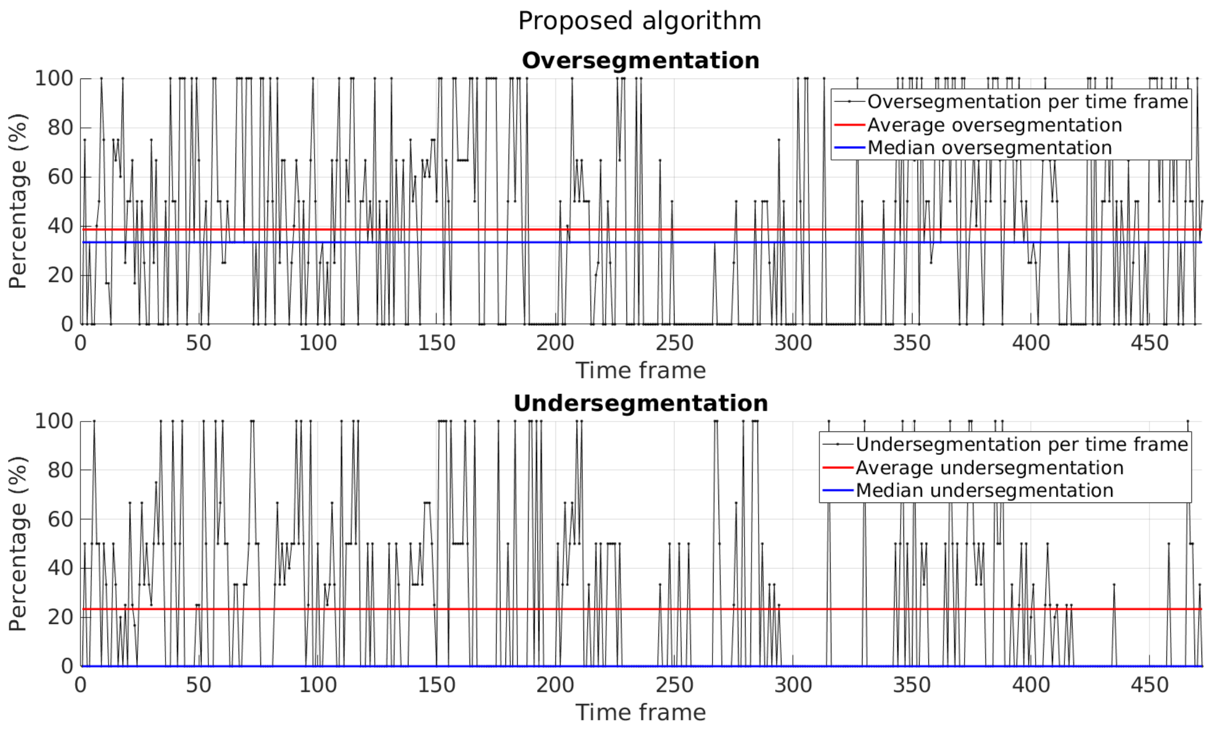


Figure 5.5: Percentage of over- and undersegmentation for all time frames in the upper and lower plot respectively, for the results obtained from the proposed algorithm. The time frames are not sorted in any way. The red and the blue lines correspond to the average and median results over all time frames, respectively. It can be seen that the median is lower than the average for both over- and undersegmentation, showing that there are some extreme values that makes the average higher.

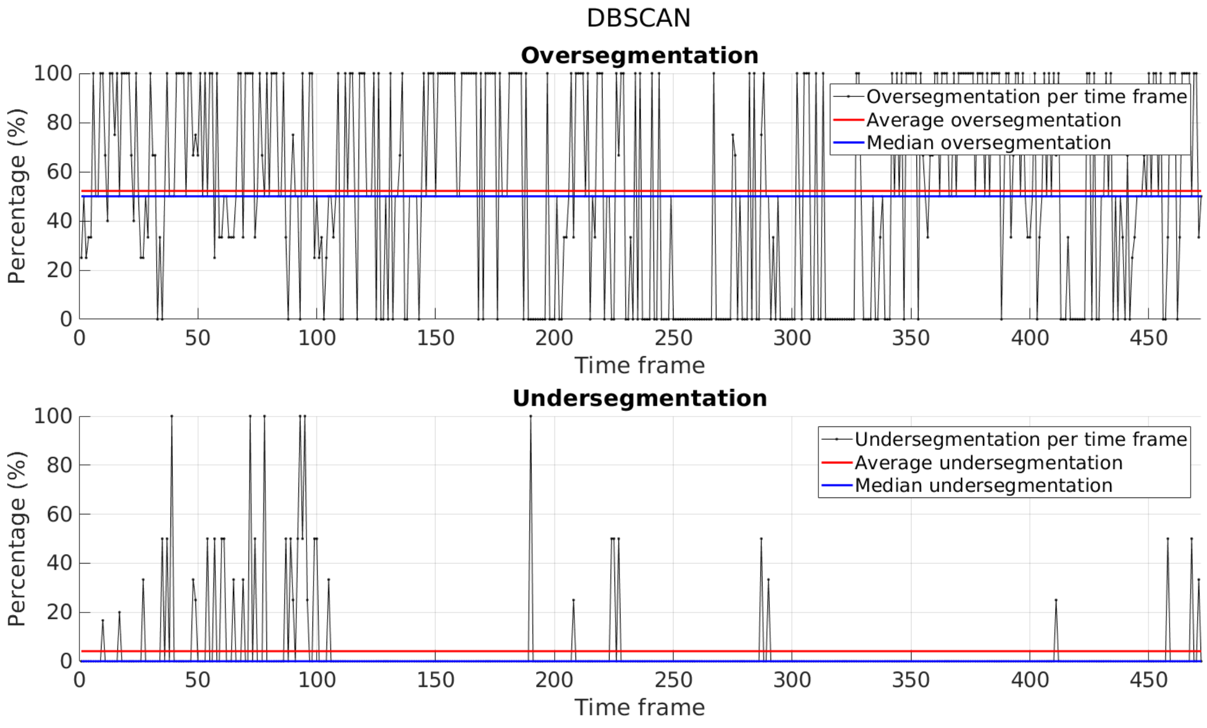


Figure 5.6: Percentage of over- and undersegmentation for all time frames in the upper and lower plot respectively, for results obtained by only using DBSCAN. The time frames are not sorted in any way. The red and the blue lines correspond to the average and median results over all time frames, respectively.

	Average oversegmentation	**Median oversegmentation**	**Average undersegmentation**	**Median undersegmentation**
Proposed Algorithm	38.55	33.33	23.36	0.00
DBSCAN	52.18	50.00	4.12	0.00

Table 5.7: The average and median values for the segmentation for the proposed algorithm and for DBSCAN, the values are given in percentage.

For Figure 5.5, the fluctuation between time frames can be explained in the same way as previously. The data sometimes differ much between time frames and sometimes not, and the time frames are clustered independently. The results could improve if two or more time frames are considered simultaneously. The proposed algorithm could also have a low robustness, which, in combination with varying data, explains the fluctuation. Also, as previously stated, the annotation is not perfect. Moreover, if a time frame contains, e.g., two reference clusters which are estimated as one, the undersegmentation for that time frame will be 100%. For another time frame, there might be five reference clusters, and among these five, two are estimated as one, while the others are correctly clustered (neither over- nor undersegmented). For that case, the undersegmentation will be 40%. An analogous reasoning holds for oversegmentation. Hence, the same type of incorrect clustering could give very different values depending on the number of reference clusters in a time frame.

Figure 5.7 and 5.8 shows the average sensitivity, precision and segmentation over all time frames for each dataset, 1-12, see Table 4.1, for the proposed algorithm and DBSCAN, respectively. Comparing these two figures, it is visible that in general, the undersegmentation is lower, while the oversegmentation is higher, for the proposed algorithm. Something else to notice is that the difference between sensitivity and precision is less and that the precision is higher for the proposed algorithm than for DBSCAN.

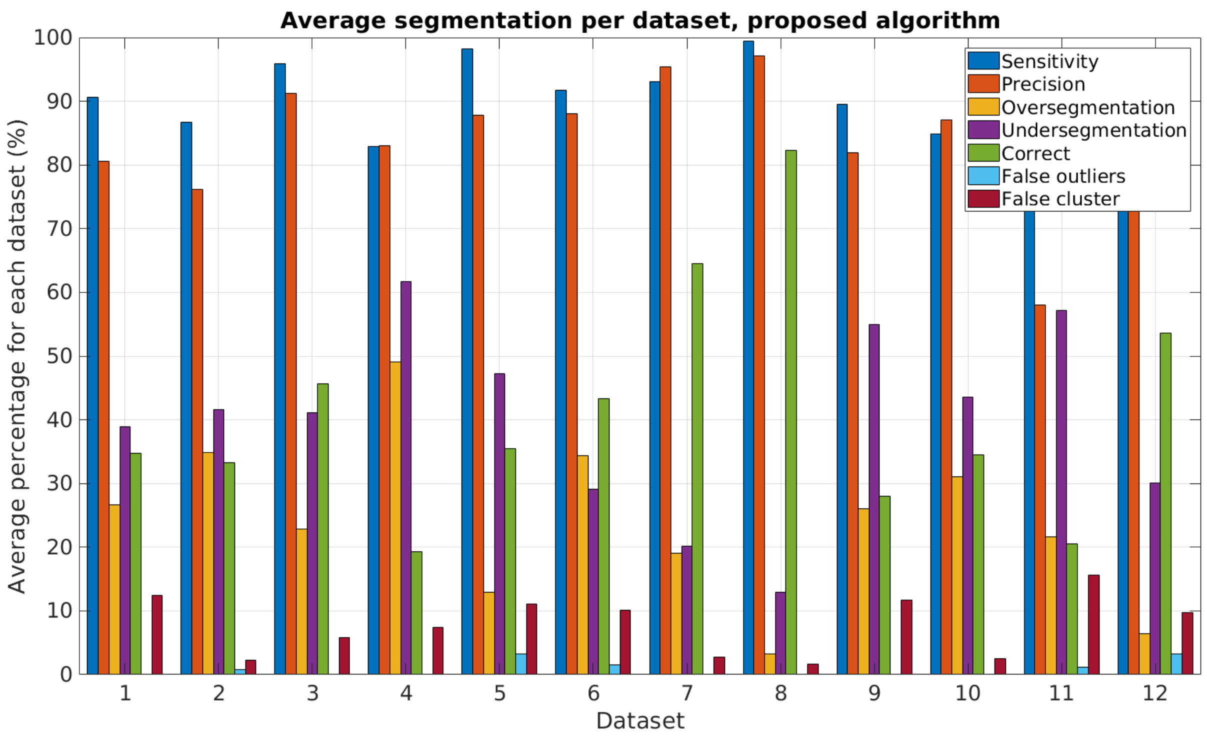


Figure 5.7: A bar plot of average percentage of sensitivity, precision, oversegmentation, undersegmentation, correct clustered, clusters with false outliers and false clusters for each one of the 12 datasets, for the proposed algorithm. Sensitivity and precision have the highest bars. Of the remaining five bars, the correct and false clusters are the highest and lowest respectively, while the bars for the false outliers are not present. This is what is wanted. The undersegmentation is in general higher than the oversegmentation which was also observed in Figure 5.5.

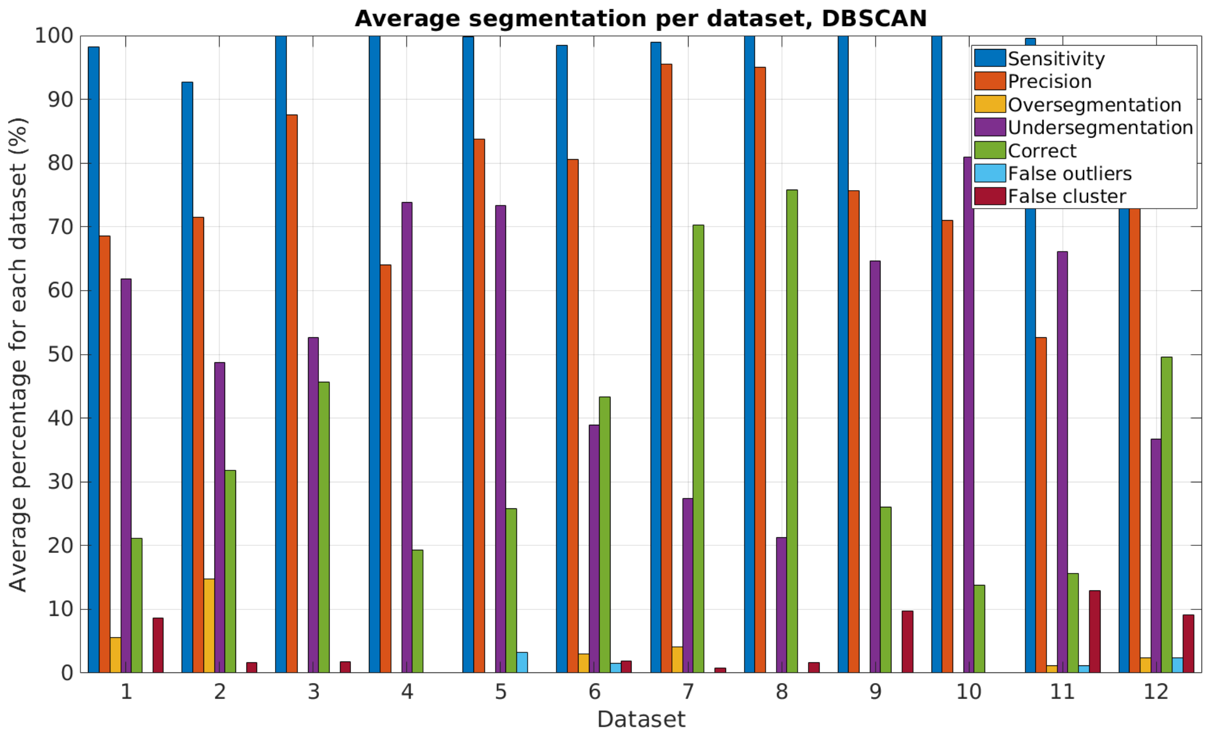


Figure 5.8: A bar plot of average percentage of sensitivity, precision, oversegmentation, undersegmentation, correct clustered, clusters with false outliers and false clusters for each one of the 12 datasets, for DBSCAN. Sensitivity and precision have the highest bars. Of the remaining five bars, the correct clusters and the vars for the undersegmentation are the highest. The bars showing the false clusters low and the false outliers are few and low. The undersegmentation is higher than the oversegmentation.

For both figures, the bars for sensitivity and precision are the highest. Examining Figure 5.7 closer, the bars for over- and undersegmentation competes with the bars for correct clusters. The desired outcome is for the bars representing the correct clusters to be as high as possible, and the bars for over- and undersegmentation and false outliers and false clusters, to be as low as possible. The bars for false outliers and false clusters are already quite low. Hence, the over- and undersegmentation need to decrease. Henceforth, the discussion will be about the evaluation of the proposed algorithm.

As have already been seen in Figure 5.5, the undersegmentation is in general higher than the oversegmentation. A higher undersegmentation is expected since DBSCAN is being used in the proposed algorithm. A target close by result in many detections close to each other, while a target further away result in fewer and more sparse detections. When choosing ε, it is possible that it becomes too large for detections close to the radar or too small for detections further away. Regarding $minPts$, it could be enough with less detections to form a cluster further away, than it is for closer detections. The fixed density parameters makes it difficult for DBSCAN to adapt and for the chosen ε and $minPts$, the proposed algorithm tends to undersegment.

As mentioned in Section 2.1, GB-DBSCAN is a more dynamic choice. However, using GB-DBSCAN, there was a loss in symmetry meaning that between two points, p and q, p could be a neighbour to q but q was not necessarily a neighbour to p. The reason for this is that the obtained width, w, of the search area for point q, was larger than the obtained w for point p. Hence, p ended up in the search area of point q, but not the other way around. This was not the case with DBSCAN. Further, studying the output from DBSCAN and GB-DBSCAN showed that DBSCAN tended to undersegment while GB-DBSCAN tended to oversegment. Trying to tune GB-DBSCAN to oversegment less, resulted in undersegmentation in the angular direction instead. Two conclusions were drawn. First, GB-DBSCAN did not seem to work well for the data in this thesis. Second, it did not provide anything to the proposed algorithm. The remaining detections in D before the final step, see Section 3.1, often formed distinct clusters, which were easy for DBSCAN to estimate.

Table 4.1 in Section 4.1 describes the scenarios for each dataset. Comparing these scenarios to Figure 5.5, we make the following inferences. It is visible from the figure that for dataset 1 and 2, the bar representing undersegmentation is the highest. In these scenarios the ego vehicle is in the middle lane with a car overtaking it to the left. Furthermore there is a guardrail to the left of the overtaking vehicle. As previously stated, targets close to the ego vehicle result in many detections. This can make it difficult for the algorithm to distinguish between the overtaking vehicle and the guardrail, which could be a reason for the inferior result of these datasets.

Datasets 7 and 8 have the highest number of correct clusters and the rest of the bars are quite low, which is good. In these scenarios there was a single car overtaking the ego vehicle, as for datasets 1 and 2, but in these cases there was no guardrail. The scenarios for datasets 9, 10 and 11 all takes place in a traffic jam. This means a lot of vehicles are close to each other, leading to a lot of detections which makes it difficult to cluster. This can explain the rather high degree of undersegmented clusters compared to the number of correct clusters. Further, the assumption about L-shapes being aligned with the ego vehicle does not hold, as stated in Section 5.1. This conservative assumption could also be a reason for undersegmented clusters.

Overall, the result is varying depending on the scenarios. However, it is important to notice that only one detection clustered incorrectly according to the reference data will lead to a cluster being placed among the undersegmented or oversegmented clusters instead of the correct ones. Looking at Figure 5.7 again, at the bars for e.g. dataset 3, the high sensitivity means that on average, approximately 95% of all detections are true positives. This means that approximately 5% give rise to higher values for over- and undersegmentation, because of incorrect clustering.

5.2.3 Shape evaluation

This section presents the shape evaluation by studying the shape label in the reference data and see if corresponding detections in the estimated data have obtained the same label. Figure 5.9 shows a bar plot of the average percentage of correctly labeled detections when considering lines and L-shapes for all 12 datasets separately. The blue bars correspond to L-shapes and the red bars correspond to lines. In general, the proposed algorithm used for clustering finds a higher percentage of the detections labeled as L-shapes than the detections labeled as lines. Furthermore, only clusters annotated as lines or L-shapes have been evaluated, which is a subset of all the data since not all clusters are lines or L-shapes. Therefore, the data used for this evaluation was limited compared to previous results. Lower data quantity means lower quality of the evaluation.

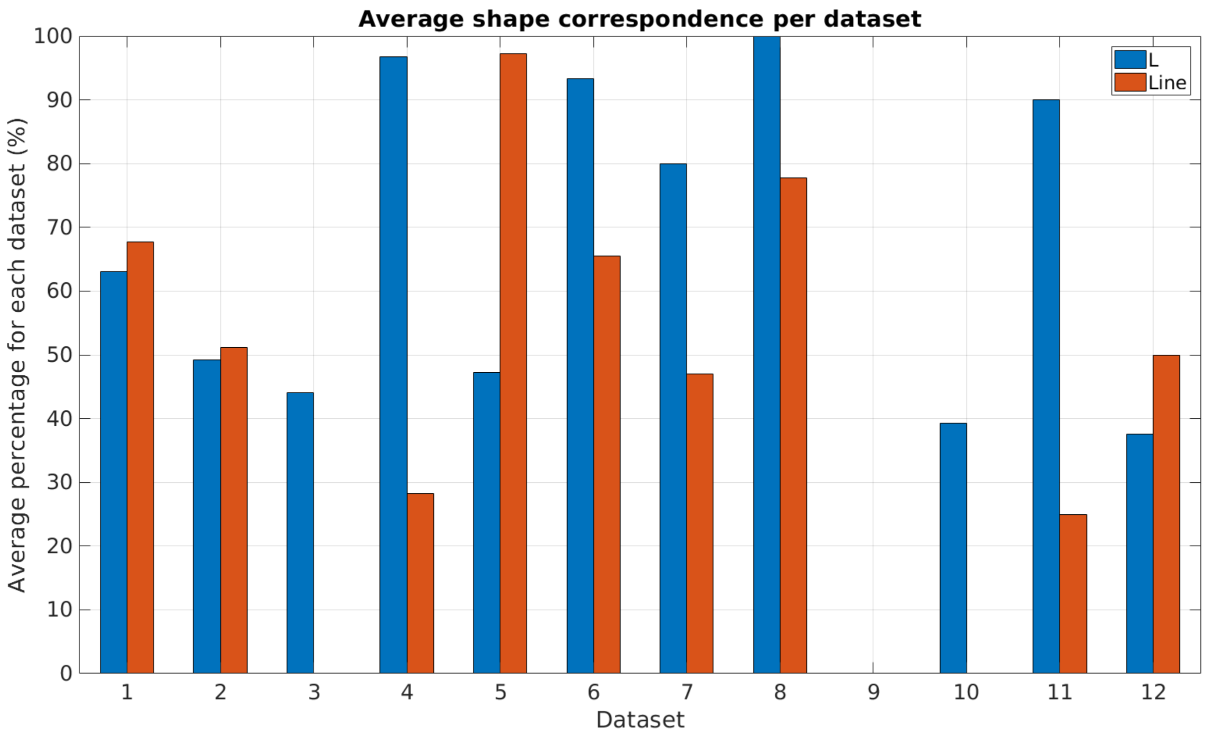


Figure 5.9: Bar plot of average percentage of shape correspondence between all the reference clusters and the evaluated clusters for each one of the 12 datasets.

Something else to consider in Figure 5.9 are the datasets with empty bars, e.g., the absence of a red bar in the third dataset. This implies that either the result is bad, or there are no clusters corresponding to L-shapes or lines in these datasets. In Table 5.8, the total number of detections forming L-shapes and lines are displayed for both the reference data and the estimated data for each dataset. With help from this table, it can be seen that the lack of bars corresponding to, e.g., dataset 9 in Figure 5.9, is reasonable since there are no L-shapes in that time frame and very few lines.

Overall, there are higher values for the L-shapes and lines in the estimated data than the reference data. Once again, one reason for the difference in labels for the reference and estimate can be explained by incorrectness in the annotation. Remember that Table 5.8 shows detections, and not clusters, that have been given a label. Failing to cluster or annotate an L-shape would correspond to at least five detections labeled as L's. For a guardrail, failing to cluster or annotate would correspond to at least eight detections labeled as lines. Therefore, it is reasonable that the number of detections in the table differs between the reference data and the estimated data.

Dataset	***L's in reference data***	***Lines in reference data***	***L's in estimated data***	***Lines in estimated data***
1	46	704	222	716
2	61	536	93	422
3	43	0	170	194
4	31	246	396	142
5	57	30	163	98
6	36	315	666	198
7	35	250	56	353
8	11	91	103	156
9	0	5	118	38
10	52	0	192	46
11	39	20	182	95
12	60	7	289	92

Table 5.8: The total number of detections forming L-shapes and lines for both the reference data and the estimated data for each sensor.

5.3 Velocity estimation

As previously stated, there is no ground truth for the velocity of the clusters. Thereby, this section does not evaluate the velocity estimation in the same manner, it will rather analyze and discuss them. First, a suitable value for N_{vel} will be examined. Then, there will be a presentation of the average percentage of clusters that were estimated with one, two and no velocity. Further, we will look into if there are any clusters with two estimated velocities that correspond to a cluster that was labeled with "Wheel". Finally, it will be examined if the velocity estimates differ much between iterations.

In Table 5.5, $N_{vel} = 3$, i.e., three detections are drawn in order to estimate the velocity of a cluster. Three different values for N_{vel} were tested for estimating the velocity. They will henceforth be referred to as methods and denoted as N_2, N_3 and

N_4, which translates into using 2, 3 and 4, randomly drawn detections to estimate the velocity, respectively.

For each method, a vector with ones and zeros was created. The ones and zeros were obtained by running RANSAC with 200 iterations for each cluster consisting of at least four detections. If a velocity was estimated for a cluster after the 200 iterations, a one was put in the vector, if it was not, a zero was put in the vector. By subtracting one vector from another, the resulting vector then consisted of zeros for all clusters where both methods succeeded in estimating a velocity, and ones and minus ones for clusters where only one of the methods succeeded in estimating a velocity. Figure 5.10, shows the vectors for $N_3 - N_2$, $N_4 - N_3$ and $N_4 - N_2$ in that order. In the top plot, it can be seen that N_3 estimates more velocities than N_2. The middle plot shows that N_3 also estimates more velocities than N_4. Finally, in the bottom plot, N_2 succeeds to estimate more velocities than N_4. Thereby, $N_{vel} = 3$, as previously stated, is chosen for robustness.

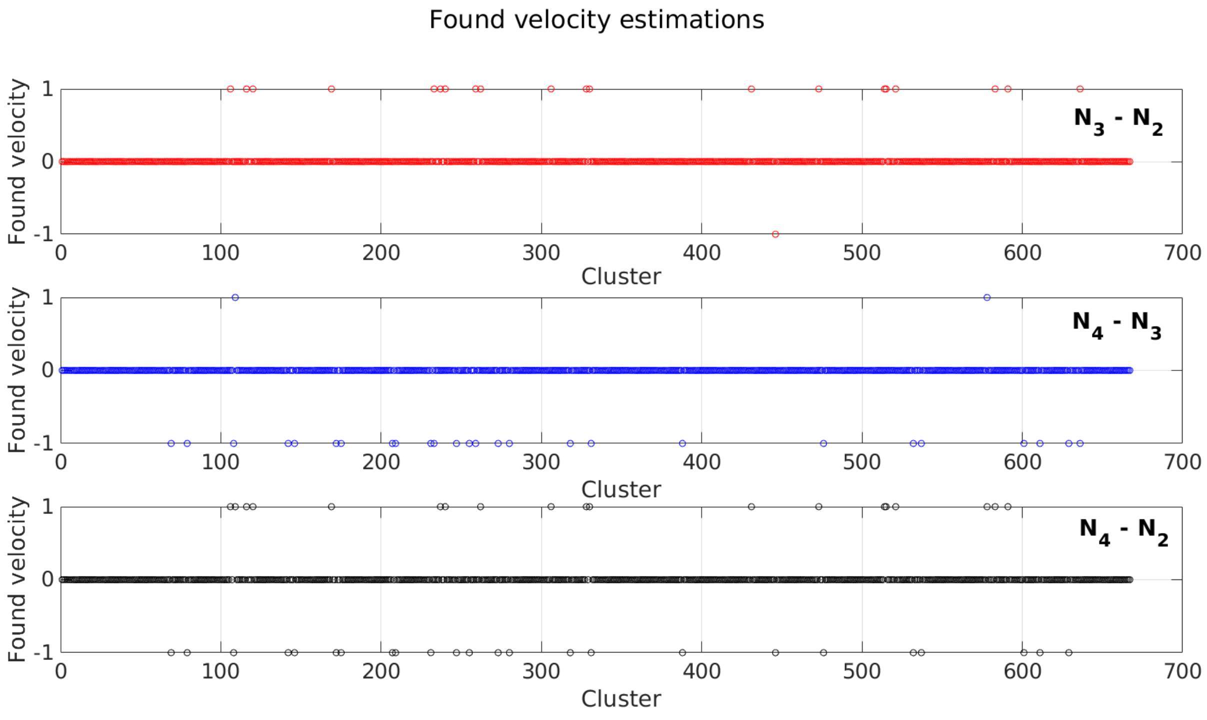


Figure 5.10: The difference when using three methods, to estimate the velocity of a cluster, denoted as N_2, N_3 and N_4. The subscripts stand for how many detections that were randomly drawn to estimate the velocity with RANSAC. Each plot shows a comparison between two methods. The value zero indicates that both methods have estimated a velocity, one and minus one indicates that one of the methods have estimated a velocity, but not the other.

In Figure 5.11, a bar plot of the average percentage of clusters with different velocity estimations for each of the 12 datasets is shown. The first bar represents the percentage of clusters where the estimated velocity had at least 80% inliers. The second represents clusters with two estimated velocities, i.e, clusters where at least 50% of the detections were inliers to one estimated velocity, and at least 80% of the remaining detections were inliers to another estimated velocity. The third bar represents clusters with one velocity of at least 50% inliers but not more than 80%. The fourth bar represents the clusters which did not fulfill any of the previous constraints, i.e. the average percentage of clusters where a velocity estimate yielded less than 50% inliers.

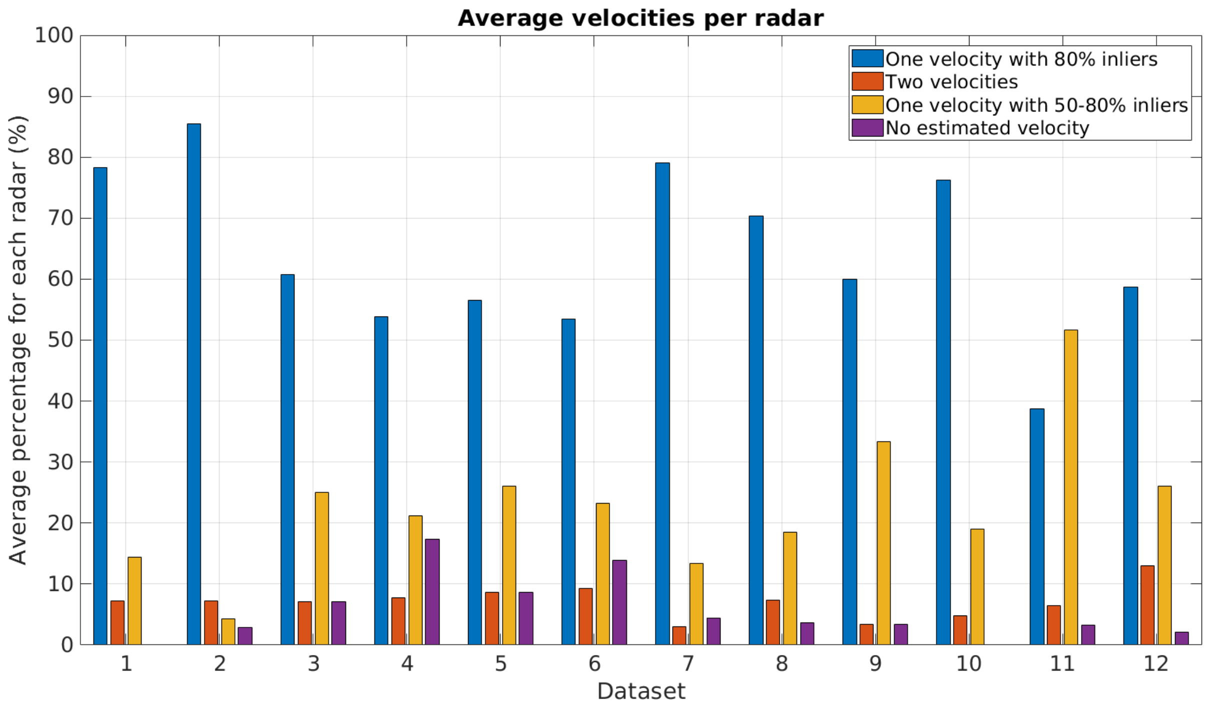


Figure 5.11: Bar plot of average percentage of clusters with different velocity estimations for each one of the 12 datasets. The first bar shows the percentage of clusters with one velocity where inliers are at least 80%. The second shows clusters with two estimated velocities and the third shows clusters with one velocity with at least 50% inliers but not more than 80%. The fourth bar corresponds to the clusters which were estimated to have have no velocity since the inliers did not reach more than 50%.

It is visible that clusters with one velocity estimate with at least 80% inliers is the most common result. This indicates a degree of correct clustering since an estimated cluster, that according to reference data corresponds to multiple clusters, should yield more varying velocities. Being able to estimate two velocities for a cluster could indicate the presence of a vehicle where the body and the wheel house are estimated with one velocity respectively.

Table 5.9 shows a comparison of the detections from the estimated data from clusters with two velocities and detections from the reference data with the label ”Wheel”, i.e., vehicles with a visible body and wheel house, for each dataset separately. The first column shows the dataset number, the second column shows the total number of detections with label ”Wheel” from the reference data, and the third column shows the total number of detections from clusters with two velocities in the estimated data. The fourth column shows the number of detections present in both the second and third column, the intersection of the columns, i.e., the number of detections from clusters with two estimated velocities and with a ”Wheel” label.

The intersection indicates that there is a correlation between a cluster having two velocities and that this cluster corresponds to a vehicle. However, the number of detections in the intersection is quite low. This can be due to, e.g., the manual annotation or to that very long objects, e.g. guardrails, can have one velocity on one side of the symmetry axis of the radar and another velocity on the other side.

Dataset	***”Wheel” labeled detections***	***Detections from clusters with 2 velocities***	***”Wheel” labeled detections with 2 velocities***
1	139	157	52
2	117	67	26
3	319	69	54
4	126	93	4
5	154	41	43
6	22	80	0
7	254	45	32
8	165	35	39
9	17	10	0
10	94	35	30
11	25	23	0
12	156	108	88

Table 5.9: Comparison of detections from clusters with two velocities and detections labeled as wheels in the reference data for each dataset. The second column shows the total number of detections with label ”Wheel” for the reference data, and the third column shows the number of detections with two velocities in the estimated data. The fourth column shows the intersection of the second and third column.

To analyze the velocity estimation for cluster assigned with one velocity, 200 iterations were run and the percentual change between each iteration was examined. Figure 5.12 shows the average percentual change in the estimated velocity between iterations in RANSAC. Figure 5.13 shows the same plots, but zoomed in. Note that the axes for the longitudinal and lateral velocity have different scales. Figure 5.14 shows the median of the percentual change. In the upper and lower plots of the figures the longitudinal and lateral change is illustrated respectively. It can be observed that the fluctuation in the lateral direction is greater than in the longitudinal direction. This is probably due to the small values of the lateral velocity compared to the longitudinal velocity, making the lateral velocity more sensitive. Vehicles on the highway move in a longitudinal direction, hence the result is expected and indicates that the estimated velocities are reasonable.

Further, as in previous results regarding sensitivity, precision and segmentation, it is visible that the median is lower than the mean. Again, this is due to a few extreme values.

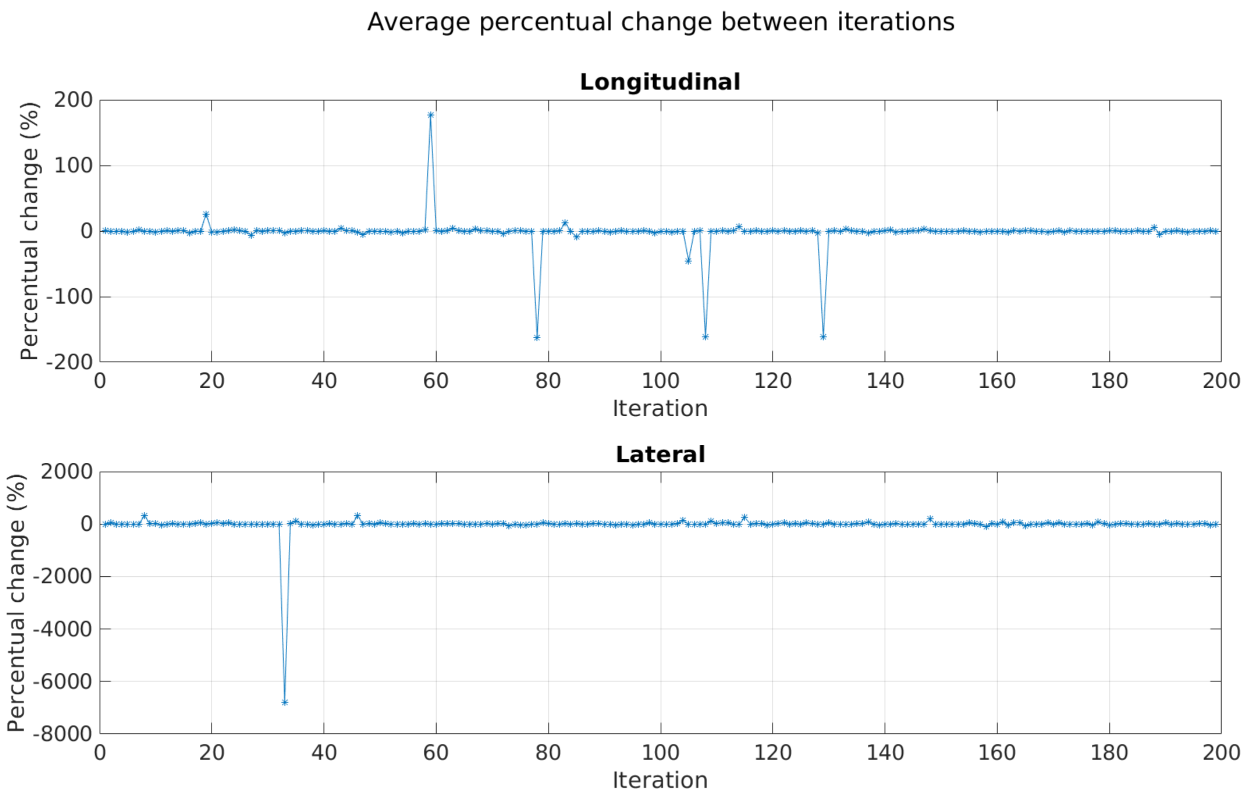


Figure 5.12: Plot showing the average percentual change of estimated velocity between 200 iterations in RANSAC. The upper plot shows the percentual change in the longitudinal direction and the lower shows the lateral.

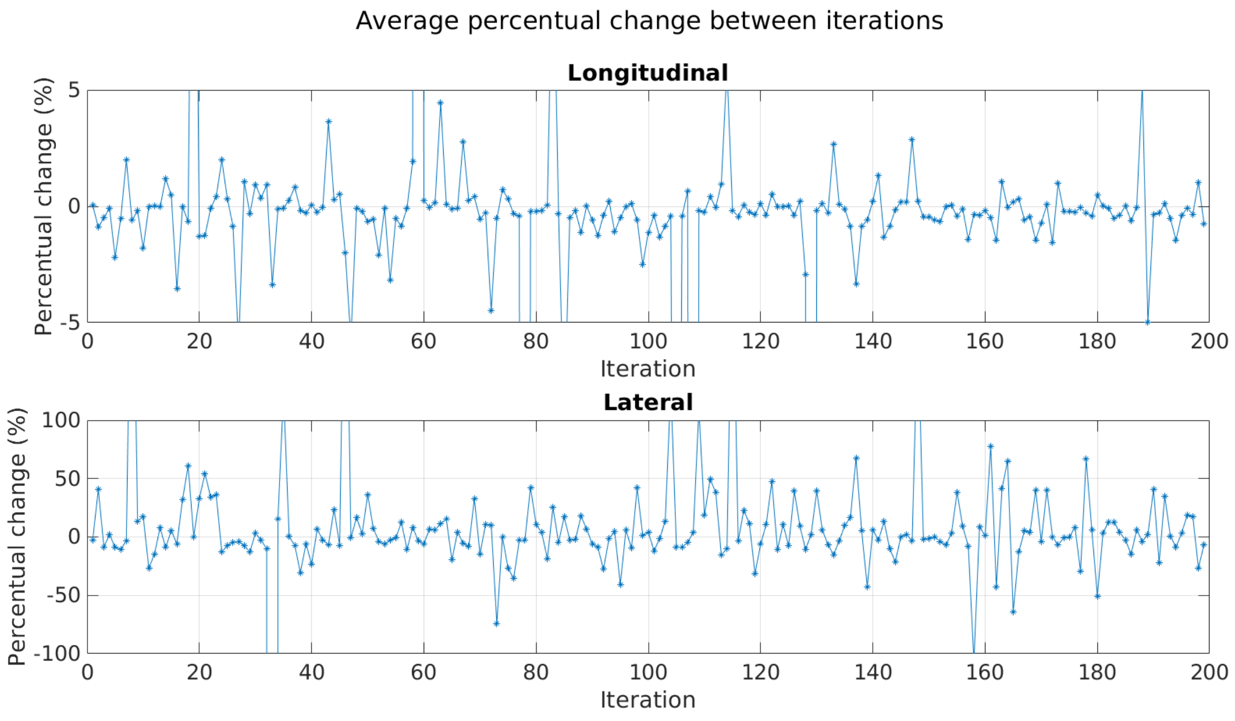


Figure 5.13: Plot showing the zoomed average percentual change of estimated velocity between 200 iterations in RANSAC. The upper plot shows the zoomed percentual change in the longitudinal direction and the lower shows the lateral. Note that the axes for the longitudinal and lateral velocity have different scales.

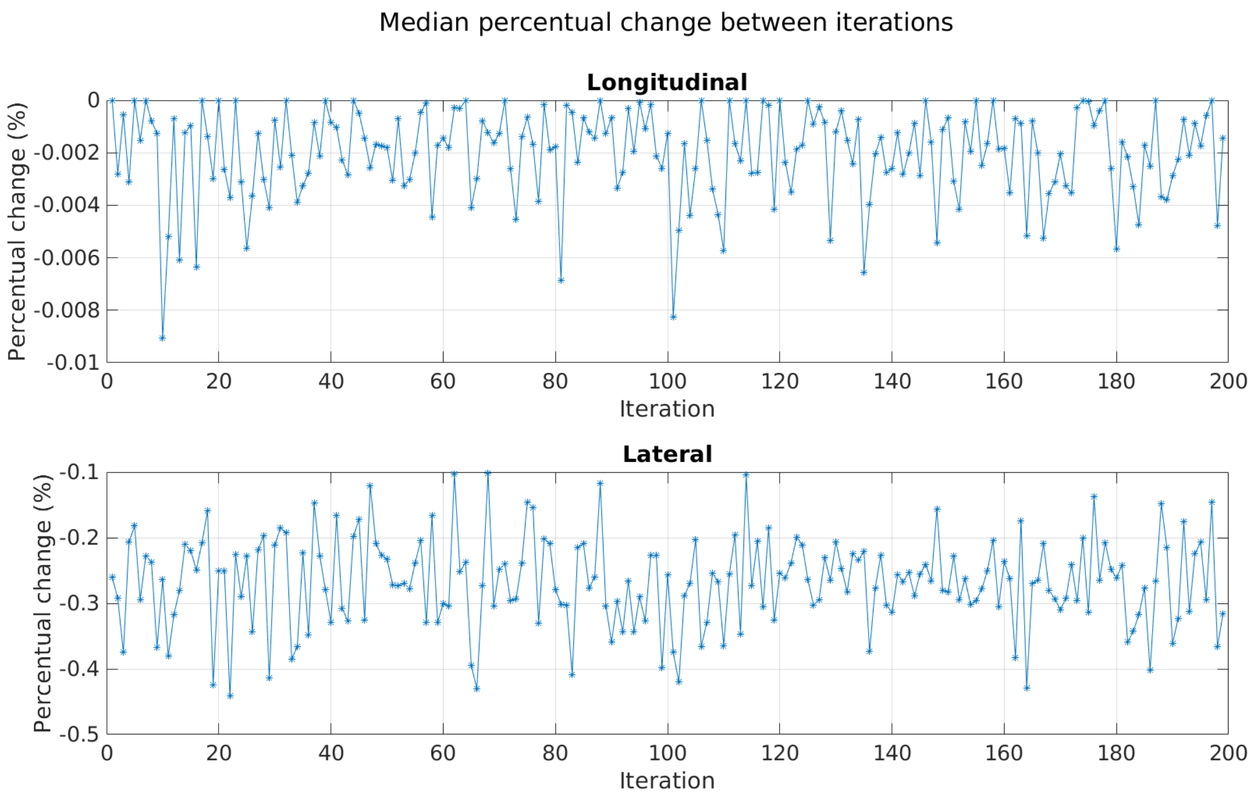


Figure 5.14: Plot of the median percentual change of estimated velocity between 200 iterations in RANSAC. The upper and lower plots shows the percentual change in longitudinal and lateral direction respectively.

6
Future work

In this thesis, there have been limitations when it comes to the data and the constraints that have been used in the proposed algorithm. This chapter will discuss suggestions for future work based on these limitations.

Using the approach of decreasing the number of detections for each step that is taken in the algorithm shows promising results. However, the algorithm could be further improved by examining more scenarios. The datasets used were limited to scenarios on highways and the number of time frames was also limited since the annotation of the data was done manually, which was time consuming. In general, tests with a larger dataset, both in terms of traffic scenarios and in terms of number of time frames could be executed to further investigate the performance and improve the proposed algorithm. For instance, more intricate scenarios in urban environments could be included. For highways in particular, where targets typically are aligned with the ego vehicle, i.e, have a longitudinal spread, the algorithm could be improved by using a static ellipse, with the major axis aligned with the ego vehicle's longitudinal direction, as search area instead of a circle.

All the scenarios considered in this thesis took place on a highway. Hence, it was assumed that the L-shapes were longitudinally parallel to the ego vehicle. However, e.g., in traffic jams, cut ins can occur at close distances. Then, when a vehicle next to the ego vehicle changes lane, the L-shape will not be parallel to the ego vehicle. Therefore, this constraint should be reviewed, for instance by using Search based rectangle fitting [21].

Further, the variable $hMax = 8$, i.e., the length of the horizontal line in the L-shape, is quite restrictive when considering a truck. However, as briefly discussed in Section 5.1, increasing $hMax$ to suit the length of a truck could lead to problems when estimating an L-shape for a car. A solution for this might be to have two different models for estimating L-shapes for cars and trucks, respectively.

Regarding the velocity, the motion of the ego vehicle should be compensated for. This was not done in this project due to lack of time. Furthermore, Table 5.9 shows potential in finding vehicles and should be investigated further. It could be interesting to investigate how the velocity might be used to adjust the obtained clusters, something that was discussed during the thesis but was not carried out due to lack of time.

Finally, another way of improving the clustering could be to use the amplitude of the detections. This could be interesting to incorporate in future work.

7

Conclusion

In this thesis, a clustering algorithm was developed, which uses DBSCAN to cluster detections and RANSAC to cluster detections and extract shapes in one step. The approach that the algorithm has used is to decrease the number of detections in a set with all detections from a time frame. The proposed algorithm has three blocks that follows the pattern: creating a subset of detections from the list with all detections, using the subset as input to RANSAC and save inliers to the final result. After these three blocks, the algorithm uses DBSCAN to cluster remaining detections, one last time.

The approach of decreasing the number of detections for each step in the algorithm shows promising results. The sensitivity and the precision takes quite high values, however, the segmentation can be improved. For highway scenarios, the results could be improved by using a static ellipse with the major axis aligned with the ego vehicle as a search area, instead of a static circle. If the ellipse works better than the current solution, with a static circle, the amount of correct clusters should increase and the over- and undersegmentation decrease. This is because the high sensitivity, on average 92.28%, indicates that 7.72% of the detections are incorrectly clustered such that the segmentation is increased, as mentioned at the end of Section 5.2.2.

Further, DBSCAN and RANSAC have been two suitable and promising methods. DBSCAN has been straightforward to implement and RANSAC has provided a versatile framework, since it is possible to estimate different models.

Finally, the analysis of the velocity estimation showed that the velocity estimation between iterations did not change much, indicating that velocity could be used to further adjust clusters. Investigating the velocity also showed that it is possible to use the velocity to find vehicles, but it needs to be further developed.

Bibliography

[1] European Commission. Intelligent transport systems, 2018. `https://ec.europa.eu/transport/themes/its_en` [Online; accessed 2-May-2018].

[2] B. Kulcsár A. Dabiri. Intelligent transport systems. Chalmers University of Technology, August 2012.

[3] Y. Gu S. Kamijo E. Javanmardi, M. Javanmardi. Autonomous vehicle self-localization based on probabilistic planar surface map and multi-channel lidar in urban area. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, Oct 2017.

[4] G. Pandey J. R. Mcbride A. Pensia, G. Sharma. Fast localization of autonomous vehicles using discriminative metric learning. In *2017 14th Conference on Computer and Robot Vision (CRV)*, pages 176–182, May 2017.

[5] J. Gong Z. Yong H. Chen H. Di W. Mei, G. Xiong. Multiple moving target tracking with hypothesis trajectory model for autonomous vehicles. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017.

[6] F. Sandblom J. Sorstedt L. Hammarstrand, L. Svensson. Extended object tracking using a radar resolution model. *IEEE Transactions on Aerospace and Electronic Systems*, 48(3):2371–2386, JULY 2012.

[7] A. Forsgren V. Nordenmark. Radar-detections based classification of moving objects using machine learning methods. Master's thesis, KTH.

[8] R. T. Hill. Airborne radar. *AccessScience McGraw-Hill Education*, 2014.

[9] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer Science+Business Media, LLC, Spring Street, New York, 2006.

[10] J. Sander X. Xu M. Ester, H-P. Kriegel. A density-based algorithm for discovering clusters in large spatial databases with noise. *AAAI*, 1996.

[11] K. Dietmayer D. Kellner, J. Klappstein. Grid-based dbscan for clustering extended objects in radar data. *IEEE Intelligent Vehicles Symposium*, 2012.

[12] C. Piech. K means. Stanford, 2013. `http://stanford.edu/~cpiech/cs221/handouts/kmeans.html` [Online; accessed 8-June-2018].

[13] S. Varshney P. Arora, Dr. Deepali. Analysis of k-means and k-medoids algorithm for big data. 2015.

[14] R. Myilsamy. A comparative study of robust ransac techniques. 2:227–232, 2012.

[15] M. Pollefeys R. Raguram, J. M. Frahm. Exploiting uncertainty in random sample consensus. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2074–2081, Sept 2009.

[16] R. C. Bolles M. A. Fischler. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[17] K. Dietmayer J. Klappstein J. Dickmann D. Kellner, M. Barjenbruch. Instantaneous lateral velocity estimation of a vehicle using doppler radar. In *Proceedings of the 16th International Conference on Information Fusion*, pages 877–884, July 2013.

[18] R. Jörnsten. Statistical learning for big data – lecture 5. Mathematical Sciences University of Gothenburg and Chalmers University of Technology, April 2017.

[19] K. Granström S. Yang, M. Baum. Metrics for performance evaluation of elliptic extended object tracking methods. In *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 523–528, Sept 2016.

[20] Volvo Trucks AB. Volvo fh specifications. `https://www.volvotrucks.se/sv-se/trucks/volvo-fh/specifications/data-sheets.html` [Online; accessed 8-June-2018].

[21] C. Dong J. M. Dolan X. Zhang, W. Xu. Efficient l-shape fitting for vehicle detection using laser scanners. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 54–59, June 2017.

A

Tables

Radar	***Average running time (s)***	***No of time frames***
1	0.4304	56
2	0.4486	52
3	0.3506	31
4	0.4033	25
5	0.3487	25
6	0.3492	33
7	0.4309	80
8	0.2220	47
9	0.3100	28
10	0.4390	16
11	0.3260	28
12	0.5215	64

Table A.1: Average run time per radar of the algorithm displayed in Figure 3.1.

Test set number	1	2	3	4	5	**Average of all tests**
Average sensitivity	0.92	0.92	0.92	0.93	0.93	0.92
Median sensitivity	1.00	1.00	1.00	1.00	1.00	1.00
Average precision	0.83	0.86	0.86	0.86	0.84	0.85
Median precision	0.93	0.94	0.94	0.95	0.92	0.94
Average oversegmentation	0.41	0.38	0.38	0.37	0.40	0.39
Median oversegmentation	0.33	0.33	0.33	0.25	0.33	0.32
Average undersegmentation	0.22	0.26	0.26	0.24	0.21	0.24
Median undersegmentation	0.00	0.00	0.00	0.00	0.00	0.00

Table A.2: Average and median sensitivity, precision, oversegmentation and undersegmentation for five test sets, respectively. The last column shows the average of each row.