CHALMERS
UNIVERSITY OF TECHNOLOGY

LiDAR Clustering and Shape Extraction for Automotive Applications

Master's thesis in Systems, Control and Mechatronics

TOBIAS NYSTRÖM JOHANSSON

OSCAR WELLENSTAM

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

Master's thesis 2017:EX 107

LiDAR Clustering and Shape Extraction for Automotive Applications

TOBIAS NYSTRÖM JOHANSSON
OSCAR WELLENSTAM

Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2017

LiDAR Clustering and Shape Extraction for Automotive Applications
TOBIAS NYSTRÖM JOHANSSON
OSCAR WELLENSTAM

© TOBIAS NYSTRÖM JOHANSSON, OSCAR WELLENSTAM, 2017.

Supervisors: Abu Sajana Rahmathullah and Daniel Svensson, Zenuity AB
Examiner: Fredrik Kahl, Department of Electrical Engineering

Master's Thesis 2017:EX 107
Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2017

LiDAR Clustering and Shape Extraction for Automotive Applications
TOBIAS NYSTRÖM JOHANSSON
OSCAR WELLENSTAM
Department of Electrical Engineering
Chalmers University of Technology

Abstract

In research related to autonomous vehicles, interest has recently fallen on the use of multi-layer laser scanners, also known as lidars. They are used with the intention to better detect and track vehicles, pedestrians and other objects on the road. A lidar receives reflections from objects in the field of view of the sensor and these detections are usually represented as a point cloud in 3D coordinates. One major challenge associated with the use of this data in real time applications is the need to reduce the complexity of the data in an efficient way. In this thesis, an algorithm is proposed to process the data obtained by the lidar by reducing the point cloud to geometric descriptors, or shapes. The proposed algorithm clusters data from each vertical scan layer of the lidar individually. Each cluster is then assigned and fitted with one of four possible shapes, namely L-shapes, lines, polygons or points. Clusters are subsequently merged based on their proximity, but with restrictions depending on their shapes to avoid undersegmentation. The proposed algorithm then fits a new shape to each merged cluster, which ideally represents a single object. Quantitative evaluation using a reference vehicle and manual assessments show promising results when compared with a reference algorithm. The proposed algorithm is proficient in distinguishing between vehicles and ground points, as well as providing accurate shape descriptors. On the other hand, both algorithms show decreased performance at certain orientations of an object relative to the lidar. The execution time of the proposed algorithm is considerably higher than the reference algorithm, but several optimisations are suggested which would greatly improve run time.

Keywords: multi-layer lidar, laser scanner, segmentation, clustering, feature extraction, shape extraction, preprocessing, object tracking, real-time, perception

Acknowledgements

We would like to thank Zenuity for making this thesis possible. Special gratitude is extended to our supervisors Abu Sajana Rahmathullah and Daniel Svensson, for providing helpful guidance and assistance throughout the project. We would also like to acknowledge the rest of the Target Tracking team at Zenuity for being helpful and welcoming in every situation.

Finally we would like to thank Prof. Kahl at the Department of Electrical Engineering for his support and advice.

Tobias Nyström Johansson and Oscar Wellenstam, Gothenburg, November 2017

Contents

List of Figures

List of Tables

1 Introduction

Research on autonomous vehicles is a topic of significant interest worldwide. The prospect of being able to spend commuting time reading the news, sending emails or even sleeping is appealing compared to actively controlling a vehicle, but the advantages that autonomous driving could offer exceeds that of human comfort. According to data published by the World Health Organization, about 1.25 million people worldwide are killed in traffic accidents each year [6]. Human errors are estimated by the National Highway Traffic Safety Administration (NHTSA) to account for 94% of all traffic deaths in the US [7], and by allowing cars to navigate and maneuver autonomously this effect can be mitigated. There is also speculation on other ways the introduction of autonomous cars can benefit society, for example by changing the ownership model of cars to allow for cheaper transportation, and saving space in cities where less parking spots will be required if the degree of car ownership decreases.

1.1 Background

In order for an autonomous vehicle to make correct decisions, it is critical that it is able to accurately assess surrounding traffic, roads and obstacles. This assessment is obtained by analysing data from a large number of sensors mounted on the vehicle. Some of the most commonly used include radar, lidar and camera. A fully autonomous vehicle would include different types of sensors to supplement each other and through sensor fusion a more reliable picture of the surroundings of the vehicle is possible than using a single type of sensor. The different sensors generally have unique strengths and weaknesses. A radar uses microwaves to determine distance and speed of an object as well as the angle to the object. It is typically accurate at determining the distance and speed, however the angular accuracy is proportional to the angular resolution of the radar which often is rather poor. A camera sensor takes pictures which are then processed using several image processing algorithms.These are exceptional at classifying and distinguishing between different objects in images, but uses large amounts of processing power. Furthermore, camera sensors are typically good at measuring the angle to an object, but are poor at distance measuring. Using a camera sensor in poor lighting conditions will also reduce its performance. A lidar uses laser to measure distances, and can depending on the type of sensor generate a precise 2D or 3D map of a vehicle's surroundings. A lidar tends to generate large amounts of data which is typically computationally expensive to process. The handling and processing of lidar data is investigated in this thesis.

The simplest version of a lidar sends out a laser pulse in one known direction, and then detects reflections from objects in its path. The reflected signals are used to measure distances to objects in those known directions, but for it to be useful in autonomous driving it needs to be able to detect objects in more than one direction. The solution that is implemented in automotive lidars is to have one or more actuated sensors in a single lidar system. A 2D lidar is a system in which the scanners sweeps over different angles, allowing objects in many directions to be covered by the sensor. There are also even more complicated lidar systems, the most well known that are suitable for use in automotive applications are the so called 3D lidars [8, 9, 10]. They have a large number of scanners placed in a vertical line, where each sensor is also referred to as a scanning layer. Each sensor is mounted with slightly different elevation angles and work in unison to obtain a point cloud of reflections so dense that it approximates a 3D image of the covered area. The system is commonly used in autonomous driving research and experiments, with the sensor mounted on the roof of the car. A well known manufacturer of such systems is Velodyne, which in its portfolio of sensors have a high resolution 3D lidar with 64 layers [9]. Due to significant costs in acquiring such systems, other automotive lidars are used that contain considerably fewer scanning layers. In this thesis, such lidars with up to eight scanning layers are considered. Such lidars are herein referred to as $2D^+$ lidars.

The major difference between the $2D^+$ and 3D LiDAR sensors is in the amount of data they produce and consequently, in the computational complexity associated with analysing them. A high resolution 3D lidar produces over two million points per second whereas a typical $2D^+$ lidar produce around a hundred thousand points per second. One car maker has to date announced an autonomous driving function in a new car model which utilises a $2D^+$ lidar among other sensors [11]. It is believed that lidar will play an important role in the sensor set of future autonomous vehicles, as most autonomous driving projects utilise $2D^+$ or 3D lidars as part of their sensor set [12].

As mentioned before, besides the lidar, the autonomous vehicle will also use other sensors to have an accurate and reliable perception of the environment. A popular way to achieve this is to combine information from multiple sensors sampled over time in an inference or estimation algorithm, commonly known as tracking filters. The algorithms typically provide prediction estimates of the current state variables using some motion model. Thereafter, the estimate is updated using measurements from sensors in some measurement model. Bayesian filters are based on Bayesian statistics and include Kalman filtering and variations thereof such as extended Kalman filtering, unscented Kalman filtering and particle filtering. Here data from different types of sensors can be analysed together resulting in an overall improved estimation. [13]

Radar data is very different from that of camera vision systems, which in turn looks different to the data obtained from lidar. An approach to combining the data from these sensors in a filter is to first preprocess the data. For both radar and lidar, preprocessing brings down the computational complexity dramatically since a single object can cause many detections. In the case of $2D^+$ lidars, trying to perform filtering on an unprocessed data point cloud would result in a very high computational cost.

1.2 Purpose

The purpose of this work is to design an algorithm to efficiently and accurately preprocess $2D^+$ point cloud data from a multi-layer LiDAR, and produce simplified geometric descriptors, or shapes, that can be used in Bayesian target tracking algorithms.

There are two fundamental challenges that the algorithm thus needs to solve. First, it should provide some clustering of the lidar detections that reliably distinguish detections from different objects, such that each cluster represents one object. Secondly, it should classify each cluster to assign it a representative shape which should subsequently be fitted to the cluster.

The algorithm shall be designed to suit a lidar mounted in the front bumper of a car, and real-time processing should be kept in mind when designing the algorithm. The geometrical descriptors which the algorithm shall produce is limited to the primitive, but for vehicles descriptive, shapes of lines, L-shapes, points and polygons. Information about the quality of the descriptors should be retained in order to allow for estimation of an uncertainty measure that could be used in a target tracking algorithm, which will have to weigh the inputs from the lidar against other sensor data.

1.3 Related Work

Research related to lidar data processing is extensive and quickly growing, which makes it challenging to provide a comprehensive overview of the area. In this section some related research will be presented which ranges from complete Detection and Tracking of Moving Objects (DATMO) algorithms to work aimed at improving underlying algorithms for clustering, shape extraction and tracking.

Several papers propose algorithms for shape extraction of 2D and $2D^+$ lidar data. In [14], Wender et al. distinguish different types of shapes for each layer of the multi-layer lidar, after which the orientation and shape of each layer is combined using some weighting to form a best estimate. It is later expanded upon in [15], where they use a support vector machine (SVM) and a Kalman filter to subsequently classify and track the detected objects. However, the shapes that are calculated, are designed for speed, and the algorithm does not always give accurate results.

A subproblem in DATMO with a lidar is how to perform segmentation on the data provided by the sensor. For 2D and 2D$^+$ lidars, a multitude of different methods exist. The method most often recurring in previous research is implementation of some version of breakpoint detection in order to partition data [16, 3, 17, 18]. In addition, approaches involving density-based spatial clustering of applications with noise (DBSCAN) include [19, 20].

In [20], DBSCAN is used to cluster points from several lidars, after which each cluster is classified as one of several shape types. Additionally, a Kalman Filter is implemented to track the detected objects. However the algorithm is slow and not suited for real time implementation.

In [17], Kim et al propose methods for robust clustering of multi-layer lidar data. They formulate and use an adaptive breakpoint algorithm which clusters multiple layers of data simultaneously, and attempt to avoid ground detections using some heuristics. This clustering method however risks causing undersegmentation when different objects are observed on the different scan layers, for example ground points detected under a vehicle. The authors later designed a method for classifying car and pedestrian data point clusters using an SVM [21], trained on manually labelled data.

Certain research specifically focus on pedestrian detection and tracking [22, 23]. In [23], Gidel et al. present a method of real-time detection and tracking of pedestrians by means of a four-layer lidar. To fuse the data from several layers they propose a Parzen Window kernel method. In [22], Gate et al. suggests a method to recursively estimate the true outline of pedestrians to improve their classification from lidar data and subsequently their tracking as well.

Several line extraction algorithms for use on 2D lidar data in SLAM applications are presented in a comparison by Nguyen et al. [24], of which the split-and-merge algorithm was singled out as having favourable performance with respect to speed and accuracy. The algorithm fits a line to a set of points and perform successive splits of the line at points which have a maximum normal distance to the line segment. The method however suffers from sensitivity to noise when choosing a splitting point. An algorithm inspired by split-and-merge, which aims at mitigating the aforementioned sensitivity to noise, was suggested by Magnier et al. [25]. The algorithm, called, the recursive best segment split (RBSS), selects the splitting point that minimises the sum of variances instead of the one with a maximum normal distance.

In [16] and [3], methods for extracting lines from data obtained from 2D and 2D$^+$ lidars are proposed for SLAM-applications, but contains clustering and shape extraction algorithms which are relevant for automotive applications.

1.4 Scope and Limitations

The algorithm developed in this work is constrained to implementation in MATLAB on a PC. Even though real-time performance is essential for the application of the algorithm, there are no thorough evaluations on performance in an optimised coding language and on specialised hardware.

The preprocessing must be performed frame by frame and may not have time dependencies.

Extracting shapes which provide accurate representation of dynamic objects detected on the road is considered most important, with representations of vehicles and pedestrians having priority over barriers and lane markings. The shapes will be extracted in 2D, representing a view of the environment as seen from directly above in the xy-plane using the coordinate frame presented in Chapter 2.

The algorithm is tested in a limited number of selected scenarios encountered in log-data, and is not evaluated for a large number of generic traffic situations. Even though care is taken to select relevant and challenging scenarios for evaluation of the algorithm, performance in a real world setting with diverse situations are not thoroughly assessed. Focus is on situations that occur on delimited highway roads in good weather conditions.

1.5 Contribution

The main contributions of this thesis are:

- An extended dual breakpoint detection clustering method which allows for clustering of sequential scan points depending on a distance or an angle-based criterion, as described in Section 4.2.2.
- A collection of criteria for classifying polygons, lines and L-shapes from clustered data, presented in Section 4.2.3.
- A group of criteria for selectively merging clusters in close proximity depending on the shape associated with them, which is elaborated on in Section 4.2.4.

1.6 Thesis Outline

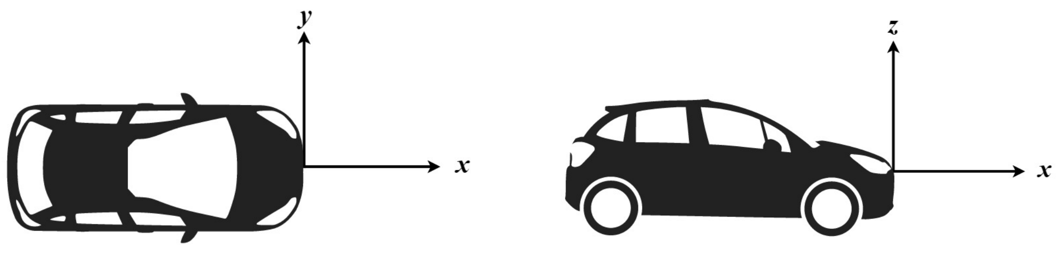
The outline of the thesis is as follows: Chapter 2 provides an overview with respect to how data from a multi-layer lidar is obtained. In Chapter 3, a theoretical background to clustering and shape extraction is presented. Next, the design of the algorithm and methods for evaluation are given in Chapter 4. In the subsequent Chapter 5 the results of the evaluation are presented. Afterwards, a discussion on the performance of the proposed algorithm compared to the reference algorithm is provided in Chapter 6. Finally in Chapter 7 conclusions and future work are presented.

2
System Overview

The experimental setup is described in this chapter. First, in Section 2.1, the coordinate system is presented. Secondly, in Section 2.2, a detailed description of multi-layer lidar data is provided.

2.1 Coordinate System

The system consists of a vehicle with a lidar mounted approximately in the center of the front bumper. The coordinate system used is placed with the sensor in the origin, as can be seen in Figure 2.1, where the x-axis represents longitudinal coordinates, the y-axis represents lateral coordinates and the z-axis represents the heights.



(a) (x, y) **(b)** (x, z)

Figure 2.1: The (x, y) plane and (x, z) plane with origin at the location where the sensor is mounted. Images are derivative work of [1] and [2].

2.2 Multi-layer Lidar

A lidar works by emitting a sequence of laser pulses in different azimuth and elevation angles, and detecting the reflections of these pulses from different objects on the road and from the road itself. Time of flight is then used to determine the distance to each point, which combined with the azimuth information provides a Cartesian coordinate representation of each point. In this case the lidar is mounted on the front bumper of a car as illustrated in Figure 2.2.

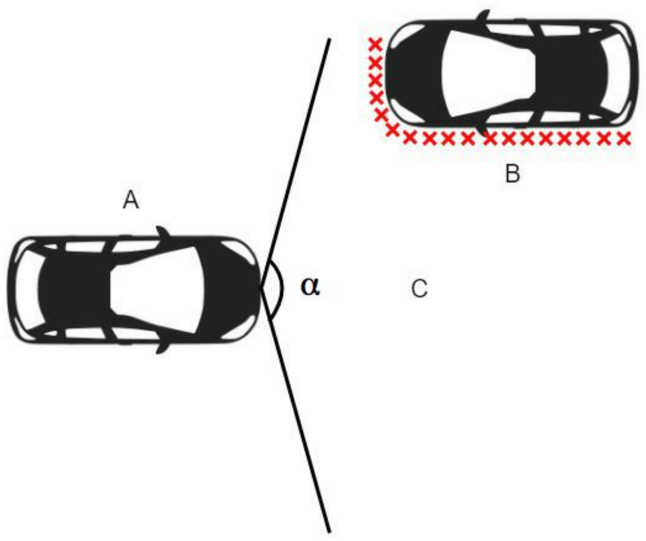


Figure 2.2: Top-view of a lidar mounted in the front bumper. A) Ego vehicle. B) Lidar detections of oncoming vehicle. C) Lidar field of view α, typically between 120° and 150°. The image is a derivative work of [1].

The laser pulses are emitted sequentially over the different azimuths, see Figure 2.3, and elevation angles, see Figure 2.4, which combined with the distance to the target gives information on the 3D position of a detected point.

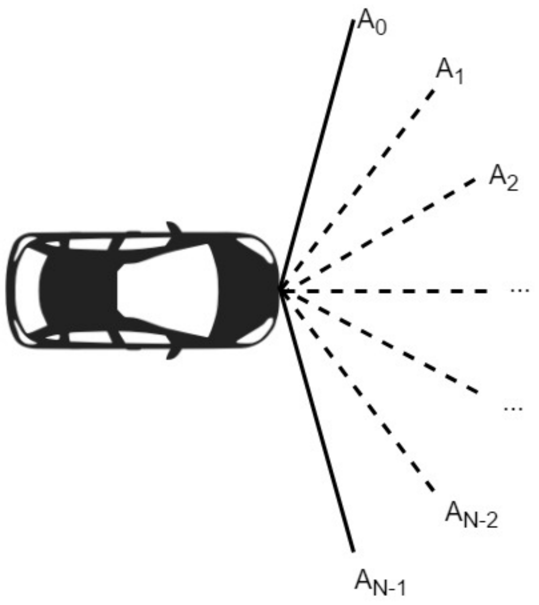


Figure 2.3: The horizontal field of view is divided into N azimuth directions $A_0, A_1, ..., A_{N-1}$ of equal spacing. Here N is sufficiently large and the angular resolution is $< 1°$ giving a total field of view of α in Figure 2.2. The image is derivative work of [1].

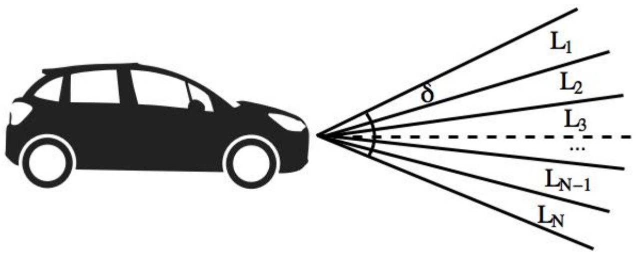


Figure 2.4: N elevation angles $(L_1, L_2, L_3, \ldots, L_{N-1}, L_N)$ of equal spacing divide the vertical field of view in slices of width, δ, giving a total vertical field of view of $N\delta°$. For a 2D$^+$ lidar, $N \in [2, 8]$ and typically $\delta < 1$. The image is a derivative work of [2].

The data is presented as a 3D point cloud in the (x, y, z) coordinate system presented in Figure 2.1, and consists of reflections from static surroundings such as the road and traffic signs as well as dynamic objects such as cars and pedestrians. Points are ordered sequentially according to the azimuth and elevation scanning order. The data is sampled at a frequency between 20Hz and 100Hz, each sample containing data from every azimuth over all active elevation angles. The active elevation angles can in some cases alternate between the different frames, but is known and predictable so that the resulting 3D point cloud can still be accurately determined. Some lidars can detect multiple echoes originating from the same laser beam. For example multiple echoes can occur, when there are multiple reflective surfaces contained in a single azimuth and elevation angle, but at different ranges. Multiple echoes can also be caused by a laser beam scattering over multiple objects or in poor weather conditions. The lidar used here is able to detect several echoes.

3 Theory

In this chapter the theoretical backgrounds of the clustering and the shape extraction methods are presented in Section 3.1 and Section 3.2, respectively. Section 3.1 focuses on clustering for automotive applications with lidar. Section 3.2 is focused on methods to generate the geometric shape descriptors that are considered in the thesis, namely lines, L-shapes and polygons. For polygon extraction, the first step can be the formation of convex or concave hulls. Therefore, methods to extract those hulls are also described.

3.1 Clustering

Clustering of data is an extensive area of research and covers many disciplines. The underlying purpose is to separate a finite set of unlabelled data points into sets of discrete data structures [26]. Cluster analysis in broad terms, as defined by Jain et al. [27], is the organisation of a set of data points into clusters, with respect to some similarity. An interpretation of this is that the data points generated by a clustering algorithm are more similar to each other than to points in any other cluster. Measuring similarity is not a trivial task and depends partially on the features of the data and how easy these features are to identify and extract.On a high level, clustering methods can be divided into hierarchical and partitional approaches [27]. a hierarchical clustering method, each data point is organised in a tree structure according to similarity between all data points. When the tree structure is created, clusters are formed by determining the maximum dissimilarity allowed between two data points of the same cluster and dividing the tree accordingly. A partitional method, on the other hand assigns each data point to a cluster without retaining information on similarity to every other point. This saves the computational effort of creating the similarity tree used in hierarchical methods, but retains less knowledge about the clusters formed. For lidar data represented as points in Cartesian coordinates, two features that could be used for similarity are the distances between the points, and the local distribution of points. Furthermore, in 2D and 2D$^+$ lidar data, the sequential nature of data is usually available, which provides an additional aspect to consider for clustering.

To process raw lidar data for automotive applications, a distinct difference between the approaches found in previous research is whether the sequential information is used in the clustering, or whether the data is treated as a set of unordered points. For the first case of using ordered points, the most prevalent methods of clustering are based on breakpoint detection [16, 3, 17, 18], which mainly utilises the similarity in-between consecutive points to construct the clusters.

For the second case of treating the points as unordered, clustering methods such as DBSCAN [28] can be used as is done in [19, 20]. The main advantage of treating the data as unordered is that one can combine data from different sensors before any clustering is performed, which simplifies the method and diminishes the need for post-clustering data association.

3.1.1 Breakpoint Detection

In breakpoint detection the points are clustered together automatically if they are neighbours in the sequence , unless the distance between consecutive points p_{n-1} and p_n surpass a certain threshold $D_{\max}$,

$$\|p_n - p_{n-1}\| > D_{\max}. \tag{3.1}$$

If the distance between the points exceeds the threshold $D_{\max}$, a new cluster will be started using the current point p_n. That current point will be considered the breakpoint between two clusters. This principle is illustrated in Figure 3.1.

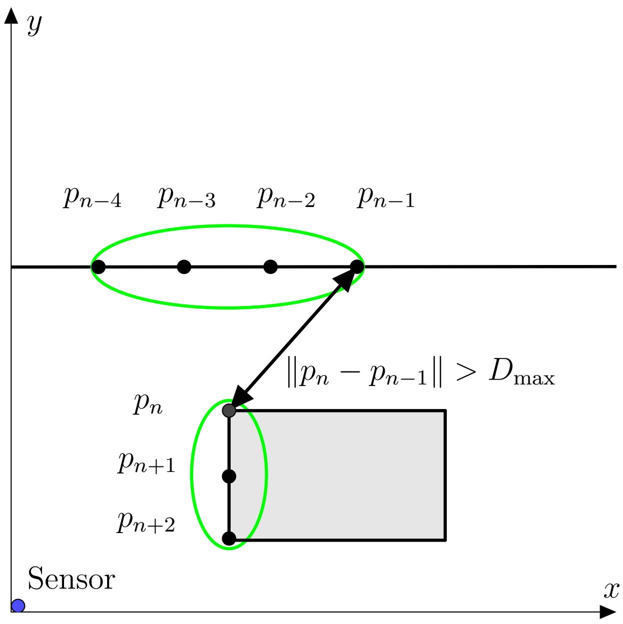


Figure 3.1: A sequence of points, p_{n-4} through p_{n+2} constitute detections on two objects. The first four points, p_{n-4} to p_{n-1} are clustered since the distance between each consecutive point is smaller than the threshold $D_{\max}$. However, the distance between the next point on the box object p_n and p_{n-1} on the line object is larger than $D_{\max}$, implying that p_n is a breakpoint and thus the start of a new cluster which contain points p_n to p_{n+2}.

The main challenge with breakpoint detection is to find a suitable distance threshold $D_{\max}$, as there are several factors which can affect the distance between two points

$\|p_n - p_{n-1}\|$. Most importantly, the lidar measurements are sparser as the range to the detections increases, which causes problems when keeping a fixed threshold.

3.1.1.1 Adaptive Breakpoint Detection

A way of adapting the threshold $D_{\max}$ in Section 3.1.1 to the range of the detection was suggested by Borges et al. [3] and is known as Adaptive Breakpoint Detection (ABD). The key to the ABD threshold is a methodology presented in Figure 3.2, where a virtual line is passing through a scan point p_{n-1}. To determine the virtual line, a parameter λ is set, corresponding to the angle with respect to the scanning angle ϕ_{n-1}. Furthermore λ can be interpreted as the worst case incidence angle for a line on which points can be reliably detected.

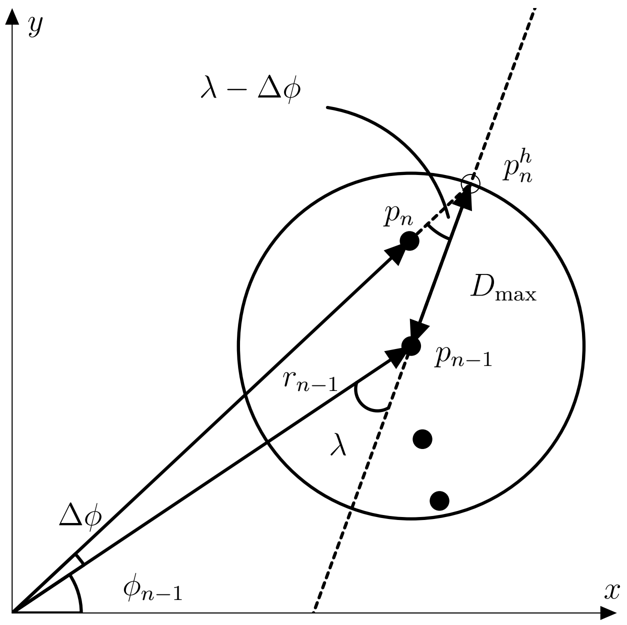


Figure 3.2: The ABD threshold $D_{\max}$ as described by Borges et al. [3] is the distance between the previous scan point p_{n-1} and the hypothetical scan point p_n^h. The distance is calculated by using the law of sines with respect to the sides of triangle created by the origin, p_{n-1} and p_n^h, and the inner angles $\Delta\phi$ and $\lambda - \Delta\phi$. Any current scan point p_n, outside the threshold circle will be considered a breakpoint.

For detections on the surface of an object, the incidence angle θ describes how that surface is oriented in relation to the sensor. For a rectangular object, each side that is visible to the sensor has an incidence angle which relates to how densely the scan points can be expected to lie. The incidence angle is the smallest angle between two vectors v_1 and v_2, where v_1 is a vector from the sensor to the detected point and v_2 is a vector parallel to the side of the object which the point is detected on. An illustration of the incidence angle can be seen in Figure 3.3. The ideal case is when the incidence angle of one side is close to 90°, as the distance between two successive points will only depend on the distance from the sensor to the point and the horizontal step size of the sensor. The worst case is when the incidence angle is close to 0°, since the distance between two successive points grows without limits [29]. A result is that the density of points can vary on different sections of a single object, as can be observed in Figure 3.3 where the side with the smaller incidence angle θ_2 has a sparser distribution of points than that of the side with a larger incidence angle θ_1.

The purpose of the worst case incidence angle λ is to extrapolate the largest allowed distance to a successive point p_n for the point to be included in the same cluster as p_{n-1}. The distance from this hypothetical scan point p_n^h to p_{n-1} as seen in Figure 3.2 will be the basis for the adaptive threshold, which is represented by a circle within which any successive point would be included in the current cluster. Using the law of sines for the triangle formed by the origin, p_n^h and p_{n-1} yields the following expression

$$\frac{\sin(\Delta\phi)}{\|p_n^h - p_{n-1}\|} = \frac{\sin(\lambda - \Delta\phi)}{r_{n-1}}, \tag{3.2}$$

where r_{n-1} is the range from the sensor to the point p_{n-1}. Equation 3.2 can be rewritten as

$$\|p_n^h - p_{n-1}\| = r_{n-1}\frac{\sin(\Delta\phi)}{\sin(\lambda - \Delta\phi)}. \tag{3.3}$$

Using this value as the threshold directly fails to take into account the noise associated with the range which means the threshold can become unstable at close distances to the sensor. In order to mitigate this, Borges et al. [3] further suggests adding a range measurement noise term $3\sigma_r$ to the threshold, where σ_r is distance resolution of the lidar. resulting in the following definition of the adaptive threshold

$$D_{\max} = \|p_n^h - p_{n-1}\| + 3\sigma_r. \tag{3.4}$$

One limitation with ABD is that it does not directly account for different incidence angles. The parameter λ which correlates to the incidence angle and determines all the distance thresholds using Equation 3.2 is predefined and unchanged regardless of the actual incidence angle of the measured surface. This makes tuning the parameter λ critical, as a large value will cause oversegmentation, and a small value will cause undersegmentation.

3.1.1.2 Dual Breakpoint Detector

A dual breakpoint detector (DBD) was proposed by An et al. in 2012 [16] for an application in simultaneous localisation and mapping (SLAM). The DBD is an extension of the ABD, which under certain circumstances allow clustering of points despite the distance-based criterion in Equation 3.1 being violated. If a cluster only contains one point, and the next point does not satisfy the distance-based criterion, the point is tested for an angle-based criterion. In the angle-based criterion, collinearity between the lines formed by three successive points p_{i-2}, p_{i-1} and p_i is considered. The need for this extension is illustrated in Figure 3.3 where one can observe that the distance between two successive points are larger on the side with a small incidence angle.

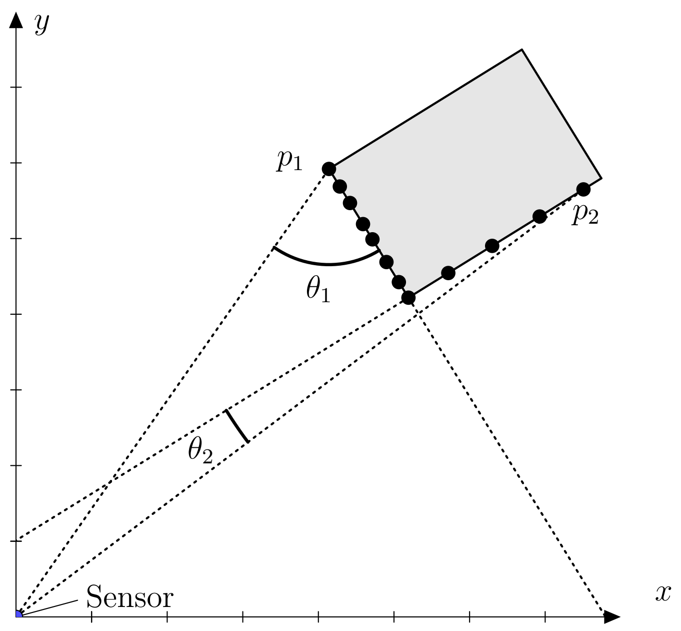


Figure 3.3: Incidence angles θ_1 and θ_2 corresponding to the two detections p_1 and p_2. The points are more sparsely scattered on the side with the smaller incidence angle θ_2 than the one with the larger angle θ_1.

The DBD classifies clusters of two types: distance-based clusters (DC) and angle-based clusters (AC). As is the case in ABD, any unclustered measurement is assigned to a new cluster and consecutive measurements are added if they satisfy the criterion in Equation 3.1 using the threshold obtained from Equation 3.4. A cluster may only be assigned one of the labels DC or AC, where the DC label has priority. If any point is added to the cluster using the distance-based criterion, it will be marked as a DC cluster and only the distance-based criterion will be considered for adding additional points to this cluster. If no points are added using this criterion, the angle-based criterion will be considered. The angle based criterion is written as

$$|\delta_i - \delta_{i-1}| < \theta_o \tag{3.5}$$

where δ_i is the angle formed by the consecutive points p_{i-1} and p_i and θ_o is a threshold. If the angle-based criterion in Equation 3.5 is satisfied, the cluster will be labelled an AC cluster and only the angle-based criterion will be considered for adding additional points to this cluster. If a cluster is labelled either a DC or an AC cluster and a consecutive point violates the respective criterion, the cluster is considered completed and a new cluster will be initialised with the next point in question. The process repeats until all points are assigned to a cluster.

3.1.2 Hierarchical Clustering

Hierarchical clustering is a family of clustering methods where the elements of a set are considered to be related up to a certain degree. A hierarchy of similarity is usually established for all points, and is subsequently used to form clusters. An advantage with hierarchical clustering over other popular clustering methods is that the number of clusters does not have to be known before initialising the algorithm. One drawback with hierarchical clustering is that it can be computationally expensive when the number of elements in a set is large, since the distance between every pair of elements must be calculated and stored. [26]

Single-Linkage Clustering

Single-linkage clustering is a hierarchical clustering method based on grouping elements and clusters together in an agglomerative fashion. Initially, every element has its own cluster and the distance between every cluster is computed. Starting with the clusters with the shortest distance between them, they are merged to form new clusters. The term 'single-linkage' refers to how the distance between clusters are updated after they have been merged. In this case the shortest distance between the elements of two clusters is considered. One way to perform the clustering is then to build up a complete dendrogram and cut it at an appropriate place to create clusters. One characteristic for single-linkage clustering is the presence of a chaining effect [30], which tends to create clusters that are elongated [27].

3.2 Shape Extraction

Shape extraction or feature extraction is used in several fields to generate descriptors in order to classify and quantify information [31]. Applications range from machine learning to image processing and pattern recognition, fields which have gained increased interest in the automotive industry as it strides towards developing fully autonomous vehicles. The immediate use of feature extraction techniques is in processing of data from sensors such as cameras, lidars and radars. Feature extraction concerning $2D^+$ lidar data mainly refers to extraction of 2D shapes related to objects on the road, road markings and barriers, which could be used in tracking algorithms and other perception based functions.

With focus on creating descriptors for objects on the road, Wender et al. [15] distinguish between shapes for objects that have a clear contour, e.g., L-shapes or I-shapes, and those that do not, e.g. O-shapes. Objects with a clear contour can be rectangular objects such as cars and trucks. These are often perceived as L-shapes by a multi-layer lidar. Barriers and lane markings on the other hand, are consistently recognised as lines (I-shapes). Objects with an unclear contour, for instance detections of pedestrians or ground points, can be perceived as singular points or polygons depending on size of the cluster. In contrast to L-shapes and lines, a polygon often does not provide information regarding the direction of the object. To incorporate more variations in the shape of clusters, Lösch [20] uses vertical and horizontal lines, L-shapes, U-shapes and O-shapes to characterise data from multiple lidars. The U-shape is designed to capture the shape of objects which due to the presence of multiple lidar have generated detections on three sides of an object.

In the following sections, an overview of methods to extract different geometric shapes is presented. The shapes considered are lines, L-shapes and polygons created from convex and concave hulls.

3.2.1 Line

Feature extraction of lines corresponds to linear regression in most cases, but the possibility of outliers necessitates careful consideration of which method that is used. The most well known method with respect to linear regression is least squares, which is however sensitive to ouliers. In contrast, a regression method more robust to outliers is the Theil-Sen estimator [32]. Other methods of extracting lines are utilising the sequential information about the points provided by a lidar. One such strategy is called iterative-end-point-fit and is performed by simply selecting the first and last point in the sequence to create the line representation. This method is used in some SLAM applications to build maps in real-time, and a related algorithm which is also used in SLAM is split-and-merge. Both of these methods are explained together with iterative-end-point-fit in a review by Nguyen et al. in [24].

Least squares regression

Least squares regression is a standard method of fitting a model to some data. It finds the minimum of the sum S of the square of the residuals $S = \sum_{i=1}^{n} r^2$, $r = y_i - f(x_i, \beta)$, where the residual r is the difference between the estimated value $f(x_i, \beta)$, for estimated parameters β, and the actual value y_i for a data point (x_i, y_i). A drawback with this method is its sensitivity to outliers in noisy data.[33]

Theil-Sen Estimator

The Theil-Sen estimator, first published by Theil [34] and expanded upon by Sen [32], is a regression method that is insensitive outliers. For a set of n points

$$(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)$$

the estimator $\hat{y} = \hat{k}x + \hat{m}$ is obtained by first calculating the slope of each possible pair of points and then taking the median slope to obtain $\hat{k}$. The corresponding line can then be calculated by the y intercept

$$\hat{m} = \tilde{y} - \hat{k}\tilde{x}, \tag{3.6}$$

where $\tilde{x}$ and $\tilde{y}$ are the medians of the x and y values in the data.

3.2.2 L-shape

The contours of vehicles in lidar data is most often represented by an L-shape, which can be used to estimate its size as well as the heading of the object. In the literature there are multiple ways of solving the problem of extracting L-shapes from clusters. Wender et. al [15] suggested a method where the shape is fitted by selecting the point closest to the sensor as the corner for the L-shape and connecting it with lines for the borders of the object. As explained by Jimenez et al. [29] this approach to differentiate the sides of the L-shape is sensitive to errors in the measurements or cases where vehicles have rounded corners, since the algorithm then could select the wrong corner point. Alternative methods include employment of a search based approach where the definitive L-shape is found by iterating and optimising certain criteria [35], error minimisation [36], and a combination of error minimisation and iterative re-estimation of the L-shape until certain conditions are met [29].

Search-based Rectangle Fitting

One recently developed method for L-shape fitting was presented by Zhang et al. [35], where they formulate the fitting problem as an optimisation problem in order to find an optimal solution. The algorithm is called Search Based Rectangle Fitting (SBRF). For a cluster of points, a rectangle defined by four lines which encompass all the associated points is fitted, as presented in Figure 3.4. The rectangle is then rotated into a set of different directions θ between $0° \leq \theta < 90°$. The number of rotations are determined by a parameter δ, which specifies at what increment the rectangle rotates. For each new rotation the rectangle is refitted to contain all points. To determine the best fit, three criteria for an optimal solution are investigated, namely area minimisation, point-to-edges closeness maximisation and squared error point-to-edges minimisation. In the algorithm a score is calculated depending on the each of these criteria, and the rotation θ with the best score is chosen as the optimal solution. Their findings indicate that the closeness and squared error minimisation perform best in a vehicle pose estimation sense.

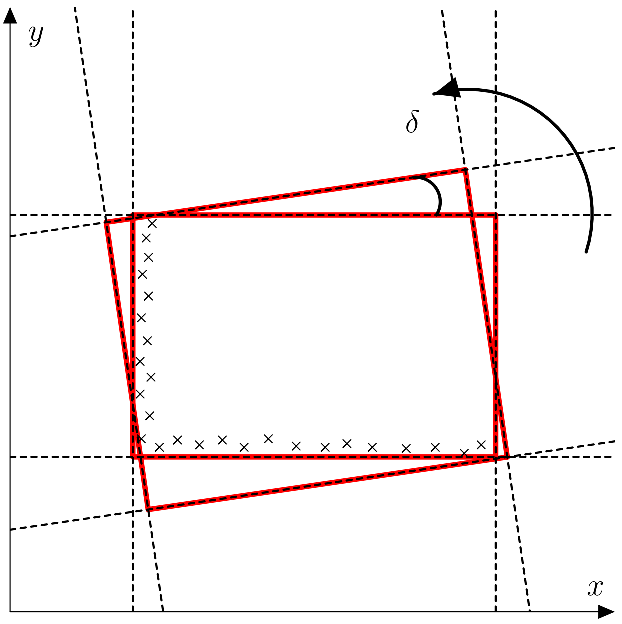


Figure 3.4: The fitted rectangle forms a bounding box for all the points in a cluster and the rotation of said rectangle around the points, from $\theta = 0°$ with an increment δ. The rectangle is re-calculated in each pose θ to always include all the points.

3.2.3 Convex Hull

For a set of points, a convex hull is the smallest convex region which contains all the points, meaning that for any two points on the hull a line segment between them will be part of the region. An example of a group of points and their corresponding convex hull can be seen in Figure 3.5. To obtain a convex hull, an incremental algorithm can be used, which was first suggested by Graham [37] and then further modified by Andrew [38]. The main part of the algorithm is to sort a set of points in ascending order depending on the x coordinate and iterate through the points while checking if the points should be added to the hull. Constructing the hull is done in two parts with one upper and one lower hull. In the upper hull, points are added if the last point and the next point create a clockwise turn, and in the lower hull points are added if the last point and the next point create a counter-clockwise turn. The two hulls are then merged together to create the full convex hull.

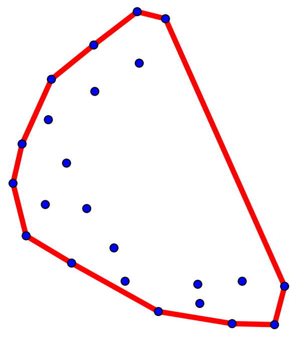


Figure 3.5: Example of a convex hull generated for an arbitrary set of data points.

To check whether two convex hulls overlap, a simple method used in graphics design called separating axis theorem is used, which is detailed by Boyd and Vandenberghe [4]. The theorem states that if there exists a line such that you can project the points of both hulls on it without the intervals of the points of both hulls overlapping, the convex hulls do not overlap, see Figure 3.6. Candidate lines to project points on can be found by taking all the lines that form the convex hull, and choosing a line perpendicular to that. If no such line can be found after testing with all the edges of both convex hulls, the hulls overlap. [4]

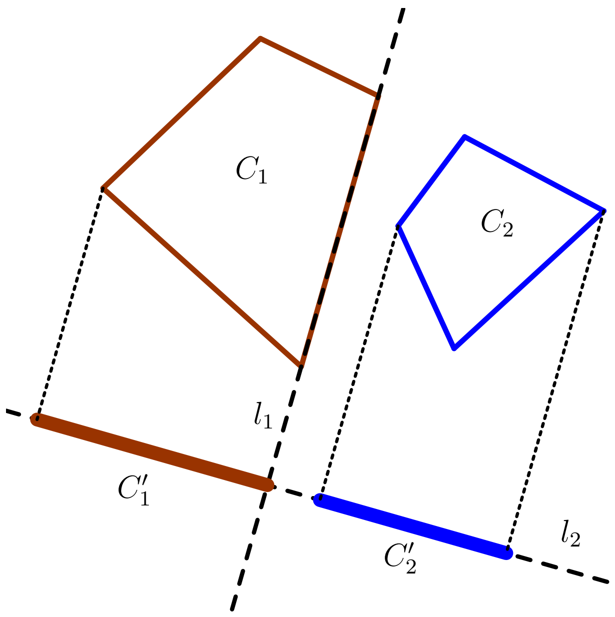


Figure 3.6: Convex hulls C_1 and C_2 are tested for overlap. A separating axis l_2 can be found such that the perpendicular projections C_1' and C_2' of the two convex hulls C_1 and C_2 do not overlap, so C_1 and C_2 do not overlap. Line l_2 is found by taking a perpendicular line to the line l_1 which corresponds to a side of one of the convex hulls. [4]

3.2.4 Concave Hull

As can be understood from the concept of convex hulls, they have a particular disadvantage when it comes to shape extraction as the produced polygons will be unable to give a satisfactory representation of more advanced shapes such as L-shapes or curved lines. A better representation of these shapes are thus concave hulls which will provide a tighter region around the set of points, as can be seen in Figure 3.7. Concave hulls are non-convex, meaning that a line segment between two points of the hull may result in a line that is outside of hull's region.

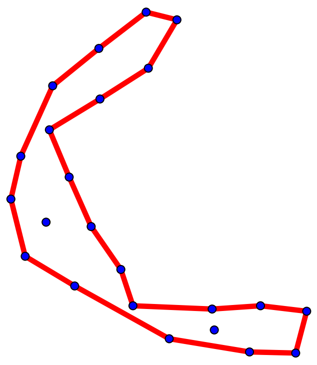
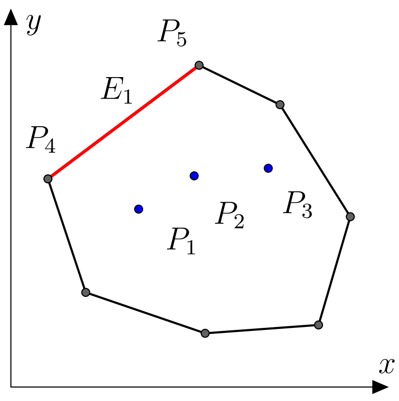


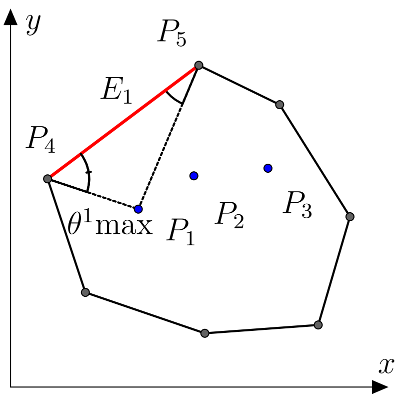
Figure 3.7: Example of a concave hull generated for the same data points as used in Figure 3.5.

In order to calculate the concave hull for a point set, an algorithm implementation by Rosen et al. [5] is available. Their algorithm utilises a convex hull as a starting point and then iterates through the set of edges.

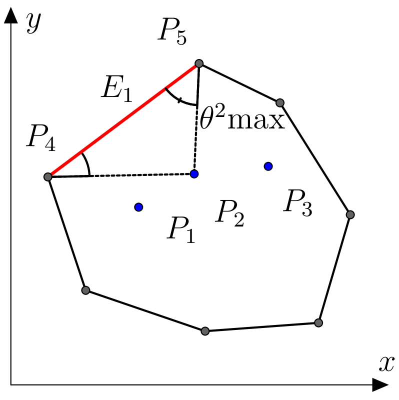
It begins with the longest edge as an example in Figure 3.8a illustrates. After this, it calculates the angles of the triangle formed by the two end-points of the edge to every other point in the cluster, see Figures 3.8b–3.8d. The stated goal is to find the point which has the smallest maximum angle. When it has been found, it is chosen as a new potential edge point as is shown in Figure 3.8e. A new edge point needs to satisfy certain conditions, for the current example these are: a maximum threshold on $\theta^2_{\max} < 90°$, and secondly, for the new edges E_{11} and E_{12} to not intersect any other edges. If the selected point is accepted the new edges, E_{11} and E_{12} will overwrite the previous edge, E_1. On the other hand if the selected point is rejected as a new edge point the previous edge will be kept and considered a final edge. This process is iterated until all edges have been tested for a split and a final selection of edges have been made.



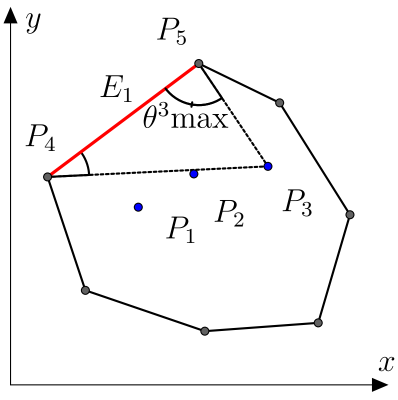
(a) Select the longest edge, E_1 of a convex hull.



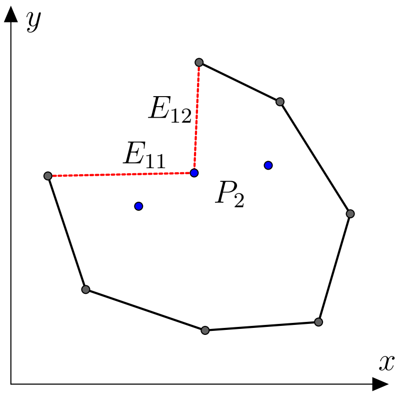
(b) For P_1, calculate angles $\angle P_1P_4P_5$ and $\angle P_1P_5P_4$ and find the maximum angle, θ^1_{max}.



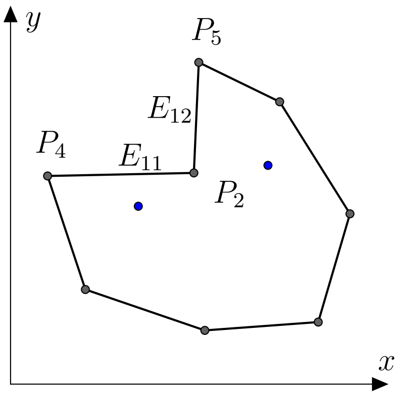
(c) For P_2, calculate angles $\angle P_2P_4P_5$ and $\angle P_2P_5P_4$ and find the maximum angle, θ^2_{max}.



(d) For P_3, calculate angles $\angle P_3P_4P_5$ and $\angle P_3P_5P_4$ and find the maximum angle, θ^3_{max}.



(e) Choose the point P_2, a potential new edge point, if it has the smallest maximum angle $\theta^2_{\text{max}} < \theta^1_{\text{max}} < \theta^3_{\text{max}}$, and create the new edges E_{11} and E_{12}.



(f) In this case $\theta^2_{\text{max}} < 90°$ and the edges, E_{11} and E_{12}, do not intersect any other edges and will thus replace the previous edge E_1.

Figure 3.8: Illustration of the concave hull algorithm in [5]. For a convex hull with interior points P_1, P_2 and P_3. The first iteration of the algorithm will select the longest edge E_1 and check if it can be split by making one of the interior points a new edge point. In the sequence above the result is the creation of two new edges E_{11} and E_{12}.

4 Methods

In this chapter, the development and implementation of an algorithm that aims to produce accurate geometric descriptors is described. The theory described in the previous chapter is used when applicable, where it has either been used as part of the algorithm or modified for this cause. The chapter begins in Section 4.1, by establishing how lidar data have been acquired and used. In Section 4.2, the design of the algorithm is described and lastly in Section 4.3, the methods used for evaluation are outlined.

4.1 Lidar Data

The lidar data has been recorded by vehicles travelling on a test track and on ordinary roads, traffic conditions mainly include highways but also some urban areas with dense traffic and pedestrian crossings.

Before the lidar data reaches the designed algorithm, preliminary processing is performed on the data. This preliminary preprocessing consists of removing obvious disturbances, and transforming the data to points in Cartesian coordinates. The obvious disturbances consist of detections very close to the sensor as illustrated in Figure 4.1. Only points with a range r to the sensor that satisfy

$$r > r_{\text{min,noise}}, \tag{4.1}$$

where $r_{\text{min,noise}}$ is the minimum range threshold, are considered for further processing. The preprocessing step is not part of this algorithm and is therefore not part of any evaluation.

4.2 Algorithm Implementation

This section aims to provide a comprehensive understanding of the algorithm implementation, including how previous research in the field has been adapted for this work as well as novel ideas that have been implemented.

4.2.1 Overview

The ideal functionality of the algorithm can be envisioned as consisting of two parts: clustering and shape extraction. The ideal clustering method would assign every

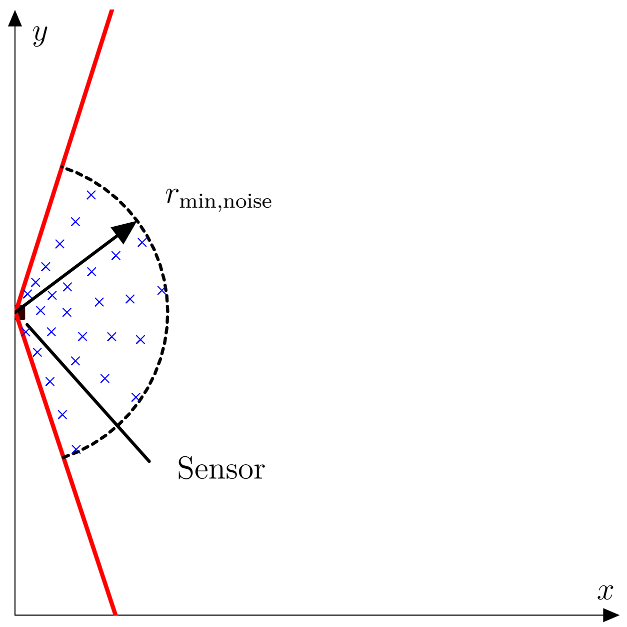


Figure 4.1: Detections within the range specified by $r_{\text{min,noise}}$, are removed in a prelimnary processing step. The red lines indicate the sensor's field of view and the blue crosses are noisy detections close to the sensor that will not be considered for further processing.

data point to a cluster such that each object in the sensor's field of view has exactly one cluster corresponding to it. Additionally, each cluster should only correspond to one object. Once these clusters are formed, they would all be handled separately in the shape extraction step, where each cluster of points would be replaced with a geometric 2D-shape in the xy-plane that approximates the points by describing the shape of the cluster. The method proposed in this chapter however splits these two ideal steps into four steps as seen in Algorithm 1, to accommodate for the fact that the data is collected from several vertical layers which need to be combined before a shape can be extracted in 2D.

The four main steps of the algorithm are:

- Clustering on line 1 to 4, presented in Section 4.2.2 uses a modified dual break-point algorithm, inspired by the algorithm that is described in Section3.1.1.2.
- Shape Extraction on line 5, presented in Section 4.2.3, is performed on the layer-separated clusters to obtain initial geometric knowledge about the clusters.
- Cluster merging on line 6, presented in Section 4.2.4 uses single linkage merging (Section 3.1.2) of the convex hulls (Section 3.2.3) of the clusters, but takes into account the geometrical information obtained in the previous step to avoid undersegmentation.

- Shape extraction on line 7, presented in Section 4.2.3, is performed to extract new shapes for all the clusters that were merged or changed in the previous step.

```
Algorithm 1: Clustering and Shape Extraction
Data: Set $P = \{p_1, p_2, \dots, p_{N_{pts}}\}$ of points $p_n = [x_n, y_n, z_n]$, set
      $L = \{l_1, l_2, \dots, l_{N_{pts}}\}$ of layer identifiers where $l_n$ is the identifier
      corresponding to $p_n$ for $l_n \in \{1, 2, \dots, N_{layers}\}$, $n \in \{1, 2, \dots, N_{pts}\}$ where
      $N_{pts}$ is the number of detections and $N_{layers}$ is the number of layers of the
      sensor
Result: Every point $p_n$ is assigned to a cluster, each for which a shape is extracted
1 for $j = 1 : N_{layers}$ do
2     Find all the points derived from layer j by finding a Set $P_j \subseteq P$ consisting of all
        points $p_k$ where $l_k = j$ and $l_k$ is an element in $L$ and $k \subseteq n$ ;
3     Perform clustering on all points $P_j$ of the current layer j, using modified dual
        breakpoint detection to assign a cluster ID $c_k \in \mathbb{N}^+$ to each point $p_k$;
4 end
5 Shape extraction for all clusters;
6 Cluster merging;
7 Shape extraction for merged clusters;
```

Each of these steps are further described later in this chapter. As the obtained shapes are ultimately intended for use in a Bayesian filter, it is important to be able to associate an uncertainty to each shape. However, a full uncertainty model for shapes have not been developed although an overlap factor have been suggested as explained in Section 4.2.5 which may be used in such a model.

4.2.2 Clustering

The ultimate goal of the clustering step is to assign every data point to a cluster such that all objects in the sensor's field of view are distinguished from each other, and are represented by exactly one cluster each. In this implementation, the clustering is performed layer-by-layer which results in an initial oversegmentation, which is later corrected in the merging step presented in Section 4.2.4.

As the clustering is done layer-by-layer without any intra-layer information exchange, the algorithm will be described as if only considering a single layer, but in the implementation it is applied to each layer separately.

The method is similar to DBD in Section 3.1.1.2 in that an ABD (Section 3.1.1.1) is extended with a method to cluster structures which are detected at a lower incidence angle based on the collinearity of these points.

The details of the implementation and the differences of this method from the DBD in Section 3.1.1.2, and the motivations for any additions or changes will be described after a short summary of the main differences:

1. Rather than separating clusters according to their classification as DC or AC, this method combines the criteria, so that a cluster can contain points included by both the distance-based or the angle-based criteria.
2. The previous DBD determines distance from each point to the successive point, and angle between a point and its two successive points. The method described here, however, compares each point to the nearest point in a three-dimensional Euclidean sense amongst the N consecutive points for the distance criterion, and additionally amongst the N preceding points for the angle criteria.
3. Rather than automatically including a point based on the angle criterion, this method increases the distance threshold depending on the estimated incidence angle of the measured surface.

Pseudocode of the full clustering algorithm can be seen in Algorithm 2.

Algorithm 2: Modified Dual Breakpoint Clustering

```
Data: Set $A = [p_1, p_2, ..., p_{N_{pts}}]$ of points $p_n = [x_n, y_n, z_n]$
Result: Every point $p$ is assigned to a cluster
1  $N_{horizon} \leftarrow$ number of points to look ahead/behind of current point;
2  for $n = 1 : N_{pts}$ do
3      Find $i$ which minimises $D_i^{dist} = ||p_{n+i} - p_n||$, $i \in [1, 2, ..., N_{horizon}]$ ;
4      if $D_i^{dist} < D_0^{dist}$ then
                   /* distance-threshold is satisfied */;
5          Cluster $p_{n+i}$ and $p_n$ together, update clusters if required;
6      else
7          Find $j$ which minimises $D_j^{angle} = ||p_{n-j} - p_n||$, $j \in [1, 2, ..., N_{horizon}]$;
8          Calculate $\delta$ using $p_{n-j}$, $p_n$ and $p_{n+i}$;
9          Estimate incidence angle $\phi$;
10         Calculate $D_0^{angle}$ using $D_{max}^{extra}$, $\phi$, $D_0^{dist}$;
11         Find the second longest distance $D_{max}^{v}$ between the points $p_{n-j}$, $p_n$ and
            $p_{n+i}$;
12         if $(\delta < \delta_0) \wedge (\phi < \phi_0) \wedge (D_{max}^{v} < D_0^{angle})$ then
                   /* angle criteria are satisfied */;
13             Cluster $p_{n-j}$, $p_n$ and $p_{n+i}$ together, update clusters if required;
14         else if $p_n$ has no cluster then
15             Start a new cluster containing only $p_n$
16         end
17     end
18 end
```

The parameter $N_{horizon}$ determines how many points ahead or behind the current point p_n the algorithm will look for candidate points for clustering, which is done on line 3 and 7. This is useful since the sequential ordering of the points in set A does not necessarily correspond to the proximity of the points, due to noise and occlusions. An example of an occlusion can be seen in Figure 4.2. The algorithm takes the point p_n in Figure 4.2a and compares it to the $N_{horizon}$ points succeeding p_n, i.e. p_{n+i} where $i \in [1, 2, ..., N_{horizon}]$, and selects the point p_{n+i} with the shortest Euclidean distance $||p_{n+i} - p_n||$ to the point p_n to consider for clustering. The distance $D_i^{dist} = ||p_{n+i} - p_n||$ is then tried against the distance threshold D_0^{dist} on line 4, which is defined as the distance in Equation 3.3, and if successful, point p_{n+i} will immediately be added to the cluster of p_n on line 5. Figure 4.2 illustrates how the forward horizon on line 3 works with the distance threshold for a sequence of points with an occluding object.

When two points p_n and p_{n+i} are being clustered on line 5, or three points p_n, p_{n+i} and p_{n-j} are being clustered on line 13, the following logic is applied with respect to cluster assignment and merging:

1. If none of the points are part of a cluster, create a new cluster and add all the two or three points to it.
2. If one point is already part of a cluster, add the other point or points to that cluster.
3. If more than one point is already part of a cluster, merge those clusters and add the considered points to the new cluster.

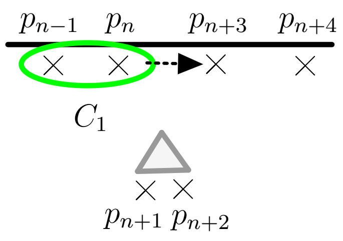
If the distance threshold on line 4 is violated, line 7 to 16 will evaluate whether the points can still be clustered using the angle criteria on line 12. A third point p_{n-j} is also considered, which is the closest point preceding p_n analogously with how p_{n+i} is the closest point succeeding p_n. Firstly, the three points p_{n-j}, p_n, and p_{n+i} are used to calculate three Sets v_1, v_2 and u which connect the points in any direction, see Figure. 4.3. v_1 and v_2 are chosen to be the two shorter vectors, while the longest one is designated u. The smallest angular difference δ between the two vectors v_1 and v_2 is calculated on line 8 using a re-ordering of the geometric definition of the dot product as

$$\delta' = \cos^{-1}\left(\frac{v_1 \bullet v_2}{||v_1|| ||v_2||}\right) \tag{4.2a}$$

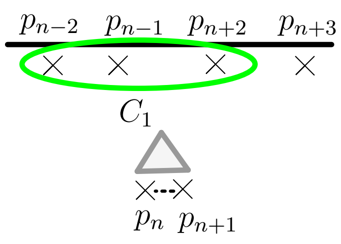
$$\delta = \min(\{\delta', \pi - \delta'\}) \tag{4.2b}$$

where Equation 4.2b is used to select the smallest angle between the vectors. Another vector w is is defined as the difference between the sensor coordinate, and the mean coordinate of the three points p_{n-j}, p_n, and p_{n+i}.

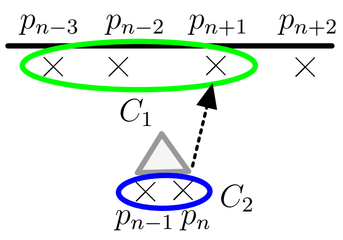
Secondly, the incidence angle ϕ is approximated on line 9 by calculating the angle between the vectors w and u using the method described in Equations 4.2a and 4.2b, see Figure 4.3.



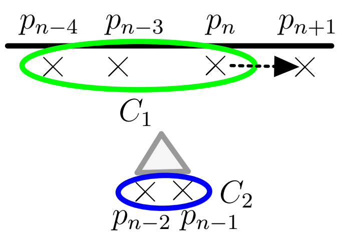
(a) In Algorithm 2, the distance between point p_n and p_{n+3} is being considered because of line 3, despite p_{n+1} being the one directly succeeding it in the data order. In this case p_{n+3} will be added to cluster C_1, on line 5.



(b) A point p_n from an occluding object is being considered. The closest successive point is p_{n+1}, which will form a new cluster C_2 together with p_n, since none of the points are previously clustered, see line 14-15 of Algorithm 2.



(c) At the next iteration, the second point of the occluding object is being considered. The closest successive point is p_{n+1}, which in this case is too far to cluster. No clusters will change since p_n is already part of a cluster.



(d) Point p_n is being considered. The closest successive point is p_{n+1} which will be clustered and included in C_1.

Figure 4.2: Example of modified dual breakpoint clustering for a sequence of points interrupted by an occlusion.

Thirdly, the length D^v_{max} is defined as the shortest vector among v_1 and v_2 which is used on line 11.

The criteria on line 12 that need to be satisfied for clustering of the points p_n, p_{n+i}

and p_{n-j} can then be formulated as

$$\begin{aligned} \delta &< \delta_0 \\ \phi &< \phi_0 \\ D^v_{max} &< D_0^{angle} \end{aligned} \tag{4.3}$$

where δ_0 and ϕ_0 are predefined thresholds, and D_0^{angle} is a dynamic threshold defined as

$$D_0^{angle} = D_0^{dist} + D_{max}^{extra} \cdot f(\phi) \tag{4.4}$$

where D_0^{dist} is the distance threshold, D_{max}^{extra} a predefined maximum extra distance and $f(\phi)$ a scaling function which takes values between 0 and 1. $f(\phi)$ is defined as

$$f(\phi) = \begin{cases} 1, & \phi \leq 0, \\ \frac{-\phi^2+\alpha_0^2}{\alpha_0^2}, & 0 < \phi \leq \alpha_0 \\ 0, & \phi > \alpha_0 \end{cases} \tag{4.5}$$

which is designed to be 1 when ϕ is close to zero and is strictly decreasing in the interval $0 < \phi < \alpha_0$, and 0 when $\phi \geq \alpha_0$. $f(\phi)$ is set to 0 for any negative value of $f(\phi)$. An example of points being clustered using these angle criteria can be seen in Figure 4.4, where the first criteria in Equation 4.3 is illustrated.

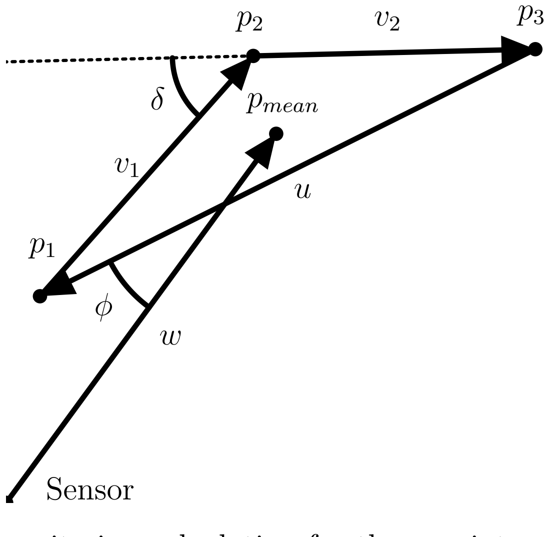
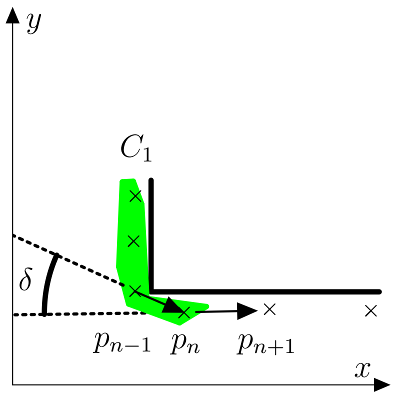
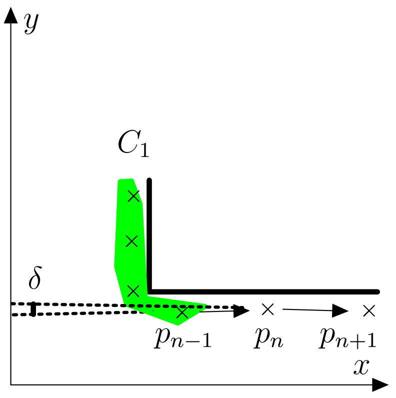


Figure 4.3: Angle criterion calculation for three points p_1, p_2 and p_3. Vectors v_1, v_2 and u, whose directions are unimportant, connect the three points. The longest vector is u, while v_1 and v_2 are the two shorter vectors. δ is the angle between the shorter vectors v_1 and v_2, and must be smaller than δ_0 on line 12 in Algorithm 2 for the criteria to be satisfied. p_{mean} is the mean of the three points p_1, p_2 and p_3, and w is the vector from the sensor to p_{mean}. The incidence angle ϕ is approximated by calculating the angle between the two vectors u and w, and must be smaller than ϕ_0 on line 12 for the criteria to be satisfied.



(a) The distance between p_n and p_{n+1} is large so the distance criterion on line 4 is violated. The angle criteria are instead evaluated, but here the angle δ is too large, and no new point is added to the cluster C_1.



(b) The distance between p_n and p_{n+1} is large so the distance criterion on line 4 is violated. The angle criteria are instead evaluated, and here the angle δ is small enough. As long as the other criteria in Equation 4.3 are also satisfied, p_n and p_{n+1} will be added to the cluster C_1.

Figure 4.4: Example of the first criterion in Equation 4.3 for modified dual breakpoint clustering for a sequence of points detected around a corner.

4.2.3 Shape Extraction

When the detections have been clustered layer by layer as in the section above, shape extraction is performed on all the clusters. In this step, the previous 3D coordinates are transformed into 2D by simply ignoring the z-component for the purpose of fitting shapes in the $xy-$plane, as mentioned in Section 1.4. The shapes are to be used to provide additional information to the cluster merging subfunction, which is described in Section 4.2.4. The first part of this section pertains to shape fitting and how shapes are represented and the second part deal with how clusters are classified as the different shapes.

4.2.3.1 Shape Fitting and Representation

There are four shapes which are considered in shape extraction, namely points, polygons, lines and L-shapes. Typical examples of data that would be assigned these shapes are presented in Figure 4.5.
Each of these shapes are assigned a shape identification number from one to four, which in addition to identity also corresponds to the shape's priority. Priority increases with the shape identification number. Assignment for each shape is summarised in Table 4.1. The priority information is used in cluster merging, as described in Section 4.2.4, where merging of clusters is considered based on shape information.
Below the extraction and representation of each shape is detailed.

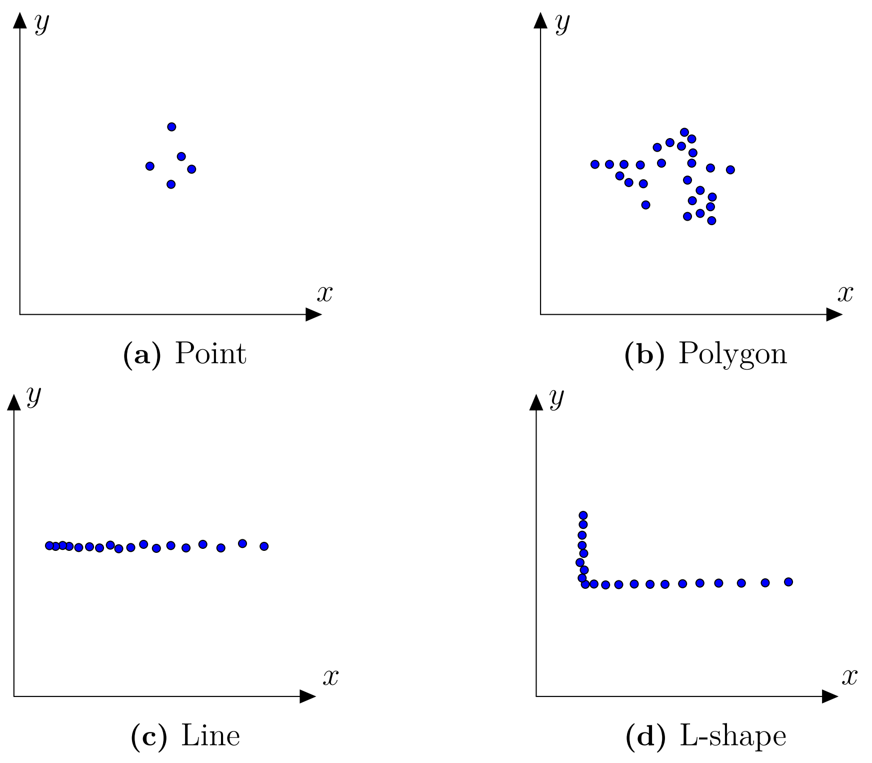


Figure 4.5: The figures above illustrate possible cluster formations and their ideally assigned shape in the respective caption. The different shapes considered for shape extraction are points (a), polygons (b), lines (c) and L-shapes (d).

Table 4.1: Shape identification number for the different shapes. The number also corresponds to the priority of the shapes, where a higher number indicates higher priority.

Shape ID number	Shape
1	point
2	polygon
3	line
4	L-shape

Point Points are represented by the mean position of the cluster. Each point shape will thus be represented by a vector

$$\text{Point} = p_{\text{mean}} = \begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix}. \tag{4.6}$$

Polygon Depending on the number of points in the cluster, a convex hull or a concave hull is calculated to describe a polygon. If the number of points n_{tot} satisfy

$$n_{\text{tot}} < n_{\text{max,concave}} \tag{4.7}$$

a concave hull will be computed and otherwise a convex hull. The reason for this is that the computational time to generate the concave hull increases significantly with the number of points in the cluster and is thus unfeasible for clusters with more than $n_{\text{max,concave}}$ points.

The calculations to generate the respective hulls follow the algorithms referred to in Sections 3.2.3 and 3.2.4. The polygon is saved in a matrix consisting of x and y coordinates, where each column represents an edge point of the hull. Considering a hull with n edge points, the matrix for the polygon will be,

$$\text{Polygon} = \begin{bmatrix} p_1 & p_2 & \dots & p_n & p_1 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & \dots & x_n & x_1 \\ y_1 & y_2 & \dots & y_n & y_1 \end{bmatrix}, \tag{4.8}$$

note that the first and last point are the same. This fact makes it easier to use the data from an implementation standpoint.

Line For line extraction, an implementation of the Theil-Sen estimator is used as described in Section 3.2.1. An additional step is added to determine the endpoints of the line, as the original estimator only gives the slope and y-intercept of the line. The method that is implemented utilises the obtained angle of the line to perform a series of coordinate transformations on the data in order to acquire the endpoints. The process is decribed below:

1. From the estimated slope k, the angle ϕ is obtained.
2. Points are made zero mean with respect to x and y, centering the cluster at the origin.
3. The points are clockwise rotated to the x-axis using ϕ. The extreme x-values, $x_{o,\text{min}}$ and $x_{o,\text{max}}$ yields the start and stop x-coordinates for the line at this orientation.
4. With y-coordinates as $y_{o,\text{min}} = y_{0,\text{max}} = 0$, a horizontal line is defined at the origin, which is then rotated counter-clockwise with ϕ and translated back using the mean of all points in order to obtain the final line.

The line is then given as the two points in the following format

$$\text{Line} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} x_{\text{min}} & x_{\text{max}} \\ y_{\text{min}} & y_{\text{max}} \end{bmatrix}. \tag{4.9}$$

Furthermore, line extraction takes into account the presence of outliers by looking at the square distances of all the points to the the fitted line, and determining a lower and upper threshold for outliers. This is done by considering the interquartile range (IQR) and the first, Q_1 and third quartile Q_3 of the square distances, according to

$$\begin{aligned} \text{Lower threshold} &= Q_1 - c \cdot \text{IQR} \\ \text{Upper threshold} &= Q_3 + c \cdot \text{IQR}, \end{aligned} \tag{4.10}$$

where c is a tuneable factor. By re-estimating the line without the outliers, a more representative shape is generally acquired, with a few exceptions. An example of when it runs into problems is when a cluster is shaped as a long, near horizontal line as shown in Figure 4.6. In this case, the right most part of the cluster would be excluded after outlier rejection due to a small curvature at the far end of the cluster. The re-estimated line would be using a smaller subset of the cluster's points, providing a significantly shorter line. A way to mitigate this weakness is to only allow the re-estimated line to replace the initial one if the length of the line is close to unchanged. To be considered unchanged, the length may not change more than a set proportion p_{line} or exceed a maximum shrinkage threshold δ_{min}.

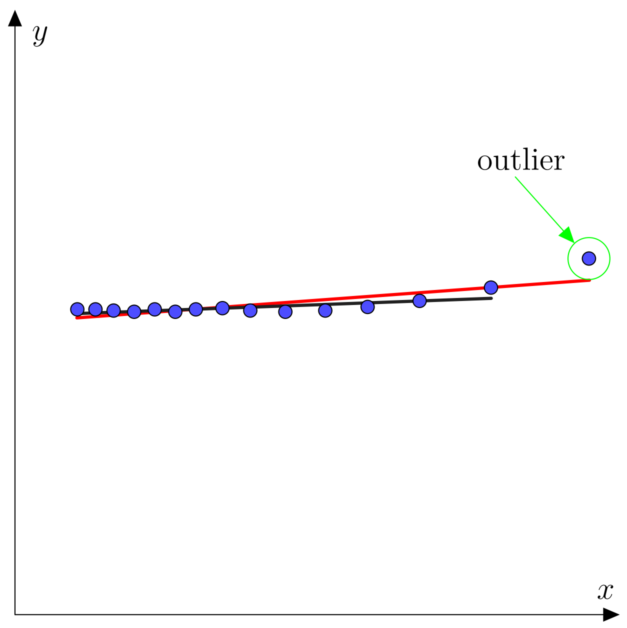


Figure 4.6: An example encountered from detections on a guardrail. The initial line fitting of all the points yields the red line. The point indicated in green is an outlier with respect to red line and is excluded in re-estimation of a line without outliers. However, the new line marked in black, is significantly shorter than the actual object as indicated by the points so the red line would in this case be retained.

When the cluster appears as a vertical or horizontal line, which is described below in Section 4.2.3.2, line extraction is performed differently from the above description. Instead, the maximum and minimum values of the y or x coordinates of all the points in the cluster are used to determine the span of the line. And the mean of the all the x or y values, are used to place the line. This concept is summarised in Table 4.2 for the horizontal and vertical case respectively.

Table 4.2: Line coordinates for a horizontal or a vertical line

	x_{min}	y_{min}	x_{max}	y_{max}
Horizontal	$\min(\mathbf{x})$	$\bar{\mathbf{y}}$	$\max(\mathbf{x})$	$\bar{\mathbf{y}}$
Vertical	$\bar{\mathbf{x}}$	$\min(\mathbf{y})$	$\bar{\mathbf{x}}$	$\max(\mathbf{y})$

L-shape The output of the SBRF algorithm described in Section 3.2.2 provides four unbounded lines, the intersection points of these lines constitute the endpoints of the rectangle that is of interest, see Figure 4.7a, and are calculated by solving a linear system of equations. Under the assumption that L-shapes from vehicles must be convex the L-shape is generated by the three points closest to the sensor. That is, the distance from the sensor to each point is calculated and the point with the greatest distance to the sensor is removed as seen in Figure 4.7b. The remaining points, shown in 4.7c, constitute the L-shape and are ordered consecutively.

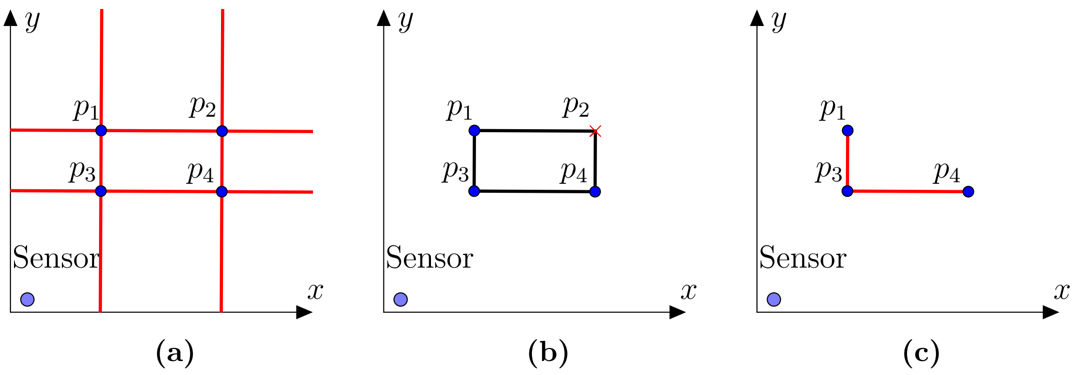


Figure 4.7: The process of generating an L-shape from the four lines obtained by SBRF. In (a) the intersection points are found. Next, in (b) the point furthest from the sensor, p_2 is identified and finally in (c) it is removed creating an L-shape consisting of the three points p_1, p_3 and p_4.

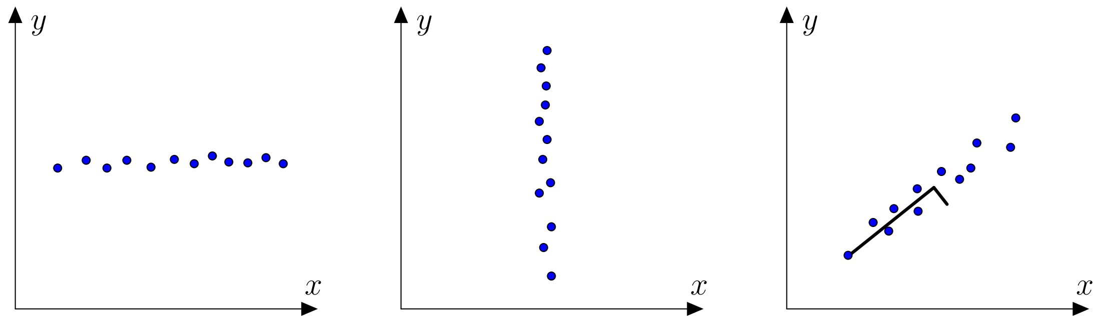
For the example in Figure 4.7, the L-shape will be a matrix consisting of the remaining points, p_1, p_3 and p_4, according to

$$\text{L-shape} = \begin{bmatrix} p_1 & p_3 & p_4 \end{bmatrix} = \begin{bmatrix} x_1 & x_3 & x_4 \\ y_1 & y_3 & y_4 \end{bmatrix}. \tag{4.11}$$

4.2.3.2 Classification of Shapes

The assignment of a shape to each cluster is done in two steps, first points and lines are classified. This is done by using the principal components of the cluster, namely the eigenvalues and eigenvectors of the covariance matrix of the cluster. Secondly, clusters classified as neither points nor lines are tested for possibly being an L-shape or a line.

Clusters where both eigenvalues $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2\}$ of the covariance matrix are smaller than a threshold, σ_{point}, or only contain one point are immediately classified as points. Next, if the cluster is not a point type, the variance of a cluster is used to establish whether the cluster can be well represented by a vertical or horizontal line. This is true for clusters with a small variance σ_x^2 or σ_y^2 as seen in Figure 4.8a and 4.8b, provided a sufficiently small covariance σ_{xy}^2. These conditions are managed as thresholds, $\sigma_{0,\text{var}}$ and $\sigma_{0,\text{cov}}$ on the variance and covariance respectively. Lastly the size of the first and second principal component of the cluster, defined by the eigenvalues of the covariance matrix are used to determine whether the cluster should should be tagged as a line despite being non-vertical and non-horizontal, as illustrated in Figure 4.8c. For this to be true, the primary principal component need to satisfy $\lambda_1 > \sigma_{\text{point}}$ and the secondary principal component, $\lambda_2 < \sigma_{\text{point}}/3$.



(a) A horizontal line is fitted when there is a small variance $\sigma_y^2 < \sigma_{\text{point}}$, and $\sigma_x^2 > \sigma_{\text{point}}$

(b) A vertical line is fitted when there is a small variance $\sigma_x^2 < \sigma_{\text{point}}$, and $\sigma_y^2 > \sigma_{\text{point}}$

(c) A non-horizontal, non-vertical line is fitted when the secondary principal component λ_2 is significantly smaller than the primary principal component λ_1 such that $\lambda_1 > \sigma_{\text{point}}$ and $\lambda_2 < \sigma_{\text{point}}/3$

Figure 4.8: Three examples of line clusters that could be classified using the variance σ_x^2, σ_y^2 or the principal components λ_1 and λ_2.

The clusters that are not classified as points or lines are analysed in a second step, designed specifically to find L-shapes and polygons. The initial action for this check is to fit a bounding box to the cluster using SBRF as described in Section 3.2.2. Some data processing is performed on the bounding boxes to obtain an L-shape representing the sides visible to the sensor, as was described above in Section 4.2.3.1. This L-shape is then analysed and compared to other shapes in several stages to verify or reject it as the best shape for the specific cluster.

Firstly, the fitted rectangle's width w, constituting the shortest side, is compared to a threshold w_{max}. If

$$w < w_{\text{max}}, \tag{4.12}$$

a line will be fitted to the cluster to be used in subsequent comparisons. The cluster can in this case be classified as either an L-shape, a line or a polygon. If $w > w_{\text{max}}$, the line will not be fitted and the cluster can subsequently only be classified as an L-shape or a polygon.

To choose an L-shape several criteria needs to be met. The first stage of this procedure is creating a diagonal vector between the two endpoints of the constructed L-shape and calculating the proportion, p of points that are distributed on the side of the diagonal towards the sensor as described in Figure 4.9. This proportion need to satisfy,

$$p > p_{\text{min}}, \tag{4.13}$$

where $p_{\min}$ is the minimum proportion allowed for an L-shape to be feasible.

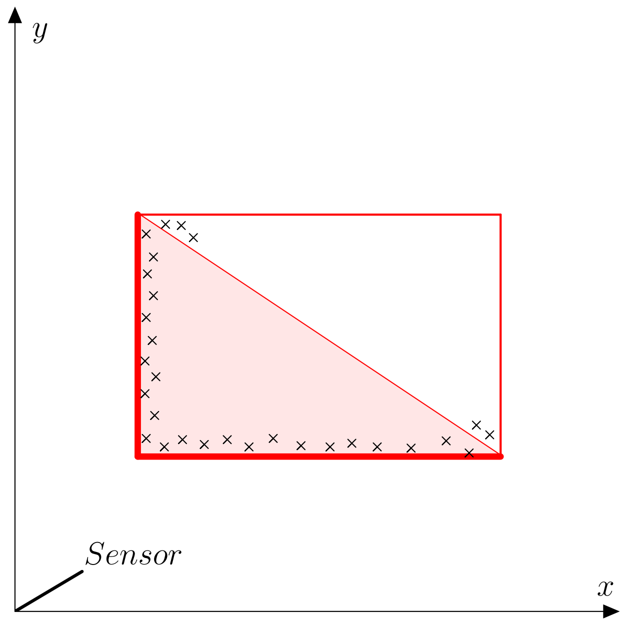


Figure 4.9: A cluster bounded by a rectangle, where the assumed L-shape is the two sides closest to the sensor. The proportion $p = n_L/n_{\text{tot}}$, where n_{tot}, is the total number of points contained in the rectangle and n_L, the number of points on the shaded side.

Additionally the mean square error MSE_L of all the points to the closest lines in the L-shape are calculated which need to meet a maximum requirement $\text{MSE}_{L,\max}$ according to

$$\text{MSE}_L < \text{MSE}_{L,\max}. \tag{4.14}$$

A further consideration is the area of the rectangle A_L that the L-shape spans, which needs to satisfy a minimum threshold $A_{\min}$ as

$$A_\text{L} > A_{\min} \tag{4.15}$$

to be feasible. Finally, the L-shape is chosen for the cluster if equations 4.13 - 4.15 are satisfied and additionally if Equation 4.12 is satisfied, the mean square error for the L-shape also needs to be sufficiently smaller than the mean square error of the line

$$\text{MSE}_L(1 + \tau_{\text{line}}) < \text{MSE}_{line}, \tag{4.16}$$

where τ_{line} is a tunable factor.

If the L-shape is not chosen, the computed line shape will instead be used as a preliminary shape unless two criteria for polygons are satisfied. Firstly, the fitting error of the preliminary line MSE_{line} has to be larger than a threshold $\text{MSE}_{line,\max}$ as

$$\text{MSE}_{line} > \text{MSE}_{line,\max} \tag{4.17}$$

and secondly, the rectangle area A_{L} has to be larger than the minimum area threshold for polygons $A_{p,\min}$ as

$$A_{\mathrm{L}} > A_{p,\min}. \tag{4.18}$$

If both of the criteria in Equations 4.17 or 4.18 are satisfied, a polygon will be fitted to the cluster.

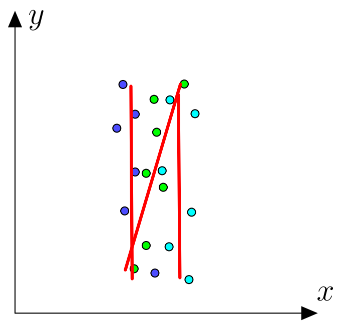
4.2.4 Cluster Merging

As the modified DBD algorithm presented in Section 4.2.2 works on a layer-by-layer basis, the initial clustering and consequently the pre-classification of shapes will result in significant oversegmentation. This is illustrated in Figures 4.10a and 4.10c, where in both cases multiple shapes describe one single object. This is a typical result when an object is detected by several layers simultaneously, and is why merging of clusters and their associated shapes is necessary.

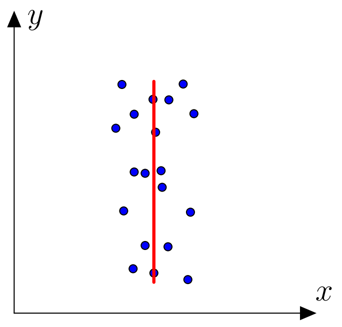
As mentioned in Section 3.1.2, there are several hierarchical clustering methods of associating elements, or in this case clusters, based on proximity. Taking the nature of the lidar data into account a suitable method is single linkage clustering, which uses the closest distance between clusters as similarity measure. By applying single linkage clustering on the multiple clusters, they can be combined into more desirable clusters and new shapes can be fitted as seen in Figures 4.10a and 4.10d.

The full pseudocode for cluster merging is presented in Algorithm 3. In order to perform single linkage merging each cluster is represented by a convex hull, see line 2, and the distances between clusters are collected in a cost matrix D according to line 3 to 17. To improve efficiency, only pairs of clusters that have an overlap in extreme x or y coordinates are considered, since this value can quickly be calculated. The distance is then calculated on line 12 using the smallest distance between the convex hulls. To check for overlap the separating axis theorem is used as described in Section 3.2.3, and if the convex hulls overlap, the distance is set to zero. After the distances between all clusters have been found, the merging is done in a loop described on line 19 to 33, which runs until the minimum distance between clusters as found in D, exceeds a certain cutoff threshold. Furthermore, relying exclusively on single linkage for merging has some disadvantages, one can be seen in Figure 4.11, the advent of ground points in the data would cause the object represented by an L-shape to merge with the clusters containing mainly ground points. As such the shape type for each clusters is used as a way of discriminating against mergers of specific shapes in line 23, which is handled by a separate function presented in Algorithm 4.

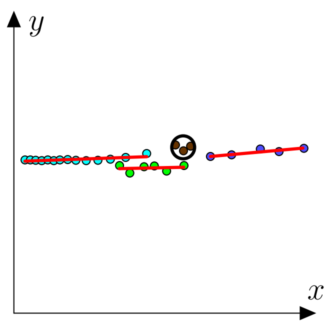
Algorithm 4: AllowMergeSingleLinkage is used as a criterion function to allow or disallow a potential cluster merging. It uses the shape information of the clusters and some heuristic to reject certain merging types more likely to result in undersegmentation or distortion of shape. The input clusters $K = \{k_1, k_2\}$ can consist of one or more shapes, if they have been previously merged with other clusters. On line 1 to line 3, the highest priority shapes of the clusters in K are therefore found and



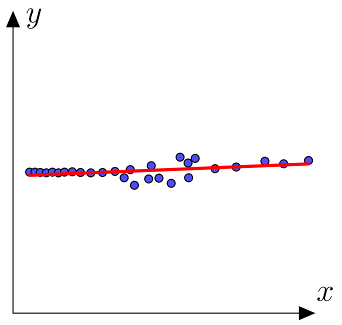
(a) Oversegmentation due to overlapping clusters from different layers on the rear end of a car.



(b) The resulting cluster and shape after merging clusters in (a).



(c) Heavy oversegmentation of detections on a guardrail.



(d) The resulting cluster and shape after merging of the clusters in (c).

Figure 4.10: The above figures illustrates two examples of DBD clustering and shape extraction, (a) and (c) which motivate merging with single linkage in order to generate one cluster and one shape for each object, (b) and (d).

ordered so that I_{max} corresponds to the highest cluster ID of both clusters in K, while I_{min} corresponds to the highest cluster ID in the other cluster. Basically only the highest shape ID in each cluster $\{k_1, k_2\}$ are being used in for the conditions being evaluated from line 4 and onward.

On line 4 to line 6, the algorithm returns true if both clusters in K only contain points or polygon shapes, for example the situation in Figure 4.12a. This is due to the fact that most noise will appear as points or polygons, while most vehicles and objects will appear as lines or L-shapes. For this reason the grouping of points and polygons rarely risk distorting any good shape representing a vehicle or other dynamic object. Conversely, the merging of a polygon and a line or L-shape will quite often distort the line or L-shape and cause the merged cluster to become a polygon as can be seen in Figure 4.12b. To avoid this, the algorithm will return false on line 7 to line 9 when this situation occurs. The final merging case that is handled by Algorithm 4 is between lines and L-shapes, that is when one cluster contains at least one L-shape, and when the other cluster contains at least one line but no L-shapes, as in Figure 4.12c. Line 10 to line 22 describe how this case is handled, with line 15 calculating the distance between the end points of the line to any point on the L-shape, as illustrated in Figure 4.12c and 4.12d.

Algorithm 3: Cluster Merging

```
Data: Set of N clusters of points $S = \{s_1, \ldots, s_N\}$, $N$ shape ids $I = \{i_1, \ldots, i_N\}$,
      shape data $F$
Result: Merged clusters
1  $P \leftarrow$ set of all cluster indices to keep track of performed mergers;
2  $C \leftarrow$ convex hulls for each cluster;
3  $D \leftarrow$ initiate a $N$ by $N$ cost matrix of infinite distances;
4  for $n$=1:N-1 do /* compute the cost matrix */
5      $E_{1,x} \leftarrow$ maximum and minimum $x$ values in $C(n)$;
6      $E_{1,y} \leftarrow$ maximum and minimum $y$ values in $C(n)$;
7      for $m$=n+1:N do
8          $E_{2,x} \leftarrow$ maximum and minimum $x$ values in $C(m)$;
9          $E_{2,y} \leftarrow$ maximum and minimum $y$ values in $C(m)$;
10         if isOverlapping($E_{1,x}$,$E_{2,x}$) then /* Extreme x values overlap */
11             if isOverlapping($E_{1,y}$,$E_{2,y}$) then /* Extreme y values overlap */
12                 $D(n, m) \leftarrow$ DistanceBetweenConvexHulls($C(n)$,$C(m)$);
13                 $D(m, n) \leftarrow D(n, m)$;
14             end
15         end
16     end
17 end
18 $d_{\min,last} \leftarrow$ initiate as 0;
19 for $i = 1 : (N^2)/2$ do /* Merging */
20     if $d_{min,last} <$ cutoff threshold then
21         $d_{\min,last} \leftarrow$ minimum distance in $D$;
22         Find indices $K$ which identifies the pairwise closest clusters;
23         while $\neg$ AllowMergeSingleLinkage(K,I,P,F) $\wedge$ ($d_{min,last} <$ cutoff threshold)
            do
24             Set $D$ for indices $K$ as infinite;
25             Update $d_{\min,last}$ and $K$;
26         end
27         if $d_{min,last} <$ cutoff threshold then
28             Update clusters $S$, matrix $D$ and list of performed mergers $P$;
29         end
30     else
31         break;
32     end
33 end
```

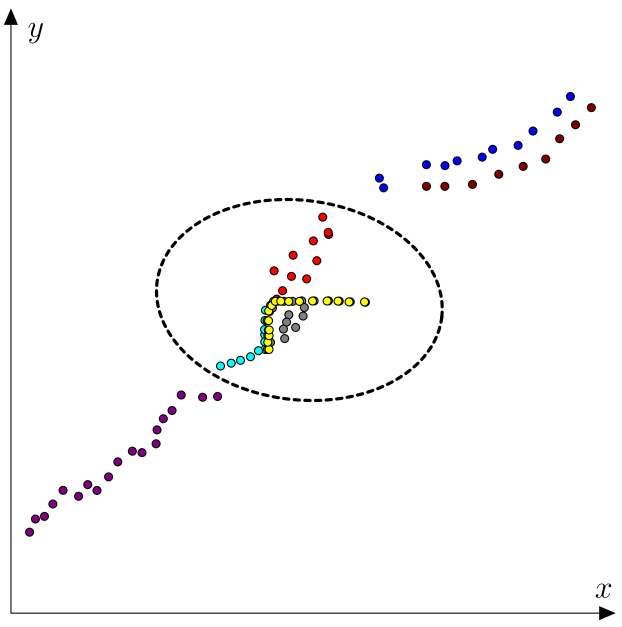
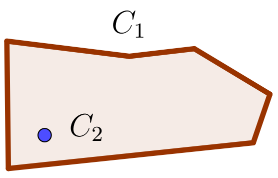


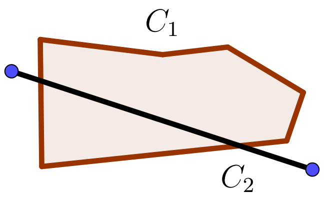
Figure 4.11: The L-shaped data, mainly constituting of the yellow cluster is surrounded by clusters which include predominantly ground points. The dashed ellipse indicate a possible grouping that a single linkage clustering algorithm might give on the scenario depicted.

Algorithm 4: AllowMergeSingleLinkage

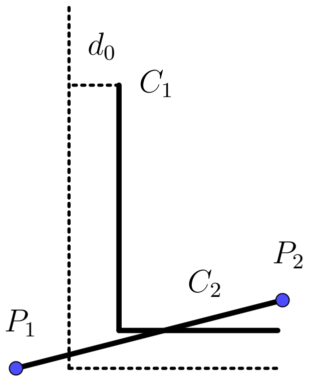
```
Data: Proposed merge of clusters with indices $K = \{k_1, k_2\}$, where each cluster
      consists of a list of Shapes $S_K = \{s_1, s_2, ..., s_n\}$ and each shape $s_i$ has an
      associated Shape ID $I_{s_i}$ and a list of points $P_{s_i} = \{p_1, p_2, ..., p_m\}$ which
      describe the geometry of the shape.
Result: Boolean flag
Find the highest shape IDs $I'_{1,\max}$ and $I'_{2,\max}$ among the shapes that are part of
  clusters $k_1$ and $k_2$ respectively;
$I_{max} \leftarrow \max\{I'_{1,\max}, I'_{2,\max}\}$;
$I_{min} \leftarrow \min\{I'_{1,\max}, I'_{2,\max}\}$;
                /* Allow merging of points and polygons */;
if $(I_{max} \leq 2) \wedge (I_{min} \leq 2)$ then
    return true;
end
        /* Disallow merging of polygons with lines or L-shapes */;
if $(I_{max} \geq 2) \wedge (I_{min} = 2)$ then
    return false;
end
        /* Allow circumstantial merging of lines with L-shapes */;
if $I_{max} = 4 \wedge I_{min} = 3$ then
    Call the cluster which includes L-shapes $k_i$, and the other cluster $k_j$;
    for each L-shape in cluster $k_i$ do
        for each line in cluster $k_j$ do
            for each endpoint of the line which is outside of the L-shape do
                Calculate distance $d$ as distance from the endpoints of the line to
                  any point on the L-shape;
                if $d > d_0$ then
                    return false;
                end
            end
        end
    end
end
return true;
```



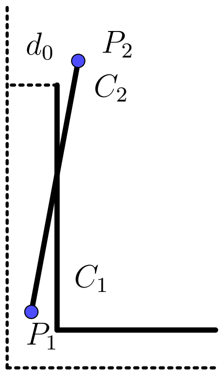
(a) A polygon cluster C_1 will always be allowed to merge with a point cluster C_2.



(b) A polygon cluster C_1 will never be allowed to merge with a line cluster C_2.



(c) Line and L-shape will not be allowed to merge since an endpoint P_1 of the line is outside the L-shape, and outside of the distance threshold d_0.



(d) Line and L-shape will be allowed to merge since the endpoints of the line P_2 is inside the L-shape, and while P_1 is outside the L-shape it is still within the distance threshold d_0 of the edges of the L-shape.

Figure 4.12: Different examples of shape combinations that are assessed by Algorithm 4.

4.2.5 Overlap factor

An overlap factor is proposed which can be used as a supporting metric for dynamic tuning of covariances in an uncertainty estimation model. The overlap factor, Γ is a confidence measure that aims to provide information of how well a final, merged cluster is represented by the individual clusters from multiple layers which forms it. Clusters that have not merged will have an overlap factor $\Gamma = 1$, while a cluster consisting of n merged clusters of identical shape from n different layers would have an overlap value of $\Gamma = n$. For all clusters the overlap is calculated by considering the merged cluster's total angular span α_{tot} and the included clusters' separate angular ranges α_j for $j = 1 \dots n$, where n is the number of clusters that have been merged. The angular spans are calculated by observing which azimuth the points of each cluster were observed in. The overlap is then defined as

$$\Gamma = \frac{\sum_{j=1}^{n} \alpha_j}{\alpha_{\text{tot}}}, \tag{4.19}$$

where each included cluster j has a minimum and maximum azimuth angle $\theta_{j,\text{min}}$ and $\theta_{j,\text{max}}$, an example case is presented in Figure 4.13, which have been used to calculate the azimuth angle range α_j and α_{tot} according to

$$\alpha_{\text{tot}} = \max_j(\theta_{j,\text{max}}) - \min_j(\theta_{j,\text{min}}) \tag{4.20}$$

$$\alpha_j = \theta_{j,\text{max}} - \theta_{j,\text{min}}. \tag{4.21}$$

The resulting overlap will be less than one if there are gaps between the merged clusters. For clusters consisting of multiple singular clusters of only one data point each, the overlap will be zero as $\theta_{j,\text{max}} = \theta_{j,\text{min}}$ in this case.

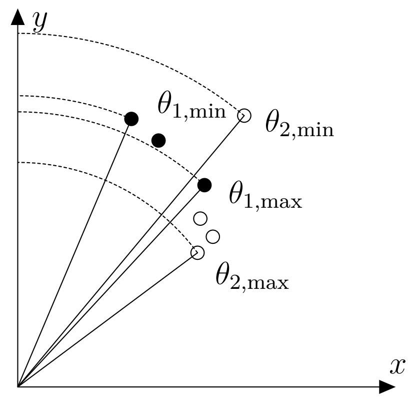


Figure 4.13: The angles,$\theta_{1,\text{max}}$, $\theta_{1,\text{min}}$, $\theta_{2,\text{max}}$ and $\theta_{2,\text{min}}$ for an example with two merged clusters, represented by the black and white dots respectively.

4.3 Evaluation

To evaluate the algorithm, qualitative and quantitative methods were used which are described in detail below. In order to assess the algorithm's comparative performance, a basic reference algorithm has been made available. The reference algorithm was compared with the proposed algorithm in all evaluation cases. The main subfunctions that the reference algorithm consists of are:

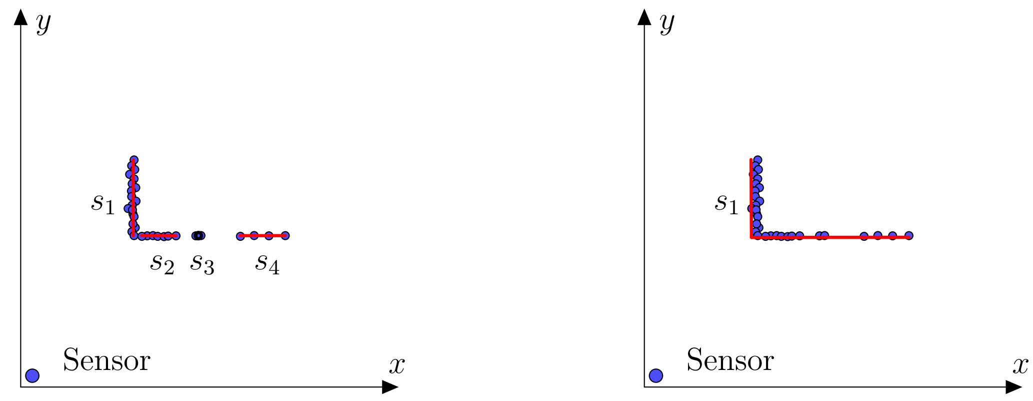
- Ground point removal (GPR) and removal of points close to the sensor due to noise.
- Clustering of data from all layers simultaneously using an adaptive breakpoint detector approach.
- Merging of clusters by proximity using single linkage clustering
- Shape extraction of points, lines and L-shapes.

4.3.1 Manual Evaluation

In order to assess the algorithm's performance when driving in ordinary traffic, a limited set of frames over different driving sequences have been manually stepped through and evaluated. The priority was to verify that dynamic objects was correctly segmented and subsequently correctly represented by the shapes extracted by the algorithm. To be able to manually assess these results, certain criteria needed to be established to facilitate consistency in deciding whether an object had been correctly or incorrectly represented.

An object has been considered correctly represented if it was only represented by one shape, and was neither oversegmented nor undersegmented. Shapes that poorly fit the data were labelled *undesired* shapes and exemplified in figures. If an object was oversegmented or undersegmented and also had an undesired shape, it was only marked as oversegmented or unsegmented. This assessment was subjective but the aim was to only note obvious cases, such as when an L-shaped cluster was fitted with a line shape. Consequently no assessment was made to verify the accuracy of the shape's poses with respect to the direction the vehicle is travelling. When an oversegmented object was found for the proposed algorithm, the overlap factors of the shapes which constituted that object were compared. If the shape with the highest overlap factor was also a shape that would give a good representation of the vehicle, this was noted. An example of this particular case is illustrated in Figure 4.14. Here the overlap factor is expected to be higher for the shape, s_1, on the primary side, since it consists of more detections from multiple layers compared to the shapes s_2, s_3 and s_4 that represent detections on the secondary side.

In a report published by the U.S. Department of Transportation [39], Najm et al. provide recommendations for test scenarios to use for assessment of integrated vehicle based safety systems based on the most common pre-crash scenarios for light vehicles and heavy duty trucks. These include base test scenarios for imminent, rear-end, lane change, run-off-road and multiple threat collisions. Ideally sequences involving all these scenarios should be covered thoroughly to verify the algorithms performance from a road safety perspective but that is not feasible within the scope of this thesis. Instead due to the focus on shape extraction, a lane change situation, scenario 1, have been prioritised which aimed to evaluate consistency of the shape when a preceding vehicle changes lane. The scenario is outlined in Figure 4.15.



(a) Oversegmented object consisting of four shapes, three lines s_1, s_2 and s_4, and one point, s_3.

(b) Ideal segmentation of an object, only one shape, an L-shape, s_1.

Figure 4.14: Resulting shape extraction for detections on the vehicle above should ideally be the L-shape shown in (b), but could for instance be oversegmented into four shapes as illustrated in (a). In (a), the vehicle's side represented by s_1, is the most important side to obtain for tracking due to it containing the most information about distance from the sensor to the vehicle as well as the vehicle's direction. The overlap factor is aimed assist in distinguishing which of the shapes that represent this side.

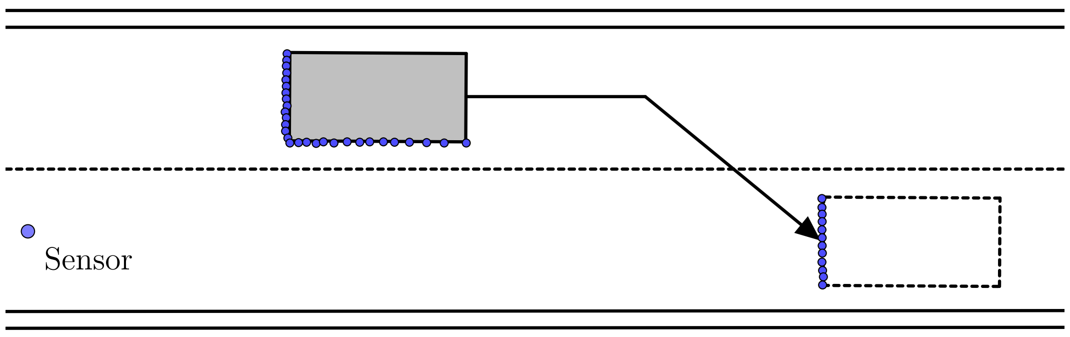


Figure 4.15: Scenario 1, where a preceding vehicle in the left lane with respect to the ego vehicle initiates a lane change. The ego vehicle is represented by the sensor in the figure. The shape of the detections on the vehicle change from an initial L-shape to a final vertical line.

For this scenario the shape of the vehicle was expected to change from an L-shape to a vertical line. Evaluating this scenario was aimed at observing the transition between these shapes as well as evaluating the shape extraction under the conditions stated above. For comparison, the reference algorithm was used with and without

GPR in order to observe the impact of GPR on segmentation and shape extraction.

In Section 4.2.4, regarding cluster merging it was mentioned that using the shapes of different clusters in merging could make it possible to differentiate between clusters containing mainly ground points and those that mainly represent a vehicle on the road. To test this, three scenarios similar to the case presented in Figure 4.11 have been used, starting with Scenario 2 which is detailed in Figure 4.16.

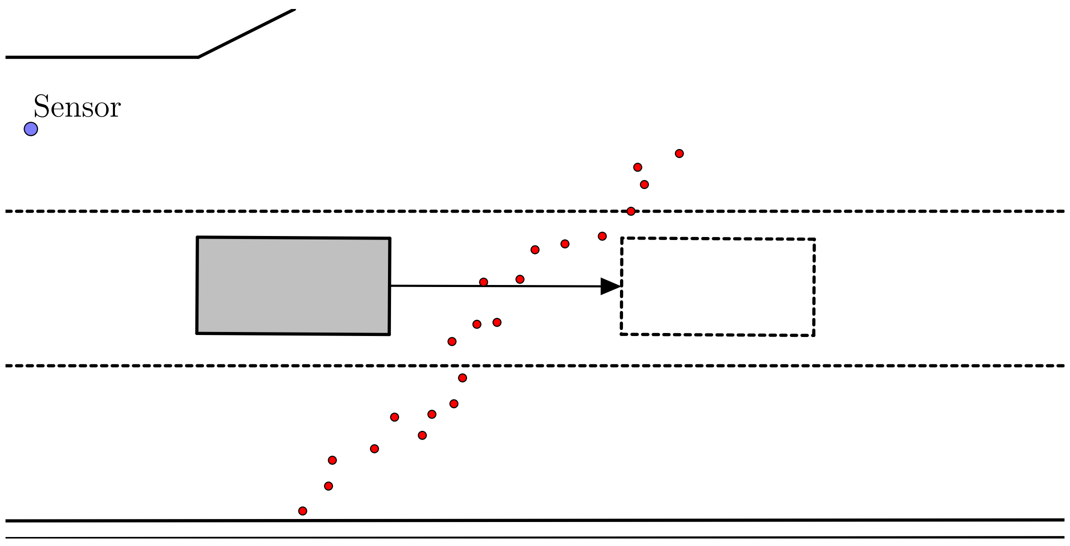


Figure 4.16: In scenario 2 the ego vehicle is stationary, waiting to turn left while facing a small incline road. A vehicle in the right lane passes through an area where a large set of ground detections are obtained.

Scenario 3 captures a situation where a preceding vehicle is travelling between traffic islands. The raised islands result in several detections near the vehicle which complicate shape extraction as also ground detections are present. An approximate illustration of this situation is shown in Figure 4.17.

Finally in scenario 4 the ego vehicle is turning left in an intersection. During this maneuver the sensor sweeps over a stationary car waiting in a slope. Due to the height difference caused by the slope many ground points are obtained around the car, making proper segmentation challenging. The scenario is illustrated in Figure 4.18.

In scenario 1-4, shape extraction for only one vehicle was considered. There was a difference at what distance the vehicle detections occurred in each scenario, and the approximate ranges of those distances are given in Table 4.3.

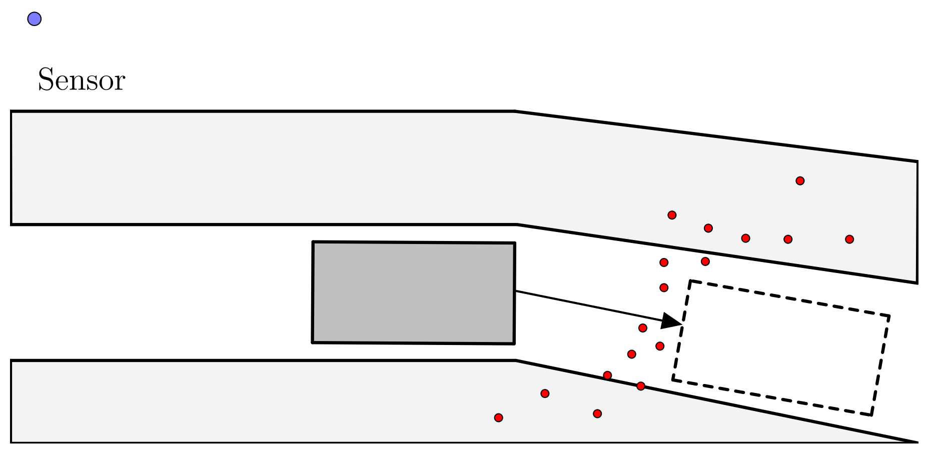


Figure 4.17: In scenario 3, the ego vehicle is moving forward, rightward in the figure. On its right side a vehicle is passing between traffic islands which cause several detections. Additionally ground detections close to the vehicle occur. This combination of detections close to the vehicle is challenging to handle and could possibly lead to undersegmentation. Note that detections on the vehicle have not been included in the illustration.

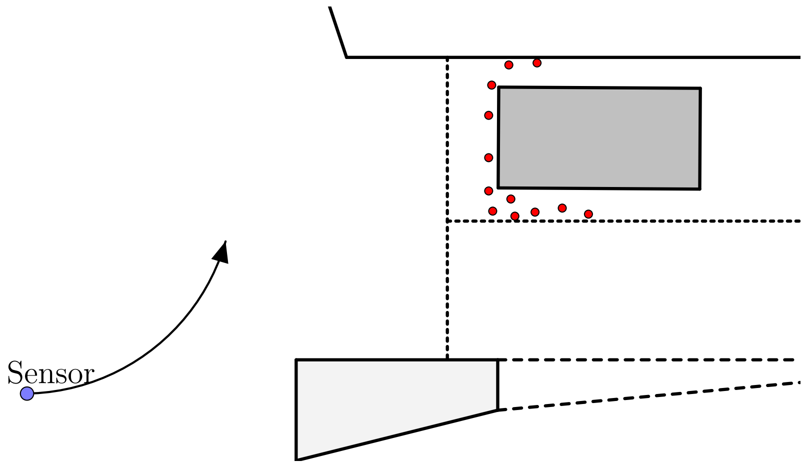


Figure 4.18: The ego vehicle is making a left turn at an intersection, while the sensor sweeps over a stationary vehicle waiting on the opposite side of the intersection. The vehicle is standing in a rising slope, which causes ground detections near the vehicle.

Table 4.3: Approximate distance to observed vehicle in scenario 1-4.

Scenario	Distance [m]
1	14 – 24
2	13 – 19
3	29 – 31
4	19 – 28

The length of the different scenarios varied, the total number of frames assessed in each scenario is summarised in Table 4.4.

Table 4.4: Number of frames assessed in each scenario.

Scenario	Number of frames
1	89
2	13
3	36
4	17

4.3.2 Reference Target Evaluation

To assist in evaluating the algorithm proposed in this work, a setup was used where an ego car is carried a lidar to collect data as described in Section 2.2. A reference vehicle, denoted “target” vehicle" was along with the ego vehicle equipped with an accurate differential GPS among other sensors to log both vehicles’ locations continuously. This setup provided a reference where lidar measurements, and subsequently extracted shapes from the proposed algorithm and the reference algorithm could be compared with the true location and heading of the detected target, see Figure 4.19. The reference algorithm was tested with ground point removal (GPR).

We call the one or two sides of the target vehicle which are visible to the sensor the "reference shape". The scenario was set up in a restricted area without any traffic apart from the ego and the target vehicles. Some errors arise from inaccuracies in synchronising the data logs from both vehicles and other factors.

For evaluating the accuracy of the shapes produced, the output shapes of the algorithm were compared to the reference shapes of the target vehicle during certain vehicle maneuvers. The primary goal of analysing this data was to assess how well the shapes obtained from the algorithm fit the target vehicle. The scenario only included one vehicle other than the ego vehicle, and few terrain structures and objects compared to a real scenario. As such, an evaluation of undersegmentation was not performed since the testing case was absent of many surrounding objects. In a real scenario, these objects typically are what cause undersegmentation by merging into the target vehicle. Oversegmentation was however assessed, as it is typically caused only by a faulty clustering and merging of the detections from the actual target, rather than from interference with the surrounding objects.

A box was created to represent the target vehicle, after which one or two of its sides were used as the reference shapes. In the case of a line shape being detected by the algorithm, the line was compared with the primary side of the target box, see Figure 4.20. In case of an L-shape being detected by the algorithm, the L was compared with both the primary and the secondary lines of the reference shape. The primary side represents the visible side of the target vehicle with the largest incidence angle, also referred to as the primary incidence angle θ_1. The secondary line represents the

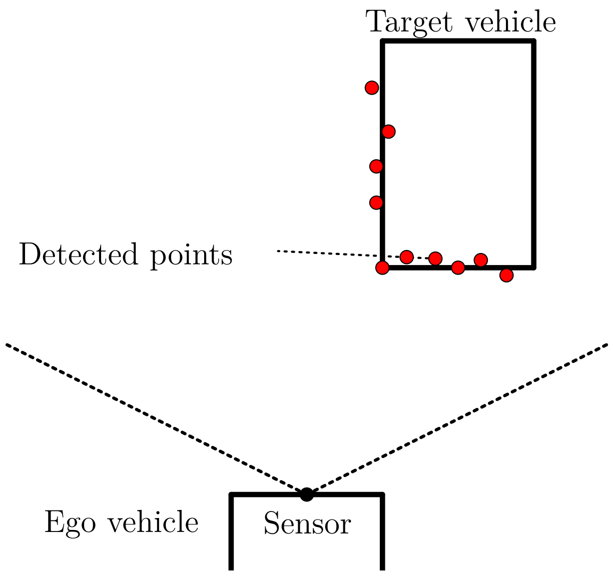


Figure 4.19: The reference car designated target vehicle is at a known location in relation to the ego vehicle. Detected points, and subsequently shapes, can be compared to the true location and heading of the target vehicle.

other visible side, corresponding to the smaller incidence angle, also referred to as the secondary incidence angle θ_2.

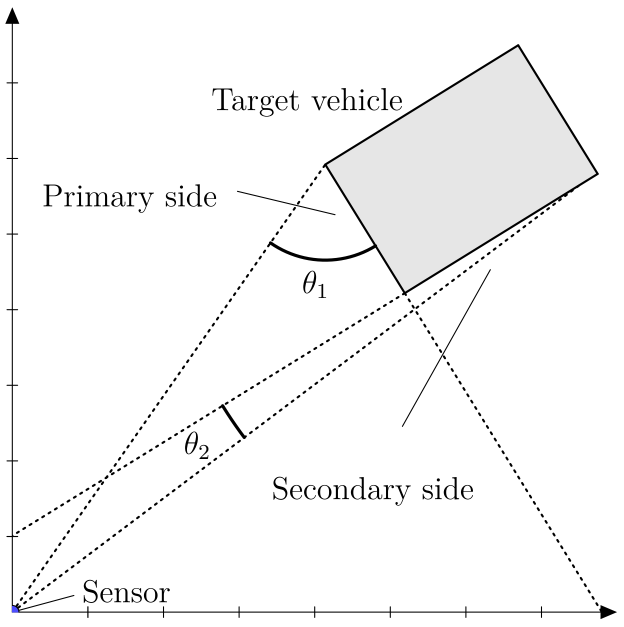


Figure 4.20: The side of the target vehicle which is visible to the sensor and has the largest (primary) incidence angle $\max\{\theta_1, \theta_2\}$ is considered the primary side. If more than one side of the target vehicle is visible to the sensor, the side with the smaller (secondary) incidence angle θ_2 is called the secondary side.

The metrics that are calculated are described in this section, and can be summarised as:

- Number of shapes within distance d_0 of the target vehicle
- Number of *valid* shapes, i.e. lines and L-shapes within distance d_0 of the target vehicle
- Angular fitting error E_{ang}
- Mean square error MSE_{prim} of primary line with best angular fit
- Mean square error MSE_{sec} of secondary line, when an L-shape is detected
- The proportion of L-shapes versus the proportion of line shapes that are output by the algorithm, at different distances between the sensor and the target D_{ST}

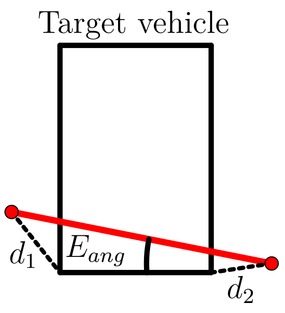
The number of shapes within distance d_0 of the target are those shapes which have at least one corresponding detection within a certain distance d_0 from the bounding box of the target vehicle. The distance is calculated in the same way for *valid* shapes, except that polygons and points are excluded from this data, as they are typically outliers. The angular fitting error E_{ang} is the angular difference between the reference line and the fitted shape.

The mean square error is the arithmetic mean of the squared distances between the endpoints of the fitted line and the reference line, see Figure 4.21. If a line shape is detected near the target vehicle, only the distance to the primary line is considered, See Figure 4.21a. If an L-shape is detected, both the primary and the secondary side are considered, see Figure 4.21b.

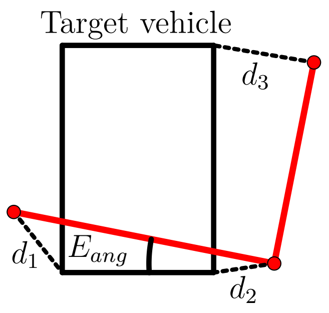
If there are multiple lines or L-shapes near the target, E_{ang}, MSE_{prim} and MSE_{sec} are not calculated.
Some informative data about the details of the evaluation scenarios is also collected, and include:

- Histogram of the number of samples N collected at certain distances to target vehicle D_{ST}.
- Histogram of the number of samples N collected at certain secondary incidence angles θ_2
- The magnitude of the primary and secondary incidence angles θ_1 and θ_2 over different distances to the target vehicle D_{ST}

Two different scenarios are used, where the ego and the target vehicle repeatedly perform certain maneuvers, see Figure 4.22. For both cases, three repetitions of the same scenario were sampled. The two different scenarios consist of:



(a) The mean square error of the primary side MSE_{prim} is calculated as $MSE_{prim} = \frac{1}{2}(d_1^2 + d_2^2)$ using the distances d_1 and d_2 between the endpoints of the fitted red line and the primary reference line of the target vehicle. The angle E_{ang} is the angular fitting error.



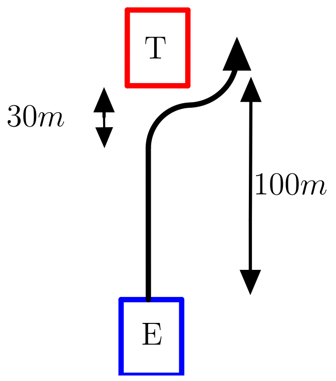
(b) The mean square error of the secondary side MSE_{sec} is calculated as $MSE_{sec} = \frac{1}{2}(d_2^2 + d_3^2)$ using the distances d_2 and d_3 between the endpoints of the fitted red L and the secondary reference line of the target vehicle. The angle E_{ang} is the angular fitting error.

Figure 4.21: The mean square errors MSE_{prim} and MSE_{sec} are calculated using the distances from the endpoints of the fitted line or L to the reference line. The angular fitting error E_{ang} is the angular difference between the fitted line or L and the primary reference line.

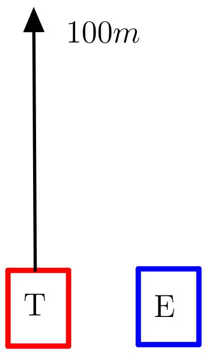
1. Ego overtaking, where the ego vehicle overtakes the target vehicle. The scenario consists of a straight approach by the ego vehicle from behind the target vehicle for most of the scenario, and is completed by a quick turn and overtaking when the target at a distance of $D_{ST} \approx 30$m, see Figure 4.22a. The scenario begins when the distance to target $D_{ST} \approx 125$m from the ego vehicle, and ends when the target is overtaken and partially out of view from the lidar. In total, 3831 frames were sampled and analysed.
2. Target overtaking, where the target vehicle overtakes the ego vehicle on the left side. The scenario consists of a straight departure of the target vehicle from the ego vehicle in the adjacent lane, see Figure 4.22b. This scenario begins when the target is alongside the ego vehicle and fully in view of the sensor, and ends when the target has reached a distance of $D_{ST} \approx 100$m. A total of 399 frames were sampled and analysed.

4.3.3 Execution time

To compare the execution time of the proposed algorithm and the reference algorithm, the algorithms were run on data previously collected in real traffic situations. The data consisted of 6833 data frames representing driving in highway and urban traffic during good weather conditions. The simulations were run in MATLAB R2015b, on a Windows 7 PC with i7-4610M @ 3.00 Ghz CPU and 8 GB of RAM. The execution time was calculated separately for each subfunction as described in



(a) The ego vehicle overtakes the target vehicle. Starting at a distance of $D_{ST} \approx 125$m, the ego vehicle performs a straight approach in the same lane as the target. The ego vehicle turns to overtake the target at a distance of $D_{ST} \approx 30$m, and the scenario ends when the target is partially out of view of the lidar.



(b) The target vehicle overtakes the ego vehicle. The scenario begins when the target vehicle has fully entered the lidar's field of view alongside the ego vehicle. The target vehicle departs from the ego in the adjacent lane, and the scenario ends when the target is at a distance of $D_{ST} \approx 100$m.

Figure 4.22: Two scenarios were evaluated using a reference target vehicle with a known location relative to the ego vehicle. The distances in the figures are approximations.

Section 4.2.1, namely clustering, shape extraction on layer separated clusters, cluster merging and finally shape extraction on combined clusters. The execution times for each frame was stored, along with information on the number of scan points in the frame, the number of clusters formed by the clustering algorithm, and the final number of shapes produced. In addition, 400 frames were analysed with the MATLAB profile tool, which allowed some insight into which parts of the respective subfunctions consumed the most time.

5

Results

In this chapter the results of the evaluation described in Section 4.3 are presented. The results consist of three main parts. First in Section 5.1 the results of the manual assessment of segmentation is presented. Next, in Section 5.2 results of the algorithms with respect to reference target data as detailed in Section 4.3.2 are given. And finally in Section 5.3 the execution times of the proposed algorithm and the reference algorithm are presented.

5.1 Manual Evaluation

The manual evaluation is composed of two parts, the first one is focusing on shape consistency of shape extraction for a preceding vehicle during a lane change, scenario 1. Secondly, scenarios 2, 3 and 4, with ground detections, have been assessed. Figures presented in this section are illustrative but closely resemble observations made on real data.

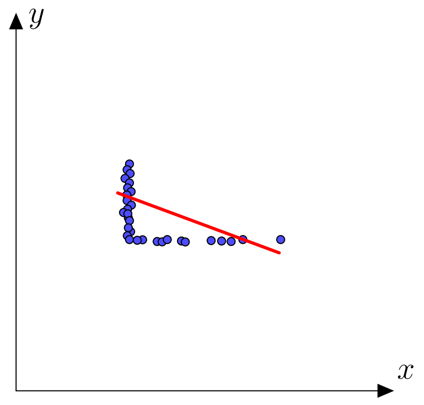
5.1.1 Scenario 1

Scenario 1, as described in Figure 4.15, includes a sequence which has been manually assessed for the designed algorithm and for the reference algorithm, with and without Ground Point Removal (GPR).

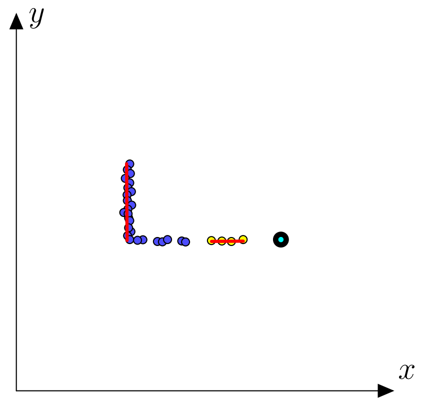
The results are presented in Table 5.1. As can be observed almost all cases of wrong shapes for each algorithm were due to oversegmentation. No undersegmentation was observed. However, for the proposed algorithm there was one case in which the extracted shape was not oversegmented but it was incorrectly classified as a line instead of an L as can be observed in Figure 5.1a. For comparison purposes, the shape extraction of the vehicle in the same frame for the reference algorithm without GPR has also been included, Figure 5.1b.

Table 5.1: Results for scenario 1, showing the percentage of frames, in which the extracted shapes were wrong and what the reason was. P alg. refers to the 'Proposed algorithm' and Ref. alg refers to the 'Reference algorithm'.

Reason for wrong shape	P. alg.	Ref. alg. without GPR	Ref. alg. with GPR
Undesired shape	1,12%	0%	0%
Oversegmentation	31.46%	44.94%	32.58%
Total	32.58%	44.94 %	32.58%



(a) Proposed algorithm



(b) Reference algorithm without GPR

Figure 5.1: Resulting shape extraction for a specific frame for the proposed algorithm and the reference algorithm. In (a) the proposed algorithm fit a the wrong shape and for the same case the reference algorithm with GPR suffer from oversegmentation as seen in (b).

Regarding oversegmentation, this occurred slightly more for the reference algorithm without GPR, compared to the proposed algorithm and the reference algorithm with GPR which performed very similarily. For all algorithms, it was common that the oversegmentation consisted of an assortment of points and lines instead of an desired L-shape. The majority of frames where oversegmentation was observed occurred when the distribution of detections for the vehicle transitioned from an L-shaped cluster to a line shape. In this transition the detections on the secondary side of the vehicle gradually became sparser as an example show in Figure 5.2. Furthermore, it was observed during oversegmentation with the proposed algorithm, that the overlap factor in all cases was highest for the primary side of the vehicle as expected.

There was some difference in performance between the reference algorithm with and without GPR. Mainly, that the configuration without GPR had more instances of oversegmentation.

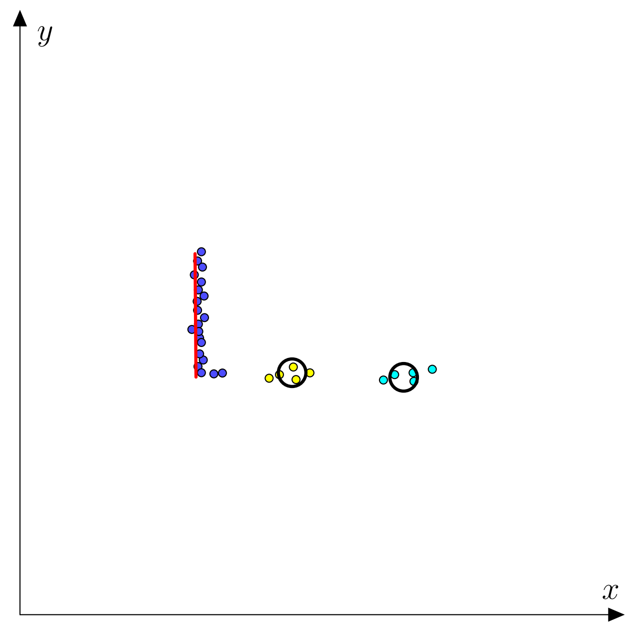


Figure 5.2: Detections on the vehicle in scenario 1 in a sample where the detections are noticeably sparser on the secondary side resulting in oversegmentation for the proposed algorithm.

5.1.2 Scenarios 2, 3 and 4

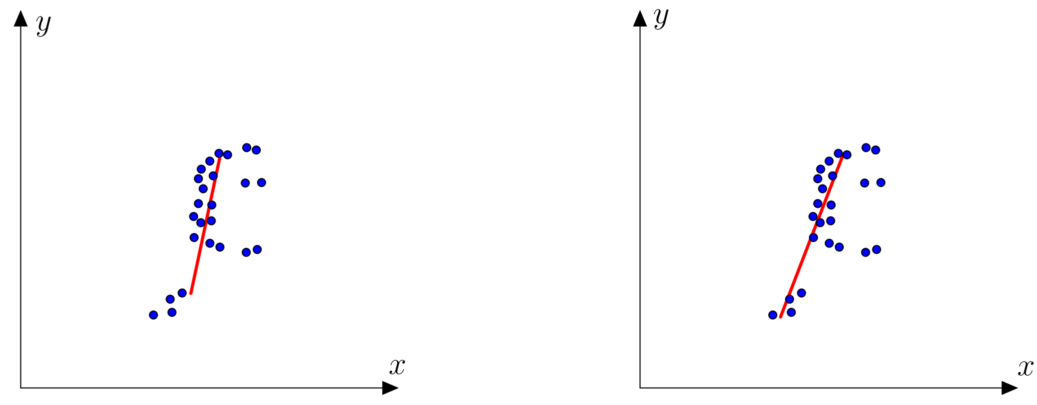
In these scenarios, different sequences containing ground points were manually assessed. The results are presented in Table 5.2 for each scenario separately.

Table 5.2: Results for manual assessment of scenario 2, 3 and 4, with the proposed algorithm and the reference algorithm with GPR.

	Scenario 2		Scenario 3		Scenario 4	
Reason for wrong shape	P alg.	Ref. alg.	P alg.	Ref. alg.	P alg.	Ref. alg.
Undesired shape	0 %	0 %	0 %	16.67%	0%	0%
Undersegmentation	23.08%	0%	25%	25%	35.29%	76.47%
Oversegmentation	0 %	23.08%	0%	0%	0%	0%
Total	23.08%	23.08%	25%	41.67%	35.29%	76.47%

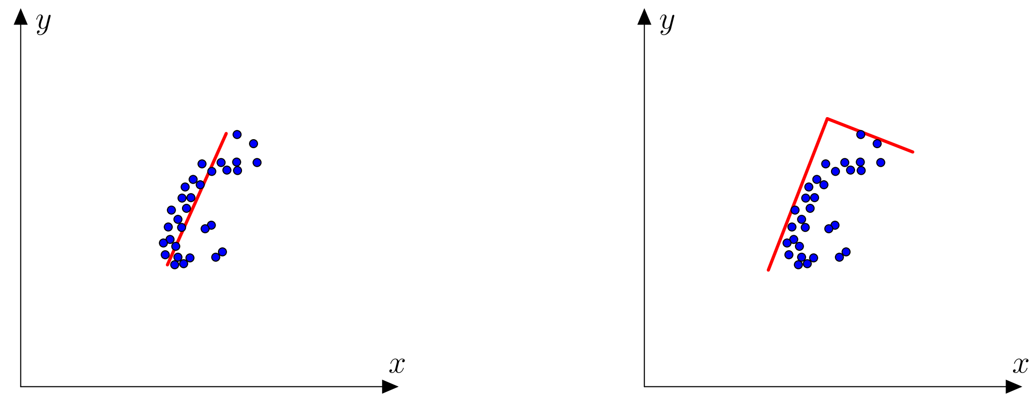
One case of undersegmentation observed in scenario 3 is exemplified in Figure 5.3. For both the proposed algorithm and the reference algorithm the cluster indicating the tail end of the vehicle and a cluster consisting of ground points are clustered. As can be seen the only difference is the shape extracted, but the result is undesired for both cases. Furthermore, in the same scenario, 16.67% percent of the points of the wrong shapes produced by the reference algorithm were correctly segmented but generated an undesired shape. This was mostly cases were an L-shape was produced instead of a desired line, an example of this is illustrated in Figure 5.4.

As can be seen for this particular case the proposed algorithm instead extract a line.



(a) Proposed algorithm **(b)** Reference algorithm with GPR

Figure 5.3: The figures illustrate the resulting shape extraction for a specific frame for the proposed algorithm and the reference algorithm with GPR. In (a) and (b) both algorithms cluster ground detections and vehicle detections together.



(a) Proposed algorithm **(b)** Reference algorithm with GPR

Figure 5.4: The figures illustrate the resulting shape extraction for a specific frame for the proposed algorithm and the reference algorithm. In (a) the proposed algorithm manages to extract the desired shape, namely a line, while in the (b) the reference algorithm fails to do so and extracts an L-shape.

The results considering all the sequences with ground points together are given in Table 5.3. As can be observed, the proposed algorithm perform better overall under the determined assessment criteria. However, looking at scenario 2 in Table 5.2 the percentage of frames with wrong shapes are the same for both algorithms. But the reason is different, for the proposed algorithm the issue is oversegmentation due to ground points. For the reference algorithm, most of the ground detections were successfully removed by GPR, but reducing the data from detections at this range instead increased the oversegmentation. The greatest difference between the algorithms was observed in scenario 4, were both algorithms suffered from undersegmentation but the reference algorithm more so than the proposed algorithm. It should be noted that the ground points in this scenario was mostly detected at a greater height than in the other scenarios due to the slope the vehicle was standing in, which the GPR for the reference algorithm seemed unable to manage.

Table 5.3: The result of the manual assessment for scenario 2, 3, and 4 combined.

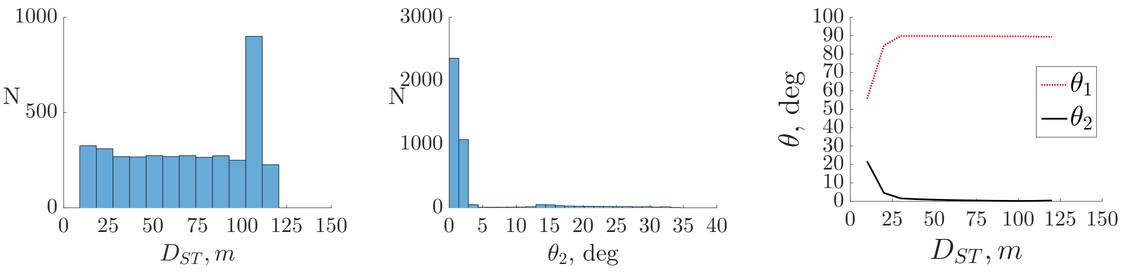
	Scenario 2-4	
Reason for wrong shape	P alg.	Ref. alg
Undesired shape	0%	9.09%
Undersegmentation	27.27%	33.33%
Oversegmentation	0%	4.55%
Total wrong shape	27.27%	46.97%

5.2 Reference Target

In this section, results are presented for the evaluation method detailed in Section 4.3.2. The evaluation consists of an ego vehicle, on which the lidar is mounted, and a target vehicle. The locations of the ego vehicle and the target vehicle are known by logging an accurate GPS, so a reference shape can be extracted from the target car and compared to the outputs of the algorithm. Both the proposed algorithm, detailed in Section 4.2 and the reference algorithm, mentioned in Section 4.3 are evaluated in two different scenarios.

5.2.1 Ego Overtaking

The scenario consists of the ego vehicle approaching the target vehicle in a straight line at relatively constant speed, and performing a quick lane change towards the end of the maneuver. In Figure 5.5a, the number of frames are distributed evenly over different distances from the sensor to the target vehicle D_{ST}, ranging from 10m to 120m. The vast majority of the frames are collected from the approaching phase, while the overtaking maneuver only constitutes a small portion of each test. The results of this can be seen in Figure 5.5b, where the concentration of secondary incidence angles, θ_2, under 5° indicates that the target was approached straight from behind. In Figure 5.5c, it can be observed that the secondary incidence angle is only larger than 5° below approximately 25m, indicating that the overtaking maneuver starts at that distance from the target vehicle.



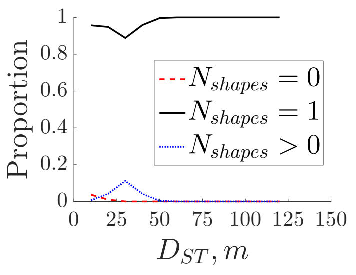
(a) Shows the number of samples N collected with distance to target vehicle D_{ST}. The samples are evenly spread between 10m and 120m, with a peak at 100-110m.

(b) Shows the number of samples N collected at different secondary incidence angles θ_2 of the target vehicle. Nearly all data is sampled with $\theta_2 < 5°$.

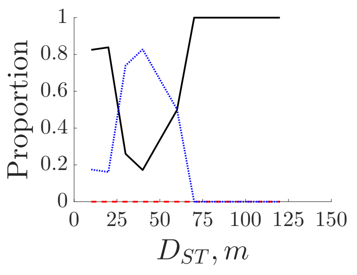
(c) Primary θ_1 and secondary θ_2 incidence angles values at varying sensor to target distances D_{ST}.

Figure 5.5: Samples collected with target distance evenly spread between 10m and 120m, with a peak at 100-110m. The θ_2 is below 5° for nearly all frames.

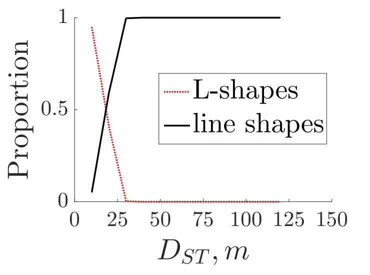
Both the proposed algorithm and the reference algorithm performed similarly with regards to segmentation accuracy, as can be seen in Figure 5.6. Figures 5.6a and 5.6d show that both algorithms only outputs a single line or L-shape in the majority of cases, but that both suffer from some oversegmentation of valid shapes at a distance of approximately 30m from the target. This coincides with the overtaking maneuver, and indicates that more than one line or L-shape is detected on the same object at these occasions. Figures 5.6b and 5.6e show that more than one shape is detected in a majority of frames between approximately 30m and 60m from target. By comparing with Figures 5.6a and 5.6d, which only show lines or L-shapes, one can deduct that most of these shapes are points and polygons.



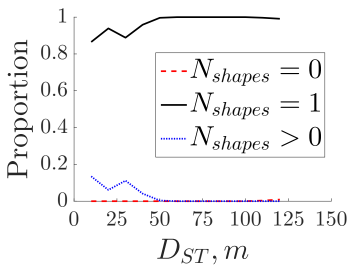
(a) Proposed algorithm: Shows the proportion of frames which has N_{shapes} valid shapes close to the target vehicle at different distances.



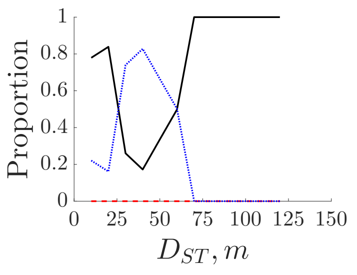
(b) Proposed algorithm: Shows the proportion of frames which have N_{shapes} total shapes close to the target vehicle at different distances.



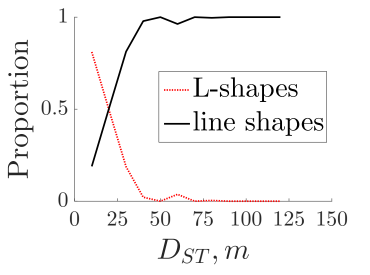
(c) Proposed algorithm: Shows the proportion of L-shapes and lines at varying distance from the target.



(d) Reference algorithm: Shows the proportion of frames which have N_{shapes} valid shapes close to the target vehicle at different distances.



(e) Reference algorithm: Shows the proportion of frames which have N_{shapes} shapes close to the target vehicle at different distances.

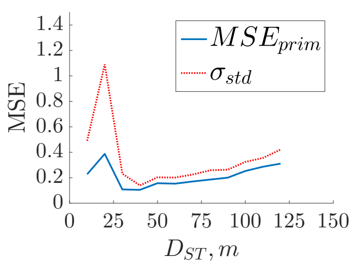


(f) Reference algorithm: Shows the proportion of L-shapes and lines at varying distance from the target.

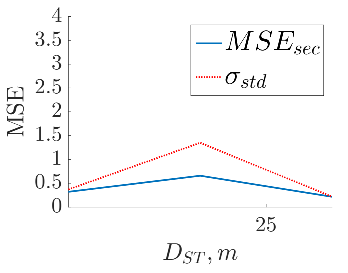
Figure 5.6: Proportion of frames with different numbers of shapes in the vicinity of the target vehicle at varying distances for the designed algorithm in (a) and (c), and the reference algorithm in (b) and (d). (e) and (f) show the proportion of L-shapes versus lines among the valid shapes, at varying distances to the target vehicle. The proposed algorithm and the reference algorithm provide largely identical results.

Figure 5.7 shows the different fitting errors for the proposed algorithm, and the reference algorithm. Figures 5.7a and 5.7d indicate that the proposed algorithm has a similar fitting error to the reference algorithm on the primary side of the target vehicle. Likewise, in 5.7c and 5.7f, the average errors of both algorithms are similar. One can observe that the standard variations for the reference algorithm in 5.7d and 5.7f are larger than the proposed algorithm in the interval between 50m and 75m.

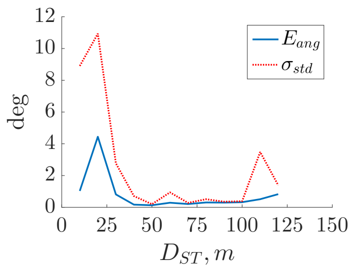
Figures 5.7b and 5.7e represent the fitting error of the secondary side of the target vehicle, meaning representing the side with the lower incidence angle for L-shapes only. Lines do not produce a secondary side fit, which explains the short range of the distance to target D_{ST} for both these figures. The proportion of L-shapes versus lines among the valid shapes is illustrated in Figures 5.6c and 5.6f for the proposed algorithm, and reference algorithm respectively. It is evident that the vast majority of all detections are of lines, with L-shapes only occurring at distances closer than 30m. In this range, the Figures 5.7b and 5.7e indicate that the proposed algorithm has more accurate secondary side fits than the reference algorithm.



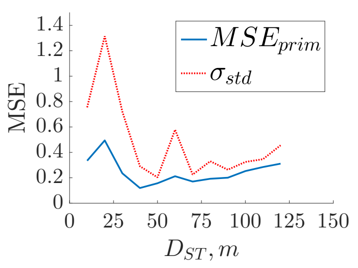
(a) Proposed algorithm: Primary side fitting error MSE_{prim}.



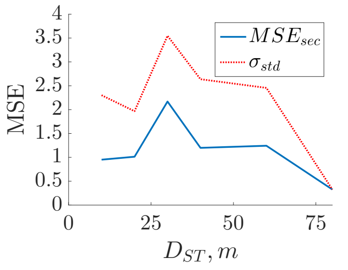
(b) Proposed algorithm: Secondary side fitting error MSE_{sec}.



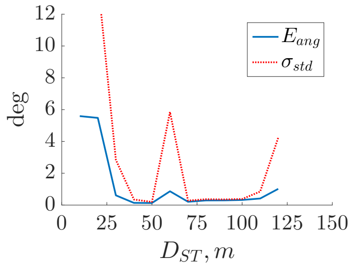
(c) Proposed algorithm: Angular fitting error E_{ang}.



(d) Reference algorithm: Primary side fitting error MSE_{prim}.



(e) Reference algorithm: Secondary side fitting error MSE_{sec}.

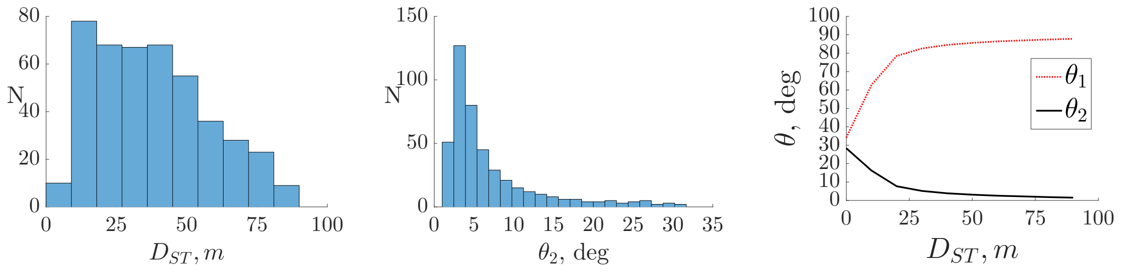


(f) Reference algorithm: Angular fitting error E_{ang}.

Figure 5.7: Fitting errors for the proposed algorithm in Figures (a), (b) and (c), and for the reference algorithm in Figures (d), (e) and (f). MSE_{prim} in (a) and (d), are similar for both algorithms, but with more stable results in the proposed algorithm. The same fact can be observed for MSE_{sec} in Figures (b) and (e), and E_{ang} in Figures (c) and (f).

5.2.2 Target Overtaking

The scenario consists of the target vehicle approaching the ego vehicle from behind, in a straight line at relatively constant speed. The target car overtakes the ego car and keeps going for approximately 100m in a straight line in the adjacent lane. The results only cover the moment after which the entire target car enters the field of view of the sensor. In Figure 5.8a, the spread of frames over varying distances show that the amount of samples are diminishing with the increasing distances from the sensor to the target vehicle D_{ST}. Since the target car keeps driving in the adjaceny lane for the duration of the scenario, two sides of the car are always within the sensor's field of view, however the secondary incidence angle θ_2 shrinks with increasing distance as can be seen in Figures 5.8b and 5.8.



(a) Shows number of samples N collected at distance to target vehicle D_{ST}. The samples are evenly spread when 10m < D_{ST} < 50m, and diminish with when D_{ST} > 50m.

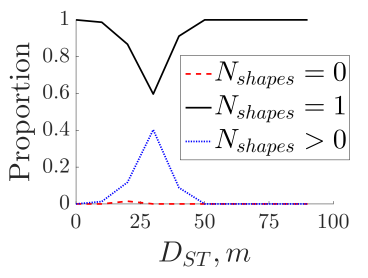
(b) Shows number of samples N collected at different secondary incidence angles θ_2 of target vehicle. The amount of sampled data follows an exponential decay with the number of samples N decreasing as the secondary incidence angle θ_2 increases.

(c) Primary θ_1 and secondary θ_2 incidence angles values at varying sensor to target distances D_{ST}.

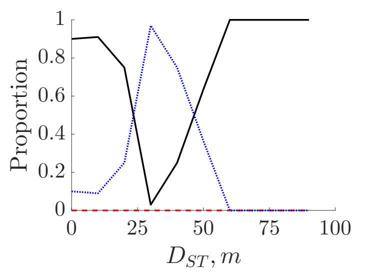
Figure 5.8: The collected data samples for different sensor to target distances D_{ST} are evenly spread when 10m < D_{ST} < 50m, and diminish with when D_{ST} > 50m. The amount of sampled data per secondary incidence angle θ_2 follows an exponential decay with the number of samples N decreasing as θ_2 increases.

Both the proposed algorithm and the reference algorithm performed similarly with regards to segmentation accuracy, as can be seen in Figure 5.9. Figures 5.9a and 5.9d show that both algorithms only outputs one valid shape in the majority of cases, but that both suffer from some oversegmentation at a distance of approximately 30m from the target. This coincides with the moment where the output shapes transition from mainly L-shapes, to mainly lines as the target vehicle moves further from the ego vehicle, decreasing the secondary incidence angle θ_2. This transition can be seen in Figures 5.9c and 5.9f.

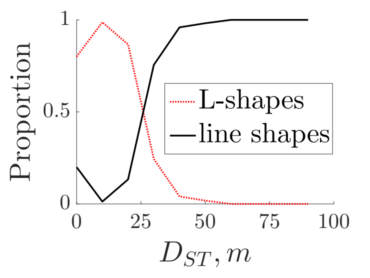
Figures 5.9b and 5.9e show that a more than one shape is detected in a majority of frames between approximately 20m and 50m from target. By comparing with Figures 5.9a and 5.9d which only shows lines or L-shapes, one can deduct that most of these shapes are points and polygons.



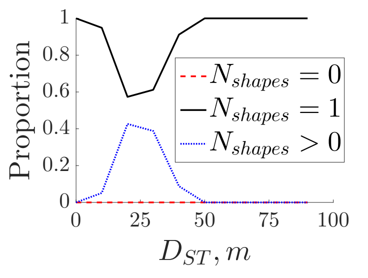
(a) Proposed algorithm: Shows the proportion of frames which has N_{shapes} valid shapes close to the target vehicle at different distances.



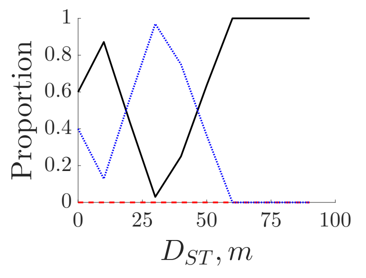
(b) Proposed algorithm: Shows the proportion of frames which have N_{shapes} shapes close to the target vehicle at different distances.



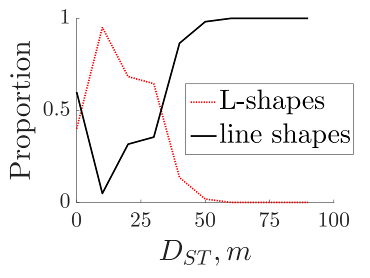
(c) Proposed algorithm: Shows the proportion of L-shapes and lines at varying distance from the target.



(d) Reference algorithm: Shows the proportion of frames which have N_{shapes} valid shapes close to the target vehicle at different distances.



(e) Reference algorithm: Shows the proportion of frames which have N_{shapes} shapes close to the target vehicle at different distances.



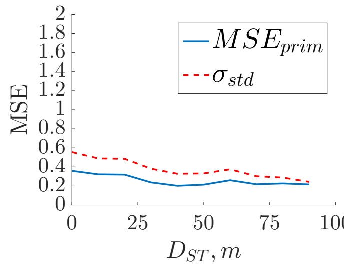
(f) Reference algorithm: Shows the proportion of L-shapes and lines at varying distance from the target.

Figure 5.9: Proportion of frames with different numbers of shapes in the vicinity of the target vehicle at varying distances for the designed algorithm in (a) and (c), and the reference algorithm in (b) and (d). Subfigures (e) and (f) show the proportion of L-shapes versus lines among the valid shapes, at varying distances to the target vehicle. The proposed algorithm and the reference algorithm provide largely identical results.

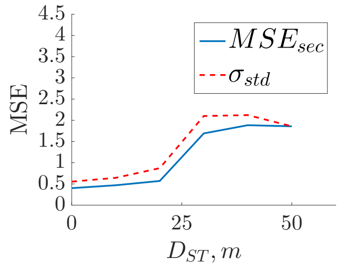
Figure 5.10 shows the different fitting errors for the proposed algorithm, and the reference algorithm. Figures 5.10a and 5.10d indicate that the proposed algorithm has a similar fitting error to the reference algorithm on the primary side of the target vehicle when $D_{SR} > 45$m, but that the proposed algorithm performs better when $D_{SR} < 45$m. Likewise, in 5.10c and 5.10f, the average errors of both algorithms are similar for $D_{SR} > 55$m, while the proposed algorithm performs substantially better when $D_{SR} < 55$m.

One can observe that the largest errors MSE_{prim} and E_{ang} for both algorithms in Figures 5.10a, 5.10d, 5.10c and 5.10f occur at a distance of $D_{ST} \approx 30$m, which again coincides with the transition between lines and L-shapes as the main descriptor for the target, as was seen in Figures 5.9c and 5.9f.

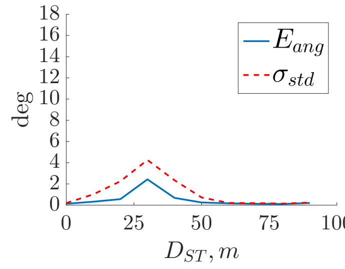
Figures 5.10b and 5.10e show that the secondary fitting error of both algorithms are roughly similar. The performance of both fits start to deteriorate when $D_{ST} > 25$, which coincides with the decreased frequency of L-shape outputs from both algorithms as seen in Figures 5.9c and 5.9f.



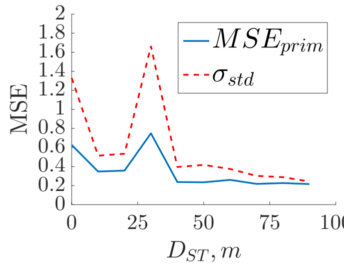
(a) Proposed algorithm: Primary side fitting error MSE_{prim}.



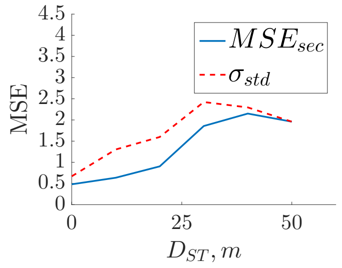
(b) Proposed algorithm: Secondary side fitting error MSE_{sec}.



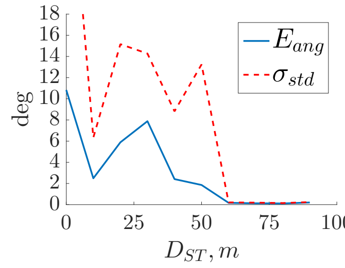
(c) Proposed algorithm: Angular fitting error E_{ang}.



(d) Reference algorithm: Primary side fitting error MSE_{prim}.



(e) Reference algorithm: Secondary side fitting error MSE_{sec}.



(f) Reference algorithm: Angular fitting error E_{ang}.

Figure 5.10: Fitting errors for the proposed algorithm in Figures (a), (b) and (c), and for the reference algorithm in Figures (d), (e) and (f). The proposed algorithm outperform the reference algorithm with regards to primary side fitting error MSE_{prim} in (a) and (d) and angular fitting error E_{ang} in (c) and (f). At shorter distances $D_{ST} < 30$, and perform similarly for longer distances. The algorithms also appear equivalent with regards to secondary side fitting error MSE_{sec} in (b) and (e).

5.3 Execution time

Data on the execution times were sampled on recorded data as briefly described in Section 4.3.3. The execution times of both the proposed algorithm and the reference algorithm were considered. The average execution time of the proposed algorithm was significantly slower than the reference algorithm, as can be seen in Table 5.4

Table 5.4: The average execution time T_{avg} and the standard deviation σ for the proposed algorithm and the reference algorithm.

Algorithm execution time, seconds		
	T_{avg}	σ
Proposed algorithm	0.287	0.134
Reference algorithm	0.039	0.026

For each subfunction of the proposed algorithm described in Section 4.2 and the preprocessing step described in Section 4.1, the execution time was measured, which can be observed in Figure 5.11. The steps consuming the most time were the two final steps, namely the cluster merging described in Section 4.2.4 and the second shape extraction described in Section 4.2.3, constituting 39.9% and 35.2% of the total execution time respectively. The clustering, described in Section 4.2.2 and the first shape extraction each took 12.8% and 12.0% of the total execution time respectively, while the execution time of the preprocessing was less than 0.1%.

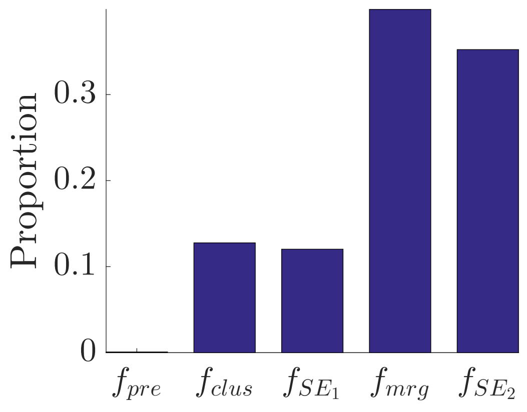
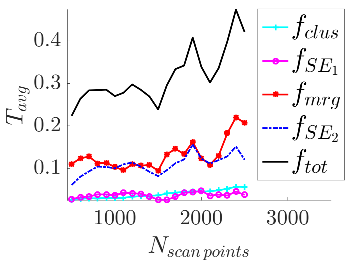
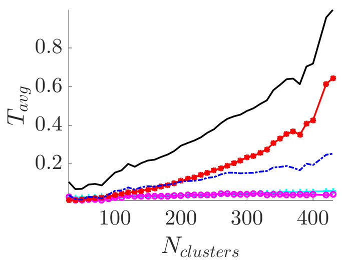


Figure 5.11: Proportion of total execution time on average spent in each step of the proposed algorithm, preprocessing f_{pre} ($< 0.1\%$), clustering f_{clus} (12.8%), initial shape extraction f_{SE_1} (12.0%), cluster merging f_{mrg} (39.9%) and final shape extraction f_{SE_2} (35.2%)

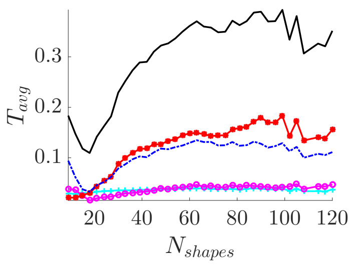
The average execution time T_{avg} for different number of scan points, number of clusters before merging and final number shapes are shown in Figure 5.12. The figure shows a noisy trend of T_{avg} increasing with a higher number of scan points $N_{scan\,points}$ in Figure 5.12a. In Figure 5.12b however there is a clear correlation with increasing T_{avg} for a higher number of clusters $N_{clusters}$. The number of shapes N_{shapes} in Figure 5.12c appears to be correlated with T_{avg} for lower numbers of N_{shapes} until T_{avg} flattens out when $N_{shapes} > 60$. The trend is most obvious for the case of clusters, implying that the number of clusters obtained in the clustering subfunction heavily affects the overall execution time.



(a) Average execution time T_{avg} for different numbers of scan points $N_{scan\,points}$ in the frame. The execution times of f_{tot}, f_{mrg} and f_{SE_2} appears to be correlated to $N_{scan\,points}$.



(b) Average execution time T_{avg} for different numbers of clusters $N_{clusters}$ in the frame. The execution times of f_{tot} and f_{mrg} appears to be heavily correlated to $N_{clusters}$.



(c) Average execution time T_{avg} for different numbers of scan points N_{shapes} in the frame. The execution times of f_{tot}, f_{mrg} and f_{SE_2} appears to be approximately correlated to N_{shapes} when $N_{shapes} < 60$, but remains unchanged for growing N_{shapes} when $N_{shapes} > 60$.

Figure 5.12: Average execution time T_{avg} for different numbers of scan points $N_{scan\,points}$ (a), clusters before merging $N_{clusters}$ (b) and final number of shapes N_{shapes} (c). The execution times of the different subfunctions of the proposed algorithm f_{tot} include clustering f_{clus}, primary shape extraction f_{SE_1}, cluster merging f_{mrg} and secondary shape extraction f_{SE_2}. f_{tot} is the sum of the subfunctions. $N_{clusters}$, as seen in (b) appears to have the strongest correlation with T_{avg} for the total execution time f_{tot}.

The execution time of a subset of 400 frames was further analysed using the profile tool in MATLAB. The average execution time was 0.45 seconds per frame, which is higher than the average in Table 5.4. However the proportion of time spent in the different subfunctions was similar to those previously established in Figure 5.12, with the merging subfunction f_{mrg} and the final shape extraction f_{SE_2} composing 37% and 35% of the total execution time, respectively. The purpose was to pinpoint the most time consuming parts of f_{mrg} and f_{SE_2}.

For f_{mrg}, the most time consuming parts were Algorithm 3: AllowMergeSingleLinkage, and the DistanceBetweenConvexHulls function on line 12 in Algorithm 3. Both these functions are detailed in Section 4.2.4 and accounted for 34% and 24% of the time spent in this step respectively.

For f_{SE_2} the most time consuming parts were L-shape fitting (42.5%), concave hull fitting (21%) and line fitting (20%). All these subfunctions are detailed in Section 4.2.3. The concave hull fitting includes an initial convex hull fitting, as explained in Section 4.2.3.1 and was observed in the same way to only constitute 2% of the concave hull execution time.

6
Discussion and Future Work

In this chapter the results presented in Chapter 5 are discussed in detail. The aim is to clarify and reason about the differences seen in the results from the evaluation of the proposed algorithm and the reference algorithm. Furthermore a discussion on the reliability and the value of the produced results is provided. In addition, suggestions for future work to improve the proposed algorithm are given at end of the chapter.

6.1 Clustering

As detailed in Section 1.2, one of the fundamental problems to be solved by the designed algorithm was reliable clustering of data. The clustering was evaluated manually in Section 5.1 and using reference target data presented in Section 5.2.

The results from the manual assessments of scenario 1 in Section 5.1.1 show that segmentation achieved by the proposed algorithm was similar to that of the reference algorithm when it used GPR, as can be seen in Table 5.1. In the transition between L-shapes and lines, the detections on the secondary side of the preceding vehicle were seen to become sparse compared to the primary side, which made the clustering less reliable as seen in Figure 5.2. This coincided with the frames that were most oversegmented, indicating that a particularly difficult case to achieve correct segmentation is in the transition phase from an L-shape to a line. It should be noted that the evaluation criteria decided on in Section 4.3.1 were very conservative, such that any case where there was any noticeable oversegmentation was labelled as incorrect. However, in many cases the oversegmented object also included a well represented shape, which could have been used as long as it could have been distinguished from the other shapes. For the proposed algorithm it was observed that the shape with the largest overlap factor consistently yielded the best shape to describe the cluster. This indicates that the overlap factor can be useful in scenarios where multiple shapes are detected on a single object.

Scenario 2, 3 and 4 were presented in Section 5.1.2, with the aim to observe segmentation in the presence of ground detections. It was found that the proposed algorithm frequently suffered from undersegmentation, while the reference algorithm had more inconsistent results, as it additionally suffered from oversegmentation and faulty classification. The inconsistency in the results for the reference algorithm is likely due to the different situations in each scenario with respect to distance to the vehicle, terrain and at what height ground detections occurred. The most striking difference between the two algorithms was found in scenario 2 where the proposed algorithm suffered from undersegmentation and the reference algorithm from oversegmentation. This is mainly believed to be related to the reference algorithm's use of a subfunction for ground removal, GPR. GPR did in this case remove detections on the vehicle rather than just the ground detections, which here lead to oversegmentation. This is a example where the GPR policy of removing data to avoid ground detections instead lead to oversegmentation, which supports the approach taken with the proposed algorithm to retain as much data as possible.

In scenario 4, undersegmentation was especially prevalent for the reference algorithm. The ground detections that occurred around the vehicle in the slope were not managed by the GPR in the reference algorithm and the proposed algorithm instead showed an edge in this scenario. However, the slope caused a lot of ground detections that both algorithms struggled to distinguish from the vehicle in a satisfactory manner. Overall, the results from scenario 2-4 seem to indicate that the proposed algorithm is more reliable and consistent at distinguishing between the vehicle and ground points in different scenarios.

For the reference target evaluation in Section 5.2 the undersegmentation differences were hard to assess due to the absence of other objects around the target vehicle, as discussed in Section 4.3.2, however it was possible to get a measure on oversegmentation. The results from ego overtaking in Section 5.2.1 and target overtaking in Section 5.2.2 again does not show any significant segmentation difference between the proposed algorithm and the reference algorithm. This indicates that the two algorithms are nearly equally good at segmentation in a situation where ground points are rare.

As mentioned above, in the manual evaluation of scenario 1, the vast majority of oversegmention cases occurred in the transition from an L-shape to a line. In the reference target evaluation of an overtaking target vehicle in Section 5.2.2, the importance of the secondary incidence angle was shown. As the angle becomes smaller, as shown in Figure 5.8c, both the proposed algorithm and the reference algorithm with GPR became more sensitive to oversegmentation, as seen in Figures 5.9a and 5.9d, until the angle becomes smaller than approximately 5°, after which segmentation accuracy is good.

6.2 Shape Extraction

The aim with shape extraction was to fit a representative geometric shape to each cluster of lidar detections. In the manual assessment some observations of incorrect shape classification were observed. During the lane change in scenario 1, there was one frame in which the proposed algorithm produced a correct segmentation of the detections from the vehicle, but fitted the wrong shape, as shown in Figure 5.1a.

This resulting shape indicated that there are certain data which even when clustered appropriately can not be correctly represented by the proposed algorithm. However, since no optimal tuning of the algorithm has been attempted, it is believed that this could be partially mitigated by finding cases such as this and modifying parameters to allow for better shape classification. Considering the results of scenario 3 in Table 5.2, it seems that the proposed algorithm in this situation is able to find the correct shape more reliably than the reference algorithm.

The two reference target scenarios described in 4.3.2 both gave slightly different results related to shape extraction accuracy. In the target overtaking scenario in Section 5.2.2, the shape fitting error of the proposed algorithm was in every case equal or better to that of the reference algorithm. In the ego overtaking scenario however, no clear difference between the two algorithms could be observed.

The main difference between the two scenarios is which angle the target vehicle is visible from, in other words the secondary incidence angle θ_2. In ego overtaking, θ_2 was very low throughout the scenario except for a small number of frames towards the end, where the ego vehicle overtook the target as seen in Figure 5.5b. The result of this was that the target vehicle for the most part appeared as a straight line for the lidar, hence mainly resulting in fitting of line shapes from both algorithms as supported by Figures 5.6c and 5.6f. We can draw the conclusion that both algorithms perform equally well, or nearly equally well when faced with this type of cluster.

In the target overtaking scenario, θ_2 had a greater spread as can be seen in Figure 5.8 which resulted in the lidar having vision of two sides of the target vehicle for a larger part of the scenario. The proposed algorithm here outperformed the reference algorithm for distances shorter than 50 meters, as observed in Figure 5.10. For both algorithms, all L-shapes observed were in this region, as can be seen in Figures 5.9c and 5.9f. It is also the region where both algorithms had the greatest errors, which could in part be due to the oversegmentation observed in 5.9b and 5.9e. The region with oversegmentation coincides with values of the secondary incidence angle θ_2 between 5° and 10°, which is typically the region when two sides of the target vehicle are detected, but the points on the secondary sides are still much sparser that the primary side. This region is also where the frequency of lines and L-shapes are roughly equal, as seen in Figures 5.9c and 5.9f. This corresponds to the transition area between L-shapes and lines which was also observed to contain more segmentation errors as discussed in Section 6.1. The conclusion is that the secondary incidence angle θ_2 is very important for the segmentation and the accuracy of the algorithm, and degrades performance in the region $5° < \theta_2 < 10°$. This is due to the relative sparsity of the points on the secondary side in this region, which leads to the oversegmentation which can be observed for both the proposed algorithm and the reference algorithm.

6.3 Execution Time

The reference algorithm was shown in Section 5.3 to have a considerably faster average execution time than the proposed algorithm. This can be explained by the costly merging and shape extraction subfunctions of the proposed algorithm, which together account for around 75% of the total execution time, as seen in Figure 5.11. It is also likely that the MATLAB implementations of the two algorithms differed in performance, and that the proposed algorithm could be implemented in a way that is considerably faster that the current implementation. Even if this is not the case, the average execution time of 287 milliseconds for the proposed algorithm could probably be reduced greatly by optimising the code and implementing it in a more efficient programming language such as C++.

The main factor that impacts the performance of the proposed algorithm is the number of clusters obtained in the clustering step, as illustrated in Figure 5.12b. This is correlated to the increase in execution time of the merging step f_{mrg}. A reason for this might be that single linkage clustering is performed on those clusters in f_{mrg}, as described in Section 4.2.4. Single linkage merging maintains a matrix of all the distances between the different clusters, as explained in 3.1, and therefore the memory and computational requirements grows as the the square of the number of elements. The MATLAB profiling showed that two functions in particular affected performance in the merging step: AllowMergeSingleLinkage and DistanceBetweenConvexHulls. Both of these are used within the single linkage merging, and has to be run more times if there are more clusters present. One way which the computation time could be reduced could be to use a different method of finding the distances between the different clusters. An idea is to save cluster distance information in the clustering step to give a preliminary guess of which clusters are likely to be in proximity, to avoid having to compute the full graph of all the clusters distances in each merging step. This could for example take the shape of arranging the different clusters in some sort of occupancy map, and subsequently limiting the search to neighbouring grids when computing distances between clusters. Another option is to attempt to optimise the implementation of the two functions AllowMergeSingleLinkage and DistanceBetweenConvexHulls in a different programming language, which is likely to improve performance.

The secondary shape extraction step f_{SE_2} also made up a large part of the total execution time. Profiling in MATLAB showed that the most significant time consumption was from L-shape fitting, concave hulls and line fitting constituting 43.5%, 21% and 20% respectively. The main issue with L-shape fitting using the SBRF algorithm detailed in 3.2.2 is that it always needs to fit a large number of boxes before it can find the optimal. If, for example an angular resolution of 1° is sought, 89 boxes will need to be fitted before the best one is found in the interval $[0°, 90°)$. The most obvious way to reduce the computation time of the L-shape fitting would therefore be to lower the angular resolution, but this would come at the cost of angular accuracy. Another approach, that could drastically improve the computation time without losing accuracy is to use a simulated annealing effect, where a larger initial angular step size is gradually decreased and re-estimated for the best region until the best global minimum is reached. Another idea is to reduce the amount clusters that are sent to the L-shape fitting function by performing additional pre-classification on the clusters, for example by removing clusters that are too large to represent a vehicle. The concave hull fitting consumed 21% of the time spent in the f_{SE_2} subfunction, which could be reduced by replacing the concave hulls with convex hulls which takes 98% less time to compute. This would however lead to less accurate results as a polygon could include empty areas that are spanned by the points. Alternatives to the implemented algorithm to generate concave hulls were not explored so it is also possible that another, more efficient algorithm, could reduce its computation time.

6.4 Future Work

In addition to the suggestions mentioned above, additional recommendations for future work are presented below.

Most pressingly, the designed algorithm needs tuning and further evaluation to assess its performance in general traffic situations. Furthermore, for the algorithm to be useful in an extended object tracker, the shape representation likely needs to be modified to suit the design of the tracker. For example, the extent of the shapes could be represented differently and with additional parameters that describe the orientation of the shape. The algorithm should then be evaluated together with the tracker to fully assess its performance. An appropriate uncertainty model for different shapes also need to be implemented in order for an extended object tracker to perform well. Further work could investigate if the proposed overlap factor could be used as a dynamic factor in such a model.

To facilitate easier data association in cases of oversegmentation for a tracker, it could be beneficial to retain information about the proximity of clusters that due to incompatible shapes were rejected for merging.

7
Conclusions

In this thesis, an algorithm is proposed with the purpose of reducing data complexity for use in a future object tracking algorithm. The proposed algorithm performs clustering and shape extraction on multi-layer lidar point cloud data, to be run in real-time in conjunction with a lidar mounted on the front bumper of an ego vehicle.

In manual assessments, the proposed algorithm has shown promising results in providing correct segmentation of dynamic objects in limited scenarios compared to a reference algorithm. The capability is especially noteworthy with respect to being able to provide adequate segmentation of vehicles that have ground detections in close proximity. The reference target evaluation showed an improvement in accuracy when the proposed algorithm was compared to the reference algorithm in the scenario of target overtaking, and equivalent performance in the second scenario of ego overtaking. The results highlighted that both the proposed algorithm and the reference algorithm performed considerably worse when an object was detected at a secondary incidence angle between 5° and 10°. This effect was also observed in the manual assessment in a scenario where a preceding vehicle performs a lane change. The conclusion is drawn that this is due to the fact that the density of data points on one of the sides of the target is much lower than on the other, leading to oversegmentation of the object and reduced stability in the fitting accuracy.

The execution time of the proposed algorithm is worse than the reference algorithm, but can be improved greatly by implementing the algorithm in a more high-performing programming language and optimising the subfunctions of cluster merging and L-shape fitting.

Bibliography

[1] BenjStaw. C3top svg. [Online; used under the creative commons CC0 1.0 universal public domain dedication license; accessed: 2017-06-01; url: https://commons.wikimedia.org/wiki/File:C3top_svg.svg].

[2] BenjStaw. C3side svg. [Online; used under the creative commons CC0 1.0 universal public domain dedication license; accessed: 2017-06-01; url: https://commons.wikimedia.org/wiki/File:C3side_svg.svg].

[3] Geovany Araujo Borges and Marie-José Aldon. Line extraction in 2D range images for mobile robotics. *Journal of Intelligent & Robotic Systems*, 40(3):267–297, 2004.

[4] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[5] E Rosen, E Jansson, and M Brundin. Implementation of a fast and efficient concave hull algorithm. Technical report, Uppsala University, Department of Information Technology, 2014.

[6] World Health Organization. Global status report on road safety, 2015. [Online; accessed: 2017-10-30; url: http://www.who.int/violence_injury_prevention/road_safety_status/2015/en/].

[7] National Highway Traffic Safety Administration. Automated vehicles for safety. [Online; accessed: 2017-10-30; url: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety].

[8] Velodyne LiDAR. HDL-32E. http://velodynelidar.com/docs/datasheet/97-0038_Rev%20K_%20HDL-32E_Datasheet_Web.pdf, 2017.

[9] Velodyne LiDAR. HDL-64E. http://velodynelidar.com/docs/datasheet/63-9194_Rev-F_HDL-64E_S3_Data%20Sheet_Web.pdf, 2017.

[10] Ocular Robotics. RobotEye RE05. http://www.ocularrobotics.com/wp/wp-content/uploads/2015/12/RobotEye-RE05-3D-LIDAR-Datasheet.pdf, 2014.

[11] CAR Magazine. How did Audi make the first car with level 3 autonomy? [Online; accessed: 2017-10-30; url: http://www.carmagazine.co.uk/car-news/tech/audi-a3-level-3-autonomy-how-did-they-get-it-to-market/].

[12] The New York Times. What Self-Driving Cars See. [Online; accessed: 2017-10-30; url: https://www.nytimes.com/2017/05/25/automobiles/wheels/lidar-self-driving-cars.html].

[13] Simo Särkkä. *Bayesian filtering and smoothing*, volume 3. Cambridge University Press, 2013.

[14] Stefan Wender, Kay Ch Fuerstenberg, and Klaus Dietmayer. Object tracking and classification for intersection scenarios using a multilayer laserscanner. In

Proceedings of the 11th World Congress on Intelligent Transportation Systems, 2004.

[15] Stefan Wender, Michael Schoenherr, Nico Kaempchen, and Klaus Dietmayer. Classification of laserscanner measurements at intersection scenarios with automatic parameter optimization. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 94–99. IEEE, 2005.

[16] Su-Yong An, Jeong-Gwan Kang, Lae-Kyoung Lee, and Se-Young Oh. Line segment-based indoor mapping with salient line feature extraction. *Advanced Robotics*, 26(5-6):437–460, 2012.

[17] Beomseong Kim, Baehoon Choi, Minkyun Yoo, Hyunju Kim, and Euntai Kim. Robust object segmentation using a multi-layer laser scanner. *Sensors*, 14(11):20400–20418, 2014.

[18] Abel Mendes, L Conde Bento, and Urbano Nunes. Multi-target detection and tracking with a laser scanner. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 796–801. IEEE, 2004.

[19] Danilo Caceres Hernandez, Alexander Filonenko, Dongwook Seo, and Kang-Hyun Jo. Lane marking recognition based on laser scanning. In *Industrial Electronics (ISIE), 2015 IEEE 24th International Symposium on*, pages 962–965. IEEE, 2015.

[20] Robert Lösch. Multitarget multisensor motion tracking of vehicles with vehicle based multilayer 2D laser range finders. Master's thesis, Technical University of Munich, 2017.

[21] Beomseong Kim, Baehoon Choi, Seongkeun Park, Hyunju Kim, and Euntai Kim. Pedestrian/vehicle detection using a 2.5-d multi-layer laser scanner. *IEEE Sensors Journal*, 16(2):400–408, 2016.

[22] Gwennael Gate and Fawzi Nashashibi. Using targets appearance to improve pedestrian classification with a laser scanner. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 571–576. IEEE, 2008.

[23] Samuel Gidel, Christophe Blanc, Thierry Chateau, Paul Checchin, and Laurent Trassoudaine. A method based on multilayer laserscanner to detect and track pedestrians in urban environment. In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 157–162. IEEE, 2009.

[24] Viet Nguyen, Stefan Gächter, Agostino Martinelli, Nicola Tomatis, and Roland Siegwart. A comparison of line extraction algorithms using 2D range data for indoor mobile robotics. *Autonomous Robots*, 23(2):97–111, 2007.

[25] Valentin Magnier, Dominique Gruyer, and Jerome Godelle. Automotive lidar objects detection and classification algorithm using the belief theory. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 746–751. IEEE, 2017.

[26] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

[27] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[28] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[29] Felipe Jiménez and José Eugenio Naranjo. Improving the obstacle detection and identification algorithms of a laserscanner-based collision avoidance system. *Transportation research part C: emerging technologies*, 19(4):658–672, 2011.

[30] George Nagy. State of the art in pattern recognition. *Proceedings of the IEEE*, 56(5):836–863, 1968.

[31] Mingqiang Yang, Kidiyo Kpalma, and Joseph Ronsin. A survey of shape feature extraction techniques, 2008.

[32] Pranab Kumar Sen. Estimates of the regression coefficient based on Kendall's tau. *Journal of the American Statistical Association*, 63(324):1379–1389, 1968.

[33] David Forsyth and Jean Ponce. *Computer vision: a modern approach.* Upper Saddle River, NJ; London: Prentice Hall, 2011.

[34] H Theil. A rank-invariant method of linear and polynomial regression analysis, Part 3. In *Proceedings of Koninalijke Nederlandse Akademie van Weinenschatpen A*, volume 53, pages 1397–1412, 1950.

[35] Xiao Zhang, Wenda Xu, Chiyu Dong, and John M. Dolan. Efficient L-shape fitting for vehicle detection using laser scanners. In *IEEE Intelligent Vehicles Symposium*, June 2017.

[36] Robert MacLachlan and Christoph Mertz. Tracking of moving objects from a moving vehicle using a scanning laser rangefinder. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pages 301–306. IEEE, 2006.

[37] Ronald L Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information processing letters*, 1(4):132–133, 1972.

[38] Alex M Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5):216–219, 1979.

[39] Wassim G Najm and John D Smith. Development of crash imminent test scenarios for integrated vehicle-based safety systems. Technical report, National Highway Traffic Safety Administration, 2007.