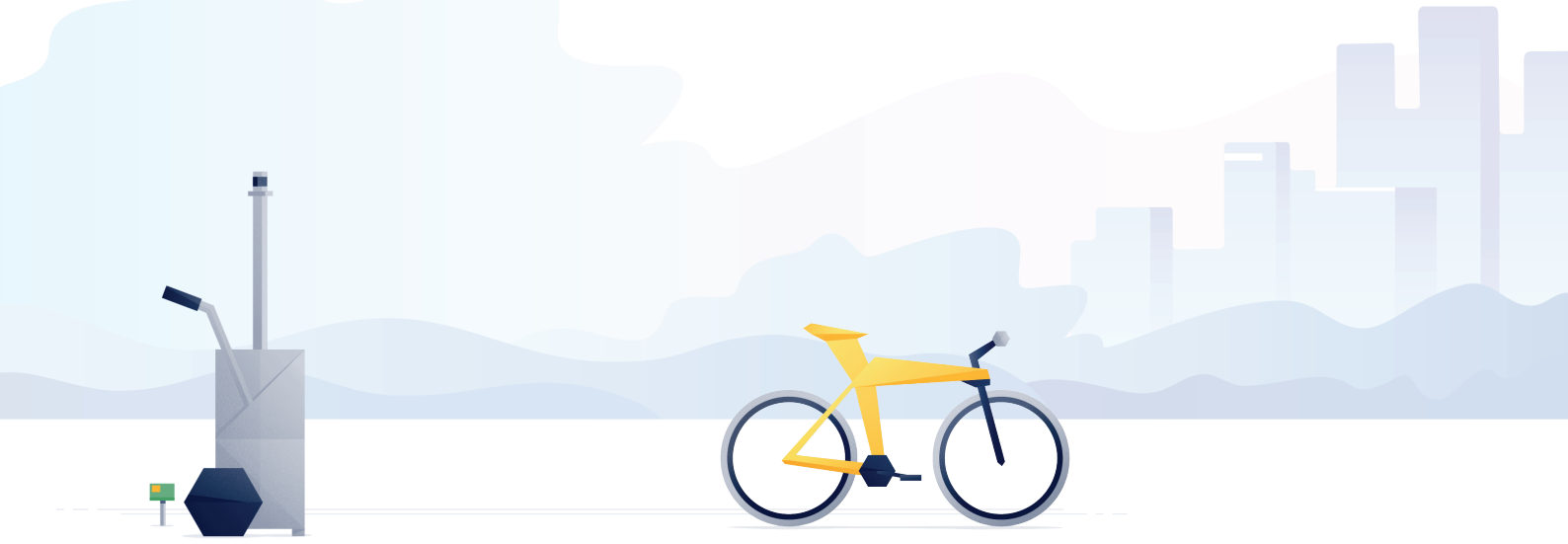
The Implementation of LiDAR-based Traffic Detection and Tracking

Sierd Meijer
M.Sc. Thesis
December, 2021

Human Media Interaction Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Supervisors
dr. G. Englebienne (UT)
dr. L.J. Spreeuwers (UT)
L. Licht M.Sc. (Mindhash)
D. van de Giessen M.Sc. (Mindhash)



Mindhash x UNIVERSITY OF TWENTE.

Summary

In order to gain insight in the behaviour of cyclists on intersections, which may lead to fewer accidents, Mindhash developed mobile LiDAR platforms for temporary observation of traffic situations. Light Detection And Ranging (LiDAR) is a laser-based distance measurement technique that is further developed to use multiple vertical channels and rotate rapidly to form three-dimensional point clouds of the sensor's surroundings. The overarching topics in this thesis are computer vision and data processing.

The main goal of this thesis is to implement an autonomous data processing pipeline that converts sequential series of LiDAR-generated point clouds into a list of tracked objects. During the implementation a new background removal method (BRM) is developed and the idea of using multiple dimensions estimation techniques is tested, leading to the following sub research questions:

- Does the use of a two-dimensional background model improve background removal performance over three-dimensional models with a rotational LiDAR sensor?
- Does applying different bounding box methods per traffic type improve the ability to estimate an object's orientation?

Based on related work a high-level data processing structure is designed and for each processing step multiple methods are compared. The data processing steps that are taken are:

1. Background identification & -removal
2. Point clustering
3. Object tracking
4. Classification
5. Orientation estimation

During the implementation a new BRM is developed called Peak Detection (PD). PD generates a two-dimensional model from a recording based on the combined measurements of an individual LiDAR angle. Whenever a significant number of distance readings are within a predefined distance from each other the measurement 'peak' is considered a stationary object and all measurements from that point and beyond are considered background. PD is tested against three existing methods to compare its background removal precision and recall, and processing speed.

Because all traffic types have vastly different dimensions and shapes, this thesis introduces the use of class-based bounding box methods for orientation estimation. Rotating calipers, Principal Component Analysis, and Minimum Error Rectangle are implemented and tested on pedestrians, two-wheelers, and vehicles. The estimated orientation, based on the alignment of the bounding box, is evaluated by calculating the alignment with an object's next location.

Each data processing step is validated in terms of expected outcome and usability for further testing. Both the BRMs and the orientation estimation are evaluated in-depth to provide statistically significant answers to the research questions. For the evaluation four 15 minute recordings are used, totalling more than 35.000 frames.

Results show that all implemented data processing steps function as expected and can reliably track objects. However, the clustering and tracking are prone to errors with closely grouped objects or occlusion. PD proves to be significantly faster than state of the art three-dimensional BRMs at an insignificant precision and recall loss. The model generation and background removal are 44% and 88% faster on average. For the orientation estimation it is determined that the differences between bounding box methods is significant for each object type. Principal Component Analysis and Minimum Error Rectangle perform especially well on two-wheelers and vehicles respectively with half of the measurements falling within a 9° alignment error.

The project goal is successfully reached and all requirements have been met. A fully autonomous data processing pipeline is designed and implemented and is able to produce a list of tracked objects. The data can also be visualised to show objects and their bounding box, class, velocity, and path. It is proven that the use of a two-dimensional background model does improve background identification and -removal by significantly increasing the processing speed at an insignificant precision and recall loss. It is also proven that using class-based bounding boxes improves the orientation estimation.

Contents

List of acronyms

LiDAR	Light Detection And Ranging
RADAR	Radio Detection And Ranging
CV	Computer Vision
UAV	Unmanned Aerial Vehicle
ROS	Robotic Operating System
IMU	Inertial Measurement Unit
FOV	Field Of View
ROI	Region Of Interest
ML	Machine Learning
PCA	Principal Component Analysis
MinErrorRect	Minimum Error Rectangle
PC	Principal Component
TSC	Threshold in Spherical Coordinates
PD	Peak Detection
RPP	Relevant Peak Percentage
RA	Raster-based Algorithm
3D-DSF	3D Density Statistic Filtering
TBC	Triangulation-Based Clustering
DBSCAN	Density-Based Spatial Clustering Applications with Noise

DTSCAN Delaunay Triangulation-based Spatial Clustering of Application with Noise

GNN Global Nearest Neighbour

ED Euclidean Distance

DO Detected Objects

ATO Actively Tracked Objects

BRM Background Removal Method

OP Object Points

BP Background Points

OPF Object Points Filtered

BPF Background Points Filtered

SDM Standard Deviation Multiplication

Chapter 1

Introduction

In this research a fully autonomous point cloud data processing pipeline is implemented that extracts tracked and classified objects from a stationary mounted LiDAR. Additionally, a background removal method is developed that performs on par with state of the art methods, but at a significantly higher processing speed. And finally, it is found that using different bounding box methods based on the object's class improves the estimated orientation, determined by the correspondence between the orientation and next location. This project is done in assignment of Mindhash, an innovation agency located in Hengelo, the Netherlands.

1.1 Context

The Netherlands counted 678 lethal traffic accidents in 2018 and it is estimated that there have been 21,700 accidents that resulted in severe injury [1], [2]. With an average cost of over 300,000 euros per accident with severe injury and 2.8 million euros per lethal accident this sums up to a total cost of approximately 8.4 billion euros. The total cost involved with all accidents in the Netherlands: between 15.8 and 18.6 billion euros [3].

A third of the lethal accidents and, based on LBZ registrations, almost two thirds of the severe injury accidents involve cyclists [2], [4]. The Institute for Road Safety Research (SWOV) in the Netherlands performs extensive research on cyclist safety, as it is one of the primary means of transportation. In a research on cycling accidents it is found that the most dangerous locations for cyclists are intersections. Between 52% and 65% of the lethal cycling accidents took place on intersections between 2000 and 2009 [5].

Increasing the safety and decreasing the number of accidents that lead to either severe injury or death can rapidly become a financially viable investment when

taking the reduction in costs into account. The World Health Organizations states:

"When safety is taken into consideration during the planning, design and operation of roads, substantial contributions can be made to reducing road traffic deaths and injuries." [6]

However, in order to be able to improve upon the safety of a given traffic situation, first it has to be known where the issues lie. Currently, little information is known about bicycle locations due to a lack of proper automated measurement methods and lack of records of non-severe injuries [5]. I.e. if an individual cyclist rides into the roadside he will likely not register the incident. Using automated measurement methods these less severe incidents can also be observed and analysed for a more in-depth understanding.

1.2 Project goal

The goal of this research is to implement a functional data processing pipeline to detect, classify, and track traffic from LiDAR-generated point clouds. The resulting data should enable data analysts and traffic experts to (automatically) analyse the data for relevant metrics or situations and behaviour of interest.

The high-level scope of this research is defined by the commissioner, which is Mindhash[1]. A mobile platform on which a LiDAR sensor is mounted is provided for the recording of data. The resulting data is to distinguish between pedestrians, two-wheelers, and vehicles and contain their direction and velocity. An additional requirement is that all traffic must be detected, even if the classification is incorrect. Furthermore, the resulting data must be visualised in a way that enables manual examination and validation. The data processing does not have to run real-time, however, with future development in mind the processing speed can be a decisive factor as a high processing time per frame is problematic for longer recordings.

During the design and development of the processing pipeline potential points of improvement have been discovered, of which two are researched in-depth. The first point is that most background identification and removal methods use three-dimensional models, while Light Detection And Ranging (LiDAR) output can be considered two-dimensional. If the two-dimensional model does not decrease performance significantly, the potential processing speed is large. Leading to RQ1 as stated below. The second point is that relevant literature often use a single bounding box method for all objects, while varying traffic types produce differently shaped

[1] https://www.mindhash.nl/

point clusters. Having a specialised bounding box method for each traffic type may improve orientation estimation, leading to RQ2 as stated below.

The aforementioned goal and scope are combined into the following research question: **How to autonomously detect, distinguish, and track traffic using a rotational LiDAR**. To which an answer will be researched based on the following sub research questions (RQ):

- **RQ1**: Does the use of a two-dimensional background model improve background removal performance over three-dimensional models with a rotational LiDAR sensor?
- **RQ2**: Does applying different bounding box methods per traffic type improve the ability to estimate an object's orientation?

1.3 Method

To answer the aforementioned research questions a literature study in combination with an iterative development cycle is used, of which the final iteration is presented in this document. The full system is validated using manually labelled data sets. For the sub questions more in-depth evaluation tests are designed and performed.

1.4 Report organisation

The context, goal, and research questions of this research are stated in the *Introduction*. Next, a *Background* chapter provides an explanation on what LiDAR is and why it is chosen. In *Related work* comparable research is discussed with respect to the main goal, leading to a processing pipeline design. For each step in the data processing multiple existing methods are compared. The *System* chapter states the final implementation of the system as it is used for answering the research questions. The *Method* & *Results* chapter describe the validation and evaluation tests and their results. Whether the results are as expected, provide an answer to the research questions, and show any anomalies is put in the *Discussion*. The *Conclusion* provides a concise answer to the achievement of the project goal and all research questions. Finally, recommendations are made on how to continue this research in the *Future work* chapter.

Chapter 2

Background

In this chapter the basic principle of LiDAR and its application in the used sensor is explained as it is fundamental to this research. Alternatives to LiDAR are discussed after.

2.1 Light Detection And Ranging

Light Detection And Ranging (LiDAR) is a remote sensing method that uses the light of a laser to measure ranges (variable distances) to reflective surfaces. Using a receiver the return of the reflection of the light is timed and thus the range is measured. Besides the duration, also the intensity of the returning light is often captured which is affected by the reflectivity of the measured surface. LiDAR has many applications, of which the automotive industry is currently the most popular due to the use of LiDAR for autonomous driving.

The most common implementation of LiDAR is in the so-called rotational LiDAR sensor. The rotational LiDAR sensor uses multiple vertically arranged lasers in a rotational motion in order to measure the surroundings of the sensor. This generates what is called a point cloud and represents the surroundings of the sensor as a three dimensional image consisting of measurement points as shown in figure 2.1. All measurements are stored in a spherical coordinate system with a radial distance r, polar angle θ, and azimuthal angle ϕ. For the conversion to Carthesian coordinates formula 2.1 can be used.

$$\begin{aligned} x &= r\,cos\phi\,sin\theta \\ y &= r\,sin\phi\,sin\theta \\ z &= r\,cos\theta \end{aligned} \tag{2.1}$$

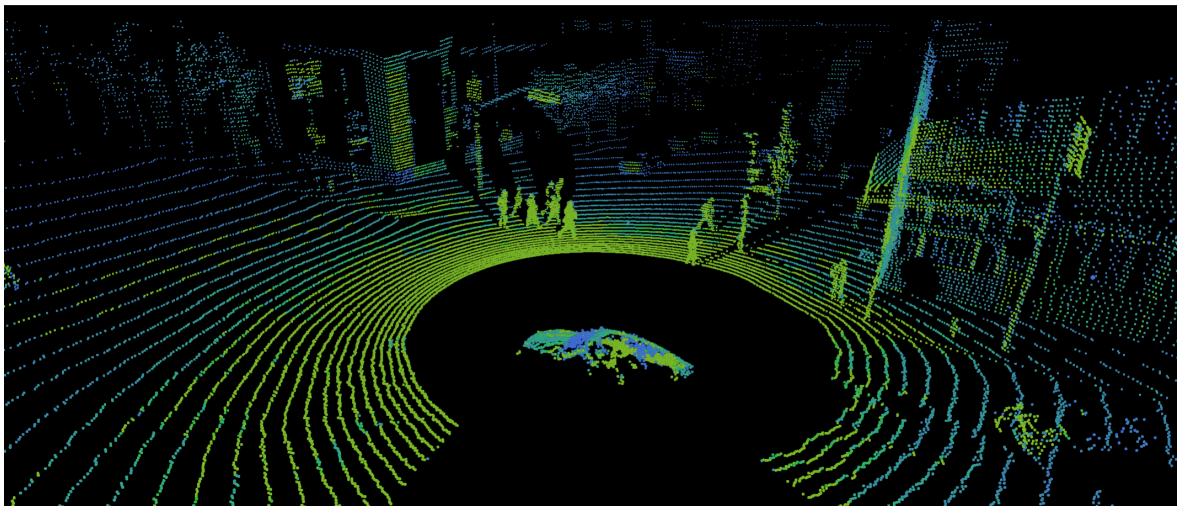


Figure 2.1: Example of a LiDAR generated point cloud

Depending on the specifications of the LiDAR sensor large amounts of data can be collected. Most rotational LiDARs range have a vertical resolution of 16, 32, 64, or 128 and a horizontal resolution of up to $\sim$4500. Rotational speeds (frame rate) are mostly between 5 and 20Hz, meaning that the most high-end sensors can record well above two million points per second. The high frame rates make them suitable for tracking objects and deriving measurements like velocity. More in-depth specifications for the used LiDAR sensors are given in section 4.2.

The LiDAR sensor generated image is sometimes called '2.5D' as objects are always partially visualised due to occlusion as shown in figure 2.2. It also prevents the observation of objects that are overlapping with respect to the sensor. The latter problem could be partially solved by placing the LiDAR sensor higher, however, this would also reduce the number of points representing the objects thus decreasing definition and range. Another option would be the use of multiple LiDAR sensors observing the same area from different angles and combining the point clouds. This concept is discussed further in section 9.

2.2 LiDAR alternatives

Multiple other technologies besides LiDAR are already being used or are actively researched for the observation of traffic. Even though the choice of LiDAR is already fixed, this section shows what potent technology it is with respect to existing alternatives. The most common type of traffic observation is determining the volume by counting objects. Table 2.1 shows a list of commonly used methods for counting vehicles either by detecting presence or sensing passing axles. However, most

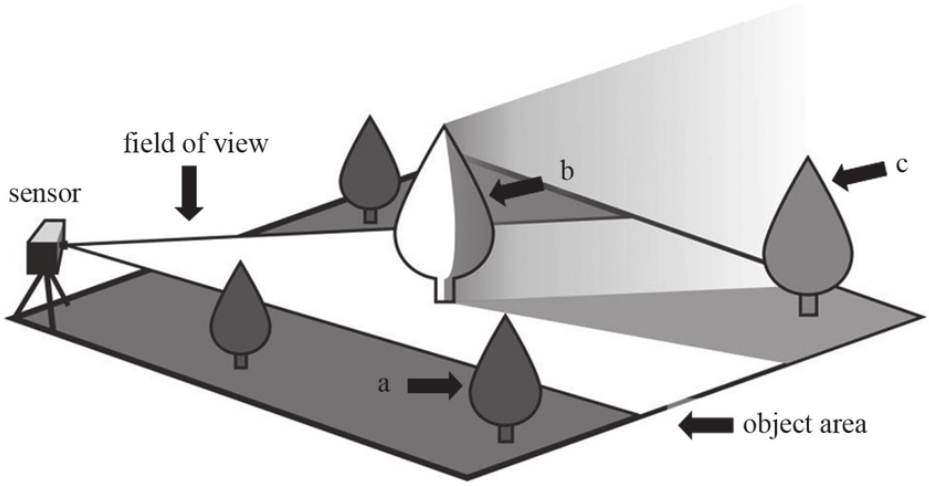


Figure 2.2: Objects within line of sight of the LiDAR sensor (b) prevent any measurement beyond the object (c), which is called occlusion. Sourced from [7].

of these technologies are solely designed for detecting objects in one location and cannot provide any other information besides count, classification, number of axles, or weight, depending on the sensor. As LiDAR is not only capable of counting traffic but also tracking it throughout 3D space, it is capable of providing a wider variety of measurements and even behavioural data. Sensing technologies for tracking traffic like video, Radio Detection And Ranging (RADAR), and LiDAR provide far more potential compared to counting only variants.

Methods for observing traffic behaviour on road segments are stationary cameras, cameras mounted on a Unmanned Aerial Vehicle (UAV) and RADAR. Research has shown that hovering over road segments using UAV-mounted cameras is an effective method for observing traffic [9]–[11]. Wang et al. [10] are capable of observing a road segment of up to 285m with 96.1% accuracy. However, UAVs and cameras have multiple limitations and/or problems [10], [11]:

- Cameras only provide 2-dimensional data
- Cameras are heavily light and weather dependent
- Wind can influence the position of the UAV and introduce a perspective change
- Untethered UAVs have limited battery capacity and therefore limited flight time

As for RADAR, Akita & Mita [12] & Zhao et al. [13] prove to be capable of identifying and tracking objects using millimeter-wave RADAR. While Zhao et al. [13] only have a usable range of approximately 5m, Akita & Mita [12] show a range of up to 40m. The latter also reach a 98.67% classification accuracy for distinction between car, pedestrian, bicycle, and parked car. Millimeter-wave RADAR has the same characteristic as LiDAR where resolution decreases over distance, thus distant objects

Table 2.1: List of available technologies for counting traffic [8]

Presence Sensing Technologies	**Axle Sensing Technologies**
Inductive loops	Infrared
Magnetic	Laser
Video detection system	Piezo-electric
Acoustic	Quartz sensor
Ultrasonic	Fiber optic
Microwave RADAR	Capacitance mats
Laser RADAR	Bending plates
Passive infrared	Load cells
	Inductive Signatures
	Contact switch closures (e.g., road tubes)

easily blob together making it harder to distinguish, identify, and track individual objects.

Extensive research between LiDAR and RADAR is done by Ryde & Hillier [14] and based on their conclusions LiDAR is more applicable for precise recognition and tracking of objects. They conclude that RADAR is robust to visually obstructing weather conditions like mist and rain, but has low precision measurements which are relatively spare. While LiDAR is more easily interpreted and provides high precision and accuracy, but at the cost of being more affected by adverse weather conditions. This makes LiDAR the better option for dimension estimation and classification.

Chapter 3

Related Work

The first step to answer the main research question is to look at existing literature with overlapping interest. If literature does already exist on the detection, tracking, and classification of objects the question is whether it is applicable for this research' specific case, and if so, whether it is available. If no directly comparable literature is available an alternative source of information is two-dimensional image based computer vision. Initially, literature overlapping with the main research question is looked for. After that, literature overlapping with individual sub questions is researched.

3.1 LiDAR-based traffic analysis

Using a stationary LiDAR to observe traffic is a recent development compared to image-based traffic observation, however, literature does exist on this topic. Tarko et al. [15], Wu [16], Zhao et al. [17], and Cui et al. [18] all use a roadside 360° LiDARs for the detection, tracking, and classification of traffic. Another closely related research is the detection of crossing deer using LiDAR [19]. Noteworthy is that they all use a combination of background removal, point clustering, object tracking, and object classification. And, all except Zhao et al. [17] also use ground- or lane identification.

Especially the combination of background removal and point clustering are remarkable as most recent developments in object recognition and classification in point clouds mostly aim at using some form of Machine Learning (ML) [20]–[24]. These ML implementations are trained to detect and classify objects directly from a point cloud collected from a moving platform. However, when looking at aforementioned research using stationary LiDAR platforms a more traditional Computer Vision (CV) approach is used starting with background removal (now referred to as *traditional CV methods*) [15]–[18]. An often used explanation for a traditional

CV approach is the ability to determine the background given the stationary LiDAR, however, this does not explain why ML is not used.

One potential explanation why ML is not used for object detection with stationary LiDARs is given by Cui et al. [18], mentioning that LiDARs mounted on a mobile platform require dense point clusters to detect objects. This results in a lower usable range due to the increasing sparsity of LiDAR based point clouds. Using background removal and point clustering would increase the range and/or enable the use of more cost-effective LiDARs with a lower resolution. However, neither of these claims are addressed by Cui et al. [18].

An advantage of traditional CV methods over a ML approach is that unclassified objects can still be detected. This advantage is twofold as distant, low resolution objects are not immediately filtered out and unknown objects are detected even though they might be wrongly classified. However, the inverse is also true as unfiltered background can result in falsely detected objects. As ML approaches are trained to recognise expected patterns, the low resolution of distant objects can decrease the confidence of a recognised pattern significantly. Also, patterns that are relevant but never seen before can be overlooked. Assuming a flawless background removal method, one can be positive that the remaining point clusters are indeed relevant objects.

That a cost-effective LiDAR is sufficient for the detection and tracking of objects using tradition CV methods is already confirmed, however, whether this is also true for ML methods is more difficult to prove. Rotational LiDARs often come with a vertical resolution of 16, 32, 64, or 128 channels, with a exponential increase in price for each doubling in resolution. Meaning that the lowest resolution LiDARs will be the most cost-effective if proven to be sufficient. Wu [16], Zhao et al. [17], and Cui et al. [18] already show that a 16-channel LiDAR is sufficient using traditional CV methods by detecting and tracking objects up to 30m.

Many ML methods for object recognition are done using a 64-channel as they are evaluated using the KITTI dataset [25]. As the dataset contains point cloud data from a moving vehicle traditional CV methods cannot be applied, making it difficult to compare traditional CV and ML methods. As the goal of this research is to implement functional object detection, tracking, and classification using a low resolution LiDAR, the reasoning why ML is not used is not further explored.

Why ML methods are not used for object recognition from point clouds cannot be clarified from literature, however, there are advantages with using traditional CV methods. As recognising objects from a point cloud is no trivial task, the implementation of a traditional CV method is less resource and time intensive since it is already proven to work. Also, traditional methods do not require a large amount of

training data and can almost directly be used. Traditional CV methods can also be used to automatically gather training data for a future ML implementation instead of manually classifying it. In order to develop a complete processing pipeline from point clouds to tracked objects within the available time frame the traditional CV approach is used.

3.2 Object detection

This section provides a starting point to solve the problem of distinguishing the traffic from the environment. Arguably, this is the most important and difficult step due to the requirement of the processing being fully autonomous. Based on existing literature the choice is made to first remove the background and then, from the remaining points, determine what objects are present in the point cloud instead of identifying objects using expected patterns. After the background is removed each remaining point needs to be associated to its corresponding object which is done using a clustering algorithm. The background identification & removal and clustering are separately discussed.

3.2.1 Background identification & removal

All aforementioned literature that use a stationary LiDAR ([15]–[19]) agree that for the detection of objects first the background has to be identified, then removed, and finally the remaining points clustered into objects. Other methods do exist, Kidono et al. [26] and Liu et al. [27] both determine the ground plane first after which Kidono et al. [26] generate an occupancy map for clustering and Liu et al. [27] apply template matching. Template matching is not a possibility as one of the requirements is that all traffic must be recognised, even if the classification is wrong. The occupancy map has a similar problem where the characteristics of objects need to be known before they can be detected. Thus, background identification and removal is used.

A perfect background identification and removal method keeps all points belonging to traffic and removes all other points belonging to e.g. roads, houses, and trees without any manual input. It is often a trade-off between removing too little, i.e. not removing all the noise, and removing too much, i.e. removing points from objects of interest. The following background identification methods are considered based on their applicability in this research and the requirements described in section 1.2.

One of the requirements is that the background identification is completely autonomous. Both 'azimuth-height background filtering' and 'background filtering based

on point association', by Zhao et al. [17] and Zhang et al. [28] respectively, require the input of an empty frame which is manually selected. As scanning through the frames can be a time consuming process, these methods are not considered.

Another method is 'slice-based projection filtering' by Lin et al. [29] which not only detects dynamic objects, but also points of interest like curbs and trees. However, as the recorded data is stored these features can always be extracted when needed and are currently not of interest, making the more conservative method undesirable.

3.2.1.1 TSC

The implementation of Tarko et al. [15] uses two background identification methods, one based on a manually selected region of interest and one that is autonomous. The autonomous method is called Threshold in Spherical Coordinates (TSC) and generates a background model by determining a maximum distance threshold in which a measurement is considered dynamic for each LiDAR angle. The exact performance is not mentioned, however, it is not expected to perform well as the maximum distance threshold calculation is not robust when traffic is passing. What is interesting is that the resulting background model is two-dimensional and in spherical coordinates, therefore potentially fast in terms of processing speed. By improving the method for determining the maximum distance threshold a sufficiently performing and fast background removal method could be uncovered.

3.2.1.2 3D-DSF & RA

The last two methods are 3D Density Statistic Filtering (3D-DSF) and Raster-based Algorithm (RA) and are both fully autonomous background identification methods. RA is a continuation of 3D-DSF and uses the same idea of generating a background model by dividing the 3D space into equally sized cubes which are labelled either background or foreground [30], [31]. 3D-DSF has already been proven effective, however, noteworthy is that the method itself and the implementations are done by overlapping authors [16], [18], [19]. Even though RA shows to be more effective than 3D-DSF by their own validation, both methods are considered as 3D-DSF is already proven and RA has the potential to be even more effective.

3.2.1.3 Comparison

Compared to TSC, 3D-DSF & RA are expected to have higher background removal performance but with significantly slower processing speed. For 3D-DSF & RA the point cloud needs to be converted to Cartesian coordinates first and then checked

with the 3D matrix. 3D-DSF & RA both claim to be usable in real-time as the average background removal duration for one frame is 100ms on average. However, this leaves no room for any additional processing as the frame rate of the used LiDAR sensor is 10Hz. If the performance of (a variation of) TSC matches that of 3D-DSF or RA it expected that TSC is the better choice due to faster processing times.

As the performance of the background identification & removal is a vital part of the data processing all three methods, TSC, 3D-DSF & RA, are all implemented and evaluated. The main point of evaluation is the performance in terms of object point retention and background point removal. However, a second subject of interest is the expected trade-off between the processing speed and performance of 2D- and 3D-based background models. The best performing method is chosen after evaluation, given that the increased performance does not come at a detrimental processing speed decrease.

3.2.2 Point clustering

After the background is removed from a point cloud all that remains are points that are assumed to represent objects of interest and need to be clustered per object. Multiple methods are available: Tarko et al. [15] use Triangulation-Based Clustering (TBC) with Delaunay triangulation, Kidono et al. [26] cluster all points within a predefined range (threshold clustering), however, the majority uses Density-Based Spatial Clustering Applications with Noise (DBSCAN) [16], [17], [19], [28]. A combination between Delaunay triangulation and DBSCAN is presented by Kim & Cho [32] called Delaunay Triangulation-based Spatial Clustering of Application with Noise (DTSCAN). Both triangulation-based clustering and DBSCAN are considered an improvement over threshold clustering, therefore threshold clustering is not considered.

3.2.2.1 TBC & DBSCAN

The fundamental difference between TBC and DBSCAN is that they are reduction- and expansion-based respectively. TBC creates connections between any point and its neighbours and prunes connections based on two criteria. The first is a predefined maximum connection length and the second is a maximum connection length based on the average length within the point's cluster. Points that remain connected are considered clusters.

DBSCAN starts by selecting a point and determining its neighbours within a distance measure ϵ. If the number of neighbours (including itself) exceeds a minimum

number of points (*MinPts*) than the selected point is considered a 'core point' and its neighbours' neighbours are also determined. Neighbours that exceed *MinPts* also become core points and repeat the process, while those with too few neighbours are marked as 'border points' and are not expanded. A cluster is completed whenever no new core points are found.

While DBSCAN has an inherent noise rejection characteristic, both DBSCAN and TBC are extended to mark clusters that do not exceed a minimum number of points to be considered an object as noise. This minimum number of points is not the same as *MinPts* for DBSCAN, but applies to the finished cluster. Cui et al. [18] also note that due to the point cloud's increasing sparsity *MinPts* has to decrease as a function of distance. Zhao et al. [17] come to the same conclusion but, opt for a solution that divides the range into three rings with corresponding values to prevent needing to calculate *MinPts* for each point or cluster. As the distance between points is dependant on the specifications of the LiDAR sensor, the minimum point function has to be calibrated.

The increasing point sparsity over distance also affects the performance of a chosen value for ϵ in DBSCAN. Again, both Cui et al. [18] and Zhao et al. [17] solve this by making the value distance dependant. On this part TBC has an advantage as it calculates the average point distance within each cluster which has a similar effect as DBSCAN's ϵ. The disadvantage is that TBC provides less control and thus potentially falsely detects clusters. Whether there is a significant difference in performance between TBC and DBSCAN has to be tested.

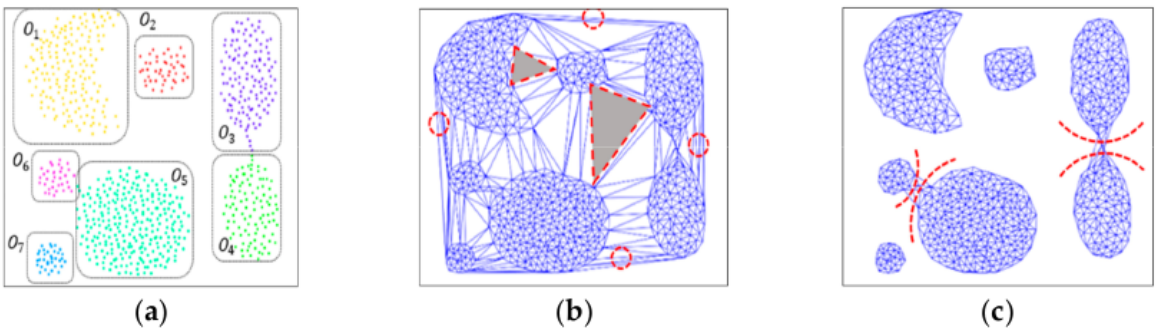


Figure 3.1: Simulated spatial data and its clustering. (a) original data; (b) Delaunay triangulation; (c) Triangulation after removing cluster-bridging connections. Sourced from [32].

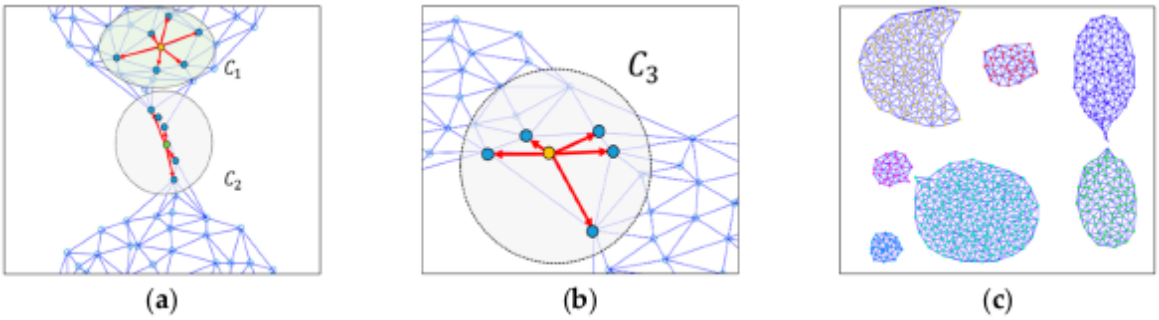


Figure 3.2: Touching problem in simulated data solved by DTSCAN. (a) and (b) are enlarged illustrations of figure 3.1c; (c) clustering result of DTSCAN. Sourced from [32].

3.2.2.2 DTSCAN

The third clustering algorithm is a combination of TBC and DBSCAN, called DTSCAN, developed by Kim & Cho [32]. By combining the methods DTSCAN has an improved ability to separate adjacent clusters that make contact (figure 3.1 & 3.2). The method starts of similar as TBC as all point connections are calculated using Delaunay triangulation (figure 3.1b). Connections between clusters are removed based on the size of the triangle that forms between edge points of clusters, which are relatively wide or long (red triangle in figure 3.1b). The result is shown in figure 3.1c. Then, a variant of DBSCAN is applied on the connected points using *MinPts*, but not use ϵ. Select a point, if the number of connected neighbours exceeds *MinPts* it is considered a core point. Neighbouring nodes are added to the cluster and those that exceed *MinPts* are also core points. This process is repeated as with DBSCAN. The result is shown in figure 3.2c.

The ability to separate adjacent clusters that are touching is based on the fact that bridging points between clusters are likely to have fewer connection nodes than points within a cluster. Figure 3.2a and b show enlarged illustrations of the indicated areas in figure 3.1c. The nodes at the center of both bridging points only have five connected nodes (C_2 & C_3 in figure 3.2) surrounding them as opposed to more with most points within the cluster (C_1). Thus, by tuning *MinPts* closely located objects like a group of pedestrians can be separated using DTSCAN.

3.2.2.3 Comparison

Based on the results of Kim & Cho [32] it is expected that DTSCAN will outperform both TBC and DBSCAN. However, as with background identification & removal, also the processing speed is a relevant metric to take into account. As both TBC and

DBSCAN are combined into DTSCAN all three methods can be evaluated based on clustering performance, noise reduction, and processing speed. As most related literature uses DBSCAN it is expected to be an improvement over TBC when applied to LiDAR-based point clouds. All three methods will be implemented for further research.

3.3 Object orientation & tracking

To find out whether bounding box methods align better with certain object classes multiple methods are required. Beside the bounding box methods also a tracking method is required, which will later be used to evaluate the orientation estimation. After an object is detected in a point cloud the next step is to associate said object with itself in the next frame, also known as object tracking. Having more information on an object's current state provides the ability to more accurately predict its next state and thus improve the tracking capability. Knowing the orientation of an object is valuable as it provides the option to determine the rotational velocity alongside the translational velocity of a tracked object. The orientation of an object can be determined in two ways: based on the rotational angle in the trajectory or by estimating the orientation based on the observed point cluster.

Both orientation estimation options are on opposite sides of a balance between real-time information and accuracy. Using the rotational angle of the trajectory to determine the orientation of an object is a reliable method as the estimation is based on the known locations. This also means that the orientation can only estimated when the next location is known which means that the estimation always lags behind. This does not have to be a significant problem with objects that do not change direction quickly or have expected moving patters, however, as this research includes pedestrians and cyclists this may not be the case. By determining the orientation using the observed point cluster of an object a real-time estimation is made. This has as disadvantage that the estimation is less reliable than using the rotational angle because only part of the object is visible. However, if a sufficient estimation can be made, the expectation is that the trajectory of an object can be predicted more reliably.

To evaluate whether the bounding box-based orientation estimation aligns with the actual orientation, and thus answer RQ2, the alignment between the estimation and the trajectory is evaluated. This alignment is evaluated on a per object class basis. Therefore, this section compares multiple bounding box-, tracking, and classification methods.

3.3.1 Orientation estimation

To estimate the orientation of a detected object a bounding box is aligned with its point cluster. As mentioned in section 2.1, a LiDAR sensor only observes part of the outer shell of an object making it more difficult to enclose the cluster in a bounding box that represents the orientation of the object. Pedestrians are the most difficult due to the organic and dynamic shape, while two-wheelers and vehicles have a more consistent and geometric shape. To determine the orientation three bounding box methods are presented: rotating calipers, Principal Component Analysis (PCA), and Minimum Error Rectangle (MinErrorRect).

An initial test is done to form a hypothesis on which method is expected to result in the closest approximation of the actual orientation. The methods are visually evaluated when applied to pedestrians, bicycles, and cars. All methods are applied in two dimensions, which means that the points are projected on a plane parallel to the z-axis (or ground plane). First, the three methods are elaborated upon, and after, a comparison is shown.

Rotating calipers The Rotating calipers method is an algorithm that can be used to determine the width or diameter of a set of points. It is called the rotating calipers method as it resembles rotating a set of calipers around a polygon. The method is first explained by Shamos [33]. Using the method a *minimum bounding box* or *minimum-area enclosing rectangle* around the point cluster is computed. When a full outer-shell point cluster exists of an object the minimum bounding box is likely to align well with the actual orientation, however, as mentioned in section 2.1 LiDAR sensors only observe part of the outer shell. This can lead to a bounding box that lays down diagonally with respect to the actual orientation as it forms the rectangle with the smallest area.

3.3.1.1 PCA

PCA is a method for the reduction of dimensionality of data, however, can also be used to form bounding boxes around point clusters. Sidharth et al. [34] provide an in-depth explanation of PCA and state: *"The central idea of PCA is to reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much as possible of the variation present in the data set."* As the point clusters are already two dimensional due to projection, it is not necessary to reduce the dimensionality. But, as both vehicles and two-wheelers are often longer than they wide, the first Principal Component (PC) is likely to correspond with the direction. A bounding box is created by fitting a rectangle at the end of the first and

second PC. However, the same problem applies as with Rotating calipers, the outer shell is not fully visible and thus the first PC may not correspond with the actual direction.

3.3.1.2 MinErrorRect

The Minimum Error Rectangle (MinErrorRect) method by Tarko et al. [15] is an alteration on the minimum area bounding box (Rotating calipers). Tarko et al. [15] recognised that the minimum area bounding box is often not ideal and presented an improved method by using a minimum error bounding box with respect to one of the edges of the object. This ensures that the bounding box is parallel to at least one of the edges of the object. The Euclidean distance between all the points and the specified edge are summed as the cost function and that cost function is minimised. Making the bounding box parallel to one of the edges of the point clusters solves the problem of only observing part of the outer shell of an object. It does not work for organic shapes as they do not have straight edges, however, it is expected to perform well on vehicles.

3.3.1.3 Comparison

All three methods are implemented and applied to a set of point clusters representing pedestrians, bicycles, and cars and the alignment of the bounding boxes is visually evaluated. An example of the result of each method is shown in figure 3.3. Rotating calipers performed the worst as almost all bounding boxes are misaligned. The problem with having only part of the outer shell in combination with the smallest area rectangle is visible in figure 3.3 where Rotating calipers is applied to a car. The same is true when Rotating calipers is applied to a bicycle or pedestrian.

PCA and MinErrorRect both provide a solution for different object classes. As expected with MinErrorRect, it performs well on rectangular shaped objects like vehicles as the cost function is minimal when aligned with a straight edge. Due to the rhombus shape of a cyclist's point cluster MinErrorRect does not align with the direction, but PCA does. The relatively long and small point cluster of a cyclist makes it so that the first PC aligns with the length, and thus the direction.

For pedestrians neither of the three methods performs particularly well as they are oval-shaped and have dynamic dimensions when walking. Moving arms and legs drastically change a pedestrian's profile when viewed from above. Both Rotating Calipers and MinErrorRect resulted in seemingly random bounding boxes due to the constant changing of shape. PCA is somewhat consistent as most pedestrians

are wider than they are long when standing still and the first PC corresponds with the width.

For the implementation both PCA and MinErrorRect are used with PCA for pedestrians and bicycles and MinErrorRect for vehicles. As the choice is made based on empirical evidence, further testing is required to determine whether this is the optimal solution. If the evaluation shows that the observed orientation of objects does correspond with the future location, it also means that the chosen bounding box method performs as expected.

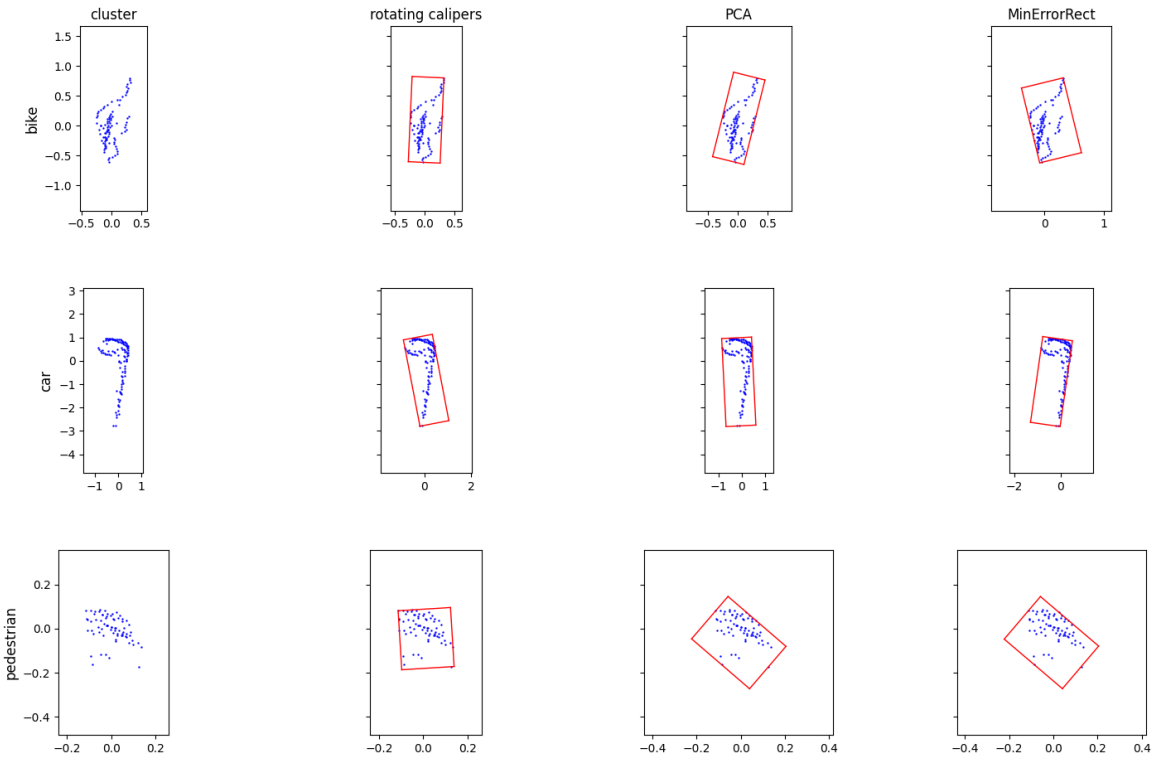


Figure 3.3: Comparison between Rotating calipers, PCA, and MinErrorRect for drawing bounding boxes around point clusters applied on a bicycle, car, and pedestrian. All objects are facing upwards and angled slightly to the right.

3.3.2 Tracking

The tracking of objects over time is a well research topic in computer vision and, as it is mostly independent of the used observation technology, many methods are available. Within LiDAR-based research multiple options have already been implemented, with the most common method being Global Nearest Neighbour (GNN) [16]–[19]. GNN does not require any input beside the locations of objects in the cur-

rent and previous frame. This makes it relatively easy to implement, but susceptible to incorrect track association. Besides GNN another method using a Kalman filter is considered, which is expected to perform better than GNN at the cost of more implementation time. Further options will only be considered if neither option proves to be sufficient.

3.3.2.1 GNN

Whenever there is a small number of objects within sight it is highly probable that the object in frame *N* corresponds to the closest object in frame *N+1*. However, when more objects are involved or closely located the association becomes less probably and path prediction can provide a solution. There are examples where path prediction is not necessary and a relatively basic tracking algorithm like GNN is considered sufficient.

Many aforementioned researches [16]–[18] refer to image 3.4 and state that the travel distance of a vehicle between frames at 10Hz is significantly smaller than the distance between following vehicles, thus GNN will suffice. The expectation is that this claim does not hold when taking groups of cyclists and pedestrians into account. Chen et al. [19] already note that when deer are moving at high speed and close to others that GNN could not always track them correctly. Especially in the Netherlands it is common to have groups of recreational- or student cyclists move in a tightly packed formation. Therefore, the more advanced Kalman filter tracking is considered that predicts the location of an object in the next frame.

3.3.2.2 Kalman filter for object tracking

Another method that is commonly used to improve object tracking is the Kalman filter, which can be considered an extension to GNN. In short, the Kalman filter models the trajectory of an object as a linear dynamical model and predicts the next location based on previous locations. The observed object that is closest to the predicted location is associated and the Kalman filter updates its state using both the observed and predicted location according to an observation- and measurement inaccuracy. Tarko et al. [15] applied this method to LiDAR-based data with success.

A drawback of the Kalman filter's linear dynamical model is that it does not consider rotational velocity. The predicted location always lays in line with the last observed direction, making it difficult to track objects that are turning. It is possible that the lack of vertical velocity is compensated by the fact that at 10Hz the deviation from a linear model is insignificant. The linearity is addressed by further developed methods called Extended- and Unscented Kalman Filter. However, this could also

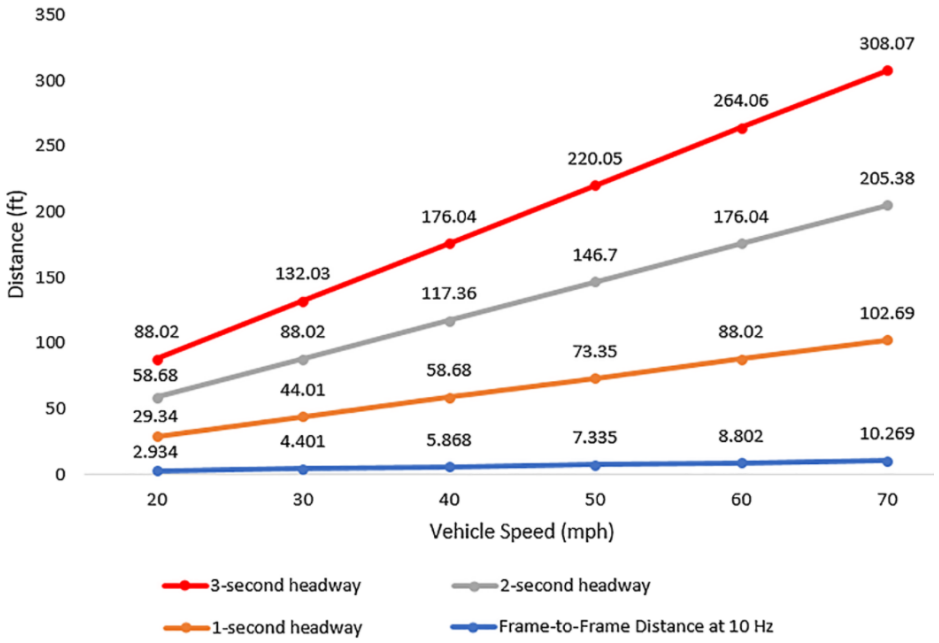


Figure 3.4: Comparison of frame-to-frame travel distance and distances between different vehicles in the same lane. Sourced from [17].

be addressed by using the point cluster-based orientation estimation if proven to be relevant.

3.3.3 Object classification

Because different orientation estimation methods are used depending on the object class, an object classification is required. As it is not the goal of this research to improve the classification of LiDAR data, a simple yet effective method is sought. All objects are to be divided into pedestrians, two-wheelers, and vehicles. Implementing a functional classifier consists of two parts: finding differentiating metrics between classes and choosing a classification method.

3.3.3.1 Differentiating features

Finding relevant features that can be used to differentiate between the object classes is vital for the performance of the classifier. An advanced classification method might be able to find a correlation where human interpretation cannot, but if a chosen feature is not consistent within one class and between classes than no classification method will be effective. E.g. trying to differentiate between cars and bikes based on color. First, a list is made of features used by related literature on the classification of LiDAR-based objects [15], [17]–[19], [35].

- Dimensions
- Velocity
- Acceleration
- Number of points in cluster
- Primary direction of the point distribution
- Distance to the LiDAR sensor
- Intensity extremes, average, and variance
- 3D covariance of clusters
- Convex hull features
- Positioning on road

Comparing classification methods shows that the choice between frame-by-frame object classification and post-tracking classification has significant influence on the choice of features. Whenever an object is far away from the LiDAR sensor it is difficult to make an accurate estimation of the dimensions based on a single frame. It is significantly easier when multiple frames of the same object are available at different ranges due to tracking. In case frame-by-frame classification is used, the previous frames, if available, can also be used making the classification more reliable over time. Having the possibility to classify after an object is tracked is beneficial as it provides overall more data and more reliable data than a single frame and requires less processing.

Because of the differences in dimensions between pedestrians, two-wheelers, and vehicles it is expected that a combination of length, width, and height are valuable features to create a classifier. Table 3.1 shows the average dimensions of pedestrians, two-wheelers, and vehicles, and their maximum allowed dimensions if applicable. Tarko et al. [15] estimate the dimensions using all observed point clusters of an object and report that on average the estimated dimensions were 38cm and 15cm shorter for the length and width respectively. Using a combination of dimensions, velocity, and positioning on the road they achieved a 98% classification accuracy between pedestrians, bicycles, and vehicles.

Pedestrians differ from the other classes as they are significantly taller than long or wide. Zhao et al. [17] show that the distribution of the clustered points for pedestrians lay in the direction of the z-axis, while that of vehicles lays in the x- and y-axis. The average dimensions of pedestrians in table 3.1 are when standing straight and stationary and will increase, especially in length, for a moving pedestrian. However, if the height of an object is larger than the length of an object it is likely a pedestrian. Two-wheelers and vehicles are more similar in shape, but do differ in absolute size. Both bicycles and motorcycles are shorter and smaller than vehicles and approximately as tall when including the driver.

Table 3.1: Average and maximum dimensions of common traffic types. The maximum dimensions are based on Dutch law. Note: the average dimensions are not scientifically proven and only act as an indication. [36]–[42]

	Dimensions (m)	Length		Width		Height	
		Average	Max	Average	Max	Average	Max
Class	**Pedestrian***	0.3	-	0.5	-	1.8	-
	Bicycle	1.9	-	0.5	0.75	0.9**	-
	Motorcycle	2.1	4	0.7	2	1.2**	2.5
	Car	4.7	12	1.7	2.55	1.5	4
	Heavy vehicles	-***	18.75	-***	2.55	-***	4

* Men standing with arms beside body

** Excluding driver

*** Varies heavily with vehicle type

As there is a significant difference in shape and dimensions between pedestrians, two-wheelers, and vehicles it is expected that no other features are necessary for classification. One problematic situation would be whenever objects keep a relatively large distance to the LiDAR sensor and only few points represent an object. However, as the observations are done on intersections traffic is expected to cross it and thus move close to the LiDAR sensor if positioned correctly. Using only dimensions does mean that a robust dimension estimation method is required.

3.3.3.2 Classification method

For the classification of objects a manually assigned decision tree is used. Due to the time limitations of this research a basic and easily implemented classification method is chosen. As described in the previous section the differentiation between pedestrians, two-wheelers, and vehicles is based on two decisions. The first separation of objects is done based on the ratio between the height and length of objects, all objects that are taller than they are long are considered pedestrians.

The remaining objects are divided into two-wheelers and vehicles based on the width. Even though the average dimensions of two-wheelers and vehicles differ enough to differentiate them, the maximum allowed dimensions of motorcycles do overlap with the expected dimensions of small vehicles. The maximum allowed width of a motorcycle does include a sidecar, thus, most motorcycles will be significantly smaller. Almost all small city cars have a width of at least 1.6m[1], making the difference between vehicles large enough to be used as a differentiating feature.

[1] https://www.automobiledimension.com/city-cars.php

3.3.4 Dimension estimation

To classify the objects based on their dimensions a dimension estimation method is required. As only a part of the outer shell of a LiDAR-observed object is visible per frame the dimension estimation is expected to improve when based on multiple observations at varying angles. This is only true for the length and width of an object as the height is never occluded from a sideways perspective. The precision of the estimation is mostly relevant to differentiate between two-wheelers and vehicles. Three methods are presented: the first uses the maximum observed dimensions, the second excludes a top percentage of the observations, and the last method aligns all observations and encloses a percentage.

3.3.4.1 Maximum observed dimensions

The simplest method to determine the dimensions of an object is to take the maximum observed values of a tracked object. The method is based on the fact that a LiDAR sensor can never observe more than the complete object, but it can observe only a part. Before the maximum value can be determined it must be known which measurements correspond to the length or the width of an object. This can be done by calculating the angle between the orientation of the length and width, and the trajectory. The side that aligns with the trajectory is the length of the object. Combining the maximum observed length, width, and height determine the dimensions of the object.

3.3.4.2 N^{th} percentile observations

A slight alteration to the maximum observed dimensions is to take a high percentile of the measurements, excluding the largest observations. This prevents incorrect measurements from dictating the estimated dimensions of an object. E.g. if multiple cyclists are grouped in a frame due to close proximity, the estimated dimensions should not change to those of a double-wide bicycle. However, this is likely to increase the underestimation of the dimensions. The robustness to incorrect measurements and the degree of underestimation is a trade-off that can be tuned. An alternative is to exclude any calculated dimensions that deviate significantly, this adds an extra processing step but it is expected to result in the most accurate dimensions.

3.3.4.3 Enclosing percentage

The last method is presented by Tarko et al. [15] alongside the MinErrorRect method and overlays all measurements aligned by one edge. When estimating the dimension using the MinErrorRect method, the angle with which the object should be rotated in order to align the leading edge parallel to the x-axis is known. Then, the clusters representing the object in each frame are laid on top of each other aligned by the leading edge. To estimate the dimensions a bounding box is drawn with one corner aligned to the combined cluster and the size is determined by covering a predefined percentage (95% [15]) of the points in the combined cluster.

3.3.4.4 Comparison

The method that is chosen is the N^{th} *percentile observations* method based on universal applicability and robustness to incorrect measurements. The maximum observed dimensions method does not leave any room for incorrect clustering, which is not desired as it is expected that some incorrect clustering will appear. With the enclosing percentage method it should be considered is that all measurements of an object are aligned in order to estimate the dimensions. If MinErrorRect is implemented, this alignment is already done when drawing the bounding box. If not, the alignment has to be implemented along the enclosing percentage method, thus increasing implementation and processing time. In section 3.3.1 is determined that a combination of PCA and MinErrorRect is used for the orientation estimation and thus the alignment is not known for pedestrians and two-wheelers. N^{th} percentile observations can be used independent of what orientation estimation method is used and can be tuned to be more precise or more robust.

Chapter 4

System

The system chapter describes the final version of the implemented system that is developed throughout this research. First, a complete system overview is given. Second, the data collection process described. And finally, each step of the data processing pipeline is explained.

4.1 System overview

Before all parts of the implemented system are described in detail, a general overview of the complete system is given. Fundamentally the system consists of three steps: data acquisition, data processing, and data analysis. The data acquisition step converts the raw LiDAR sensor readings into a usable data format. The data processing step processes the sequential point clouds to tracked objects that appear throughout the recording. The processing steps are: background identification and -removal, point clustering, object tracking, and object classification. The process is visualised in figure 4.1, including the *data analysis* step representing what can be done with the processed data.

4.2 Data collection

In this section the hard- and software setup that are used for collecting LiDAR data is described. Two almost identical setups are used, one on a car and one on a pole in a mobile metal enclosure as shown in figures 4.2a and 4.2b respectively. As the platforms are described, differences between the platforms will be noted whenever applicable. The data flow through the relevant hardware is shown in the *data acquisition* section of figure 4.1.

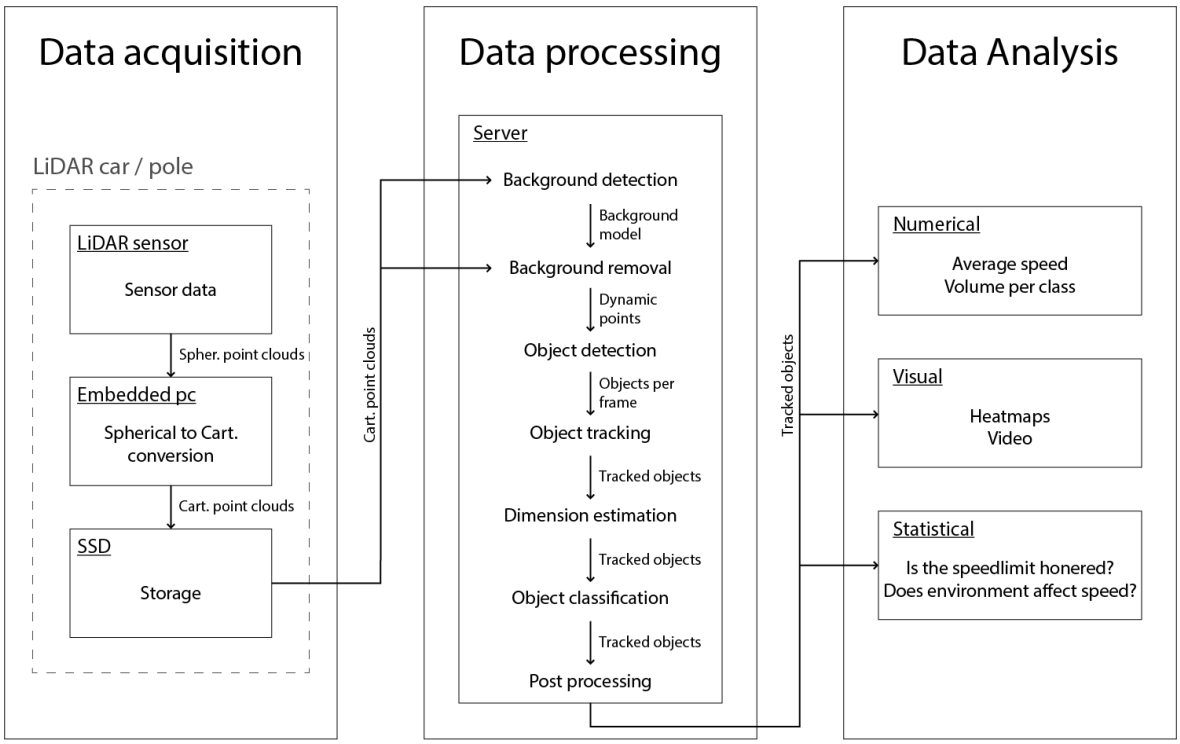


Figure 4.1: Simplified system overview of the complete data flow.

The LiDAR sensor is the source of all collected data and mounted at the highest point on each platform. The LiDAR sensor is also one of the main differences between the platforms as the car contains an OS1-32 sensor, while the pole contains an OS1-16 sensor, both from Ouster[1]. The relevant specifications of each LiDAR are stated in table 4.1. Within both LiDAR units an Inertial Measurement Unit (IMU) is present. Each sensor is connected to an interface box that provides connections for power and Ethernet. Power is supplied by a 12V battery through a voltage converter making it 24V. The Ethernet output is directly connected to the embedded computer.

The embedded computer used for receiving and storing the data from the LiDAR is a UDOO[2] X86 II ULTRA with a quad core 2.56GHz processor running Ubuntu 20.04. Connected to the computer is an external SSD for storing the LiDAR data. Relatively large storage is required as the OS1-32 can generate up to a gigabyte of data per minute. As the UDOO accepts 12V it is directly connected to the battery. Both LiDAR and IMU data is sent to the computer via Ethernet.

On the software side the communication, coordinate conversion, and storing is done using Robotic Operating System (ROS)[3], which runs on the UDOO. ROS is a set of software libraries and tools aimed at building robot applications. Each mea-

[1]https://ouster.com
[2]https://www.udoo.org
[3]https://www.ros.org/

(a) Mobile steel casing with the LiDAR mounted on an extendable pole

(b) Car with a LiDAR on a rigid roof mount

Figure 4.2: Hardware platforms used for data collection.

Table 4.1: Hardware specifications of both the Ouster OS1-16 and OS1-32

Sensor	**OS1-16**	**OS1-32**
Range (80% Lambertian Reflectivity)	110 m	100 m
Range (10% Lambertian Reflectivity)	50 m	45 m
Range accuracy	±5 cm - lambertian targets ±10 cm - retroreflectors	±3 cm - lambertian targets ±10 cm - retroreflectors
Precision	0.8 - 1 m: ± 1 cm 1 - 20 m: ± 1.1 cm 20 - 50 m ± 3 cm >50 m: ± 5 cm	0.3 - 1 m: ± 0.7 cm 1 - 20 m: ± 1 cm 20 - 50 m ± 2 cm >50 m: ± 5 cm
Range resolution	0.3 cm	0.3 cm
Vertical resolution	16	32
Horizontal resolution	1024 or 2048*	1024 or 2048*
Rotation rate	10 Hz or 20 Hz*	10 Hz or 20 Hz*
Field of view	Vertical: ±16.6° (33.2°) Horizontal: 360°	Vertical: ±22.5° (45°) Horizontal: 360°
Angular sampling accuracy	Vertical: ±0.01° Horizontal: ±0.01°	Vertical: ±0.01° Horizontal: ±0.01°
Points per second	327,680	655,360

* A horizontal resolution of 2048 can only be run at 10 Hz

surement of the LiDAR consists of positional data, namely range, vertical channel, and azimuth angle, and measurement data, namely intensity, reflectivity, ambient near-infrared, and time stamp. The ROS driver[4] provided by Ouster converts the UDP data stream from the LiDAR to a universal ROS data format called *PointCloud2*, which contains the point cloud in Cartesian coordinates and any additional provided data per point.

During the recording the point clouds are stored on an SSD in a *Rosbag* format. The Rosbag format is useful as the recording can be played back at a later points as if it is a live data stream. The recording is split into separate 15 minute files to prevent significant data loss during longer recordings and optimise the data processing.

4.3 Data processing

The pith of the matter of this research is the processing of the LiDAR recording into a list of tracked objects. A simplified overview of the processing steps is shown in the *data processing* section of figure 4.1. A visualisation of the resulting data after the data processing is shown in figure 4.3. The processing steps are discussed in order of data flow.

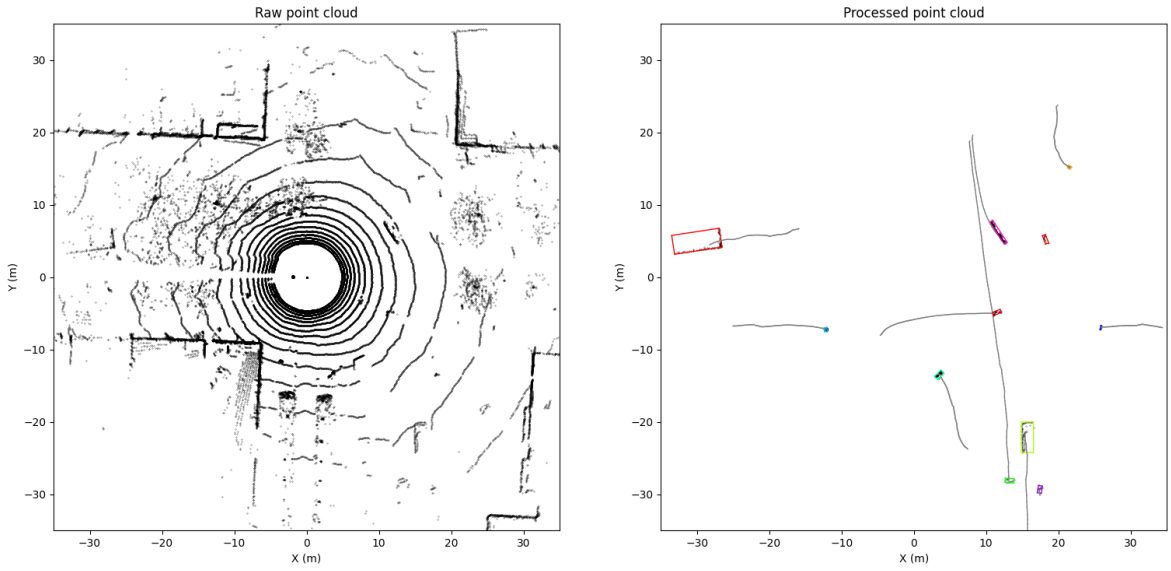


Figure 4.3: Visualisation of the resulting data (right) after all processing steps applied to the raw point cloud (left).

[4]https://github.com/ouster-lidar/ouster_example/

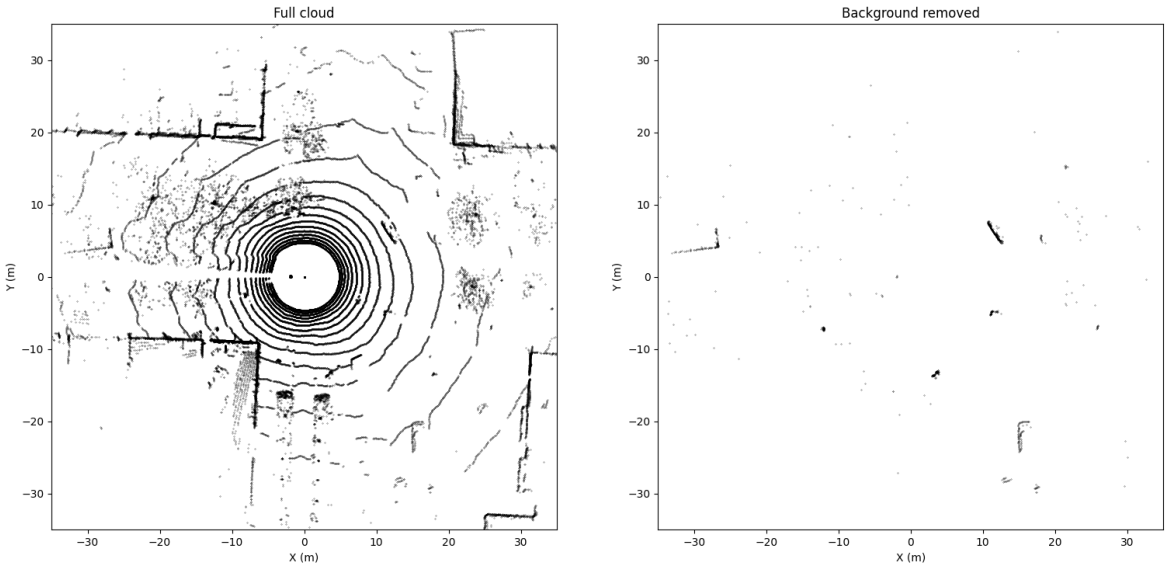


Figure 4.4: Example of the remaining data (right) after applying a BRM to a raw point cloud (left).

4.3.1 Background removal

Presented in this section are four Background Removal Method (BRM)s of which three are from related literature and one is a method developed during this research. The sought-after effect of a BRM is to remove all points corresponding to the background and keep those that correspond to objects as shown in figure 4.4. The first two methods, Threshold in Spherical Coordinates (TSC) and Peak Detection (PD), determine a range threshold for each measurement angle of the LiDAR. The remaining two methods, 3D Density Statistic Filtering (3D-DSF) and Raster-based Algorithm (RA), divide 3D space into equally sized cubes and individually label them as back- or foreground. The implementation of each method and any alterations that are made are described in this section.

4.3.1.1 Threshold in spherical coordinates

Threshold in Spherical Coordinates (TSC) is a method that generates a background model by determining the maximum range for each LiDAR angle in which a measurement is considered dynamic. As mentioned in chapter 3 TSC is developed by Tarko et al. [15]. TSC starts by taking a recording of at least 3000 frames, preferably during low traffic. In this research the LiDAR data is also processed after it is recorded and the 3000 frames are selected randomly from the recording. Then, for each angle of the LiDAR all measurements are grouped and for each group the mean and standard deviation are calculated using formula 4.1 and 4.2 respectively.

Finally, the range threshold is determined by the mean minus three times the standard deviation (formula 4.3).

The calculated range thresholds are stored in a two-dimensional matrix with identical dimensions to the sensor resolution. When the background of a new frame is filtered, the measured range of each LiDAR angle is compared against its corresponding range threshold. All measurements within the corresponding range threshold are kept as dynamic points.

$$\mu_{i,\alpha} = \frac{\sum_{n=1}^{n=k} D_{i,\alpha,n}}{b} \tag{4.1}$$

Equation 4.1 computes the mean of a group of measurements for laser i at angle α where:

$\mu_{i,\alpha}$ is the mean value of a group of readings from laser i at angle α
$D_{i,\alpha,n}$ is the distance reading given by laser i when fired at angle α in frame n
k is the total number of frames in the batch
b is the number of non-zero distance readings in the batch because a zero distance value means a null or no return

$$\sigma_{i,\alpha} = \sqrt{\frac{\sum_{n=1}^{n=k}(D_{i,\alpha,n} - \mu_{i,\alpha})^2}{b}} \tag{4.2}$$

Equation 4.2 computes the standard deviation of each group of measurements where $\sigma_{i,\alpha}$ is the standard deviation value of a group of readings from laser i at angle α.

$$c_{i,\alpha} = \mu_{i,\alpha} - 3\sigma_{i,\alpha} \tag{4.3}$$

Formula 4.3 describes the calculation for the range threshold c from laser i at angle α.

4.3.1.2 Peak detection

Peak Detection (PD) is a new BRM that is developed during the this research aimed at improving the background identification performance of TSC, while maintaining the processing speed as much as possible. Like TSC, PD generates a background model by determining the maximum range for each LiDAR angle in which a measurement is considered dynamic. PD also aims at improving the robustness with respect to the traffic intensity during the model generation.

While initial testing shows that TSC is capable of removing static objects that are never occluded, both dynamic- and occluded background objects result in large

standard deviations and thus in over-restrictive background models. PD combines close measurements in clusters and determines the relevance of said cluster based on the number of measurements. Traffic will pass by and generate a small measurement peak at a relatively short distance and background objects generate large measurement peaks as they are continually measured. The background model is generated using a random selection of frames to prevent stationary vehicles from generating large measurement peaks.

The PD method consists of four steps to determine the background model. The first step is to process the recording and store all measurements per LiDAR angle. Then, for each LiDAR angle close measurements are grouped together and too small peaks are removed. Third, based on predefined conditions a measurement group is selected (if present) and considered as background. And finally, the minimum range of the selected measurement group is taken and decreased by the range measurement inaccuracy of the LiDAR sensor as the maximum range threshold. Each step is explained in more detail next.

As with TSC, 3000 frames are randomly selected from a recording and used for determining the background model. For each LiDAR angle a histogram is generated using the Freedman-Diaconis rule [43] (equation 4.4) to compute the bin width. Whenever a measurement does not return any values due to range limitations it is discarded. Theoretically there are four types of measurement distributions possible (figure 4.5):

1. No measurements at all
2. Only measurements of passing traffic (figure 4.5a)
3. Only background measurements (figure 4.5b)
4. Background and passing traffic measurements (figure 4.5c)

Non-empty bins that are within three bin-widths of each other are grouped and their values summed. Whenever the summed value of a combined peak is larger than N% of the amount of selected frames it is considered a potential background element. This value is called the Relevant Peak Percentage (RPP). All peaks smaller than the RPP are removed. The result of this process is shown in figure 4.6. The optimal value for RPP is found to be 15% (see results section 6.1.2).

$$B_{i,\alpha} = 2\frac{IQR(D_{i,\alpha})}{\sqrt[3]{k}} \tag{4.4}$$

Equation 4.4 computes the width of the bins $B_{i,\alpha}$ to be used in a histogram, where $IQR(D_{i,\alpha})$ is the interquartile range of the group of readings $D_{i,\alpha}$ from laser i at angle α and k is the total number of frames in the batch.

When all peaks are detected within a group of measurements, a range threshold

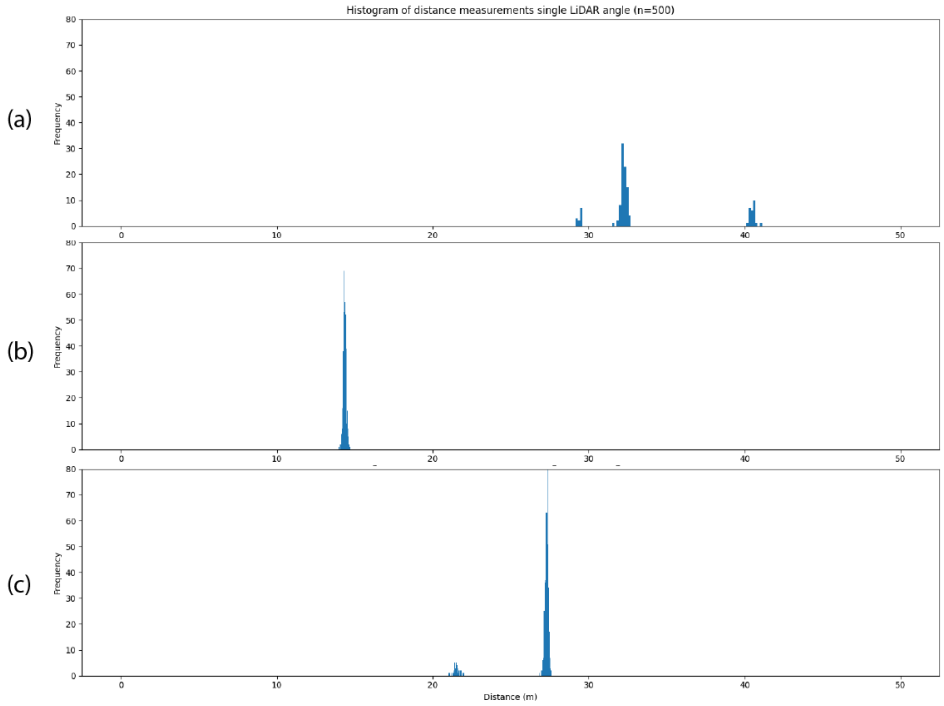


Figure 4.5: Examples of measurements from a single LiDAR angle. (a) Only passing traffic is observed (b) Only a static object is observed (c) both a static object and passing traffic is observed.

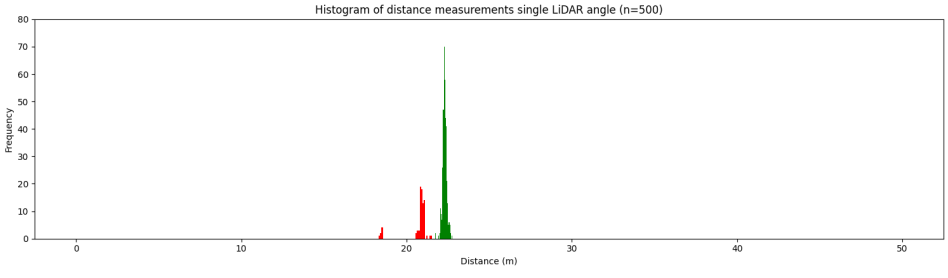


Figure 4.6: Histogram peaks are clustered together and clusters with a frequency lower than N% (here 30%) are removed (red), while bigger peaks are kept (green).

is determined if only one relevant peak is found. The distance to the closest side of the peak identified as the background determines the base background threshold value, which is lowered according to the range measurement inaccuracy at that distance of the used LiDAR sensor as shown in table 4.1. Both sensors have a maximum range accuracy of $\pm$10cm. Whenever no peaks are available at all the range threshold is set to infinite. All range thresholds are stored in a matrix identical in size to the sensor resolution, which represents the background model.

There are two situations in which additional processing is done to further improve the performance. The first situation is when no peaks larger than the RPP are present while having a significant amount of measurements in total ($\geq$RPP). For example, this could happen whenever the recording is made in a forest and leaves continually move in and out of the observation line of the LiDAR angle. In this case the range threshold should not be infinite as it will result in many unfiltered background measurements. An option is to leave the range threshold infinite and let the point clustering method in section 4.3.2 deal with the noise. However, again in case of a forest this can result in many background points close to each other, resulting in a falsely detected object. Thus, the LiDAR angle is considered too noisy and the range threshold is set to the shortest range measurement, reduced by the sensor's range accuracy.

The second situation is when multiple relevant peaks are present. This is the case when e.g. many vehicles pass the same location and both that location and the background generate a significant peak. In this case it is vital that the background peak is chosen correctly as otherwise all passing traffic is filtered out. Two conditions have been added to solve this problem, the first is that whenever there is a peak that is larger than 50% of the total number of measurements, it is considered the background peak. The second condition is that if there are multiple peaks, non of which larger than 50%, the peak furthest away is chosen as the background peak. The reasoning is that if a peak is present behind another peak, the peak in front cannot be a static object. Exceptions for this also exist, e.g. a flag in front of a building. However, given the presence of traffic, it is expected to improve the background model.

4.3.1.3 3D density statistic filtering

The 3D-DSF method is a method developed by Wu et al. [30] and divides the 3D space around the LiDAR in equally sized cubes labelled either background or not. 3D-DSF excludes background points based on the spatial distribution of laser points. Generally, the point density of background cubes is higher compared to non-background cubes due to the consistency of measurements within that cube. A

matrix is generated representing all (non-)background cubes which is used to filter new frames.

The method consists of four steps, the first being *frame aggregation* which is similar to the grouping of measurements per point as for TSC and PD. The difference begin that the aggregated frames are not stored per measurement angle, but per cube in 3D space as explained in the next step. As the implementation of this Graduation Project is not time critical, the recommended amount of frames determined by Wu et al. [30] is 3500, however, for consistency with the previous methods 3000 is used which is still within their recommendation.

In the second step, *point statistics*, the 3D space is divided into continuous cubes that can either represent background or not. All cubes in the space are stored in a 3D matrix and each matrix entry records the number of aggregated points in the corresponding cube. The size, and thus number, of cubes is a parameter that influences the ratio between processing speed and performance. More, smaller cubes results in better performing background identification but with a higher processing time. The recommended cube size by Wu et al. [30] is 0.1m. The size of the complete matrix is determined by the usable detection range of the LiDAR, measurements outside the matrix are filtered out.

The third step is *Threshold (TD) learning* where the density of each cube in space is calculated using the aggregated 3D points. A point density threshold of the cubes (TD) is determined to differentiate between background and non-background cubes. Cubes with a density higher than the TD are considered background cubes. Tweaking the TD parameter is a trade-off between considering slow-moving objects as background when it is too low and not excluding the distant background when it is too high.

Before the TD is determined the Field Of View (FOV) of the LiDAR is divided into multiple ranges to compensate for the points getting more sparse further away from the LiDAR. Six ranges are chosen (10, 15, 20, 30, 40 & 50) and for each range a TD is calculated using the following formula:

$$slope = \frac{F_i - F_{i-1}}{N_i - N_{i-1}} \tag{4.5}$$

"...where N_i is the i^{th} number of points per cube (from lowest to highest), and F is the frequency of i^{th} number of points per cube. When the slope first becomes 0 or positive, the frequency of number of points per cube in equation 4.5 (F) is used as TD." [30]

Adjustment An adjustment has been made to the 3D-DSF algorithm in order to improve performance. During initial testing it is discovered that the slope used for determining the TD is not a smooth decreasing line. Whenever there is a small peak in the slope, the TD is set to that frequency which is likely too low. Thus, a moving average filter is applied over the slope, increasing the average TD. Results show that increasing the moving average filter width improves recall at the cost of precision. A filter width of one (direct neighbours only) is used.

When all the cubes are labeled either background or not using the determined TD the positions of the background cubes are stored in a 3D matrix. The final step is to exclude any LiDAR points that fall within a cube labeled as background for each new incoming frame.

4.3.1.4 Raster-based algorithm

The same authors of the 3D-DSF method developed a further improved method called Raster-based Algorithm (RA), based on the principles of 3D-DSF, by looking at the change in point density per cube [31]. This has as advantage over 3D-DSF that no threshold has to be determined. RA consists of 5 steps: Region Of Interest (ROI) selection, frame aggregation, point statistics, background area detection, and background exclusion.

Before the *frame aggregation* and *point statistics*, as with 3D-DSF, an initial step is added for determining the ROI. As the height of the ROI is determined by using the angle of the laser aimed most vertically upwards in combination with the maximum detection range it results in a relatively high vertical limit. As vehicles in the Netherlands are not allowed to be taller than 4.0m there is no reason to set the height of the ROI higher[5]. This can significantly decrease the number of cubes in the matrix and thus increase the processing speed. The used ROI is l/w/h 50m/50m/4m, or when applicable the dimensions of the ROI are stated.

The *frame aggregation* is almost identical to 3D-DSF, with the only difference being that RA uses sequential frames instead of random ones. This makes it possible to detect peaks and valleys in the point density for each cube whenever an object passes or it is occluded. The *point statistics* step is identical to 3D-DSF.

The final step is the *background area detection* where each cube is labelled either background or not based on the change in point density over the aggregated frames. Lv et al. [31] claim that the point density of background cubes never changes more than two between consecutive frames when a vehicle passes, while

[5]https://www.om.nl/onderwerpen/beleidsregels/aanwijzingen/verkeer—vervoer/instructie-voertuigafmetingen-2017i007

non-background cubes can (see table 2 in [31]). The table present the number of cubes that have a change in point density of one to four between consecutive frames while a vehicle passes. None of the background cube point densities change more than two points, while those of the non-background cubes do.

An exception occurs when a background point is occluded, making the point density zero, then visible again, returning the point density to its original value which is potentially larger than two. It is assumed that, due to the length of the recording, a background point is only occluded for a small percentage of the measurements. With this assumption an extra distinction can be made by comparing the number of frames in which the point density is zero (D_0) against when it is larger than zero (D_+). When $D_+ > D_0$, the cube is assumed to be background.

Initial testing showed that the RA method as presented by Lv et al. [31] did not perform as expected, potentially due to interpretation and/or writing errors. The pseudocode shown in algorithm 1 is the algorithm that labels all cubes either background or foreground as interpreted from the paper. Section 6.1.4 elaborates on why the result is not as expected. Some effort is made to adopt the concept stated in the paper and implement a better performing version.

The adjusted cube labelling algorithm is shown in algorithm 2. It works opposite of the the original method as in this case all cubes are initially considered foreground. All cubes where $D_+ > D_0$ are considered background cubes based on the assumption that foreground cubes are not occupied more than half the frames. All remaining cubes that have at least 15% non-zero densities (*Pmin* in algorithm 2) are checked for the minimum density change (ΔD). The ΔD is initially set to two, however, it can be adjusted. If any of the sequential density changes exceeds ΔD the cube is kept foreground, otherwise it is background.

Algorithm 1 Original foreground cube identification.

```
Dif = density D of cube i in frame f
if min(Di) == 0 then
    if count(Di != 0) < count(Di == 0) then
        i is a foreground cube
    end
else
    if Di(f+1) - Dif > 2 then
        i is a foreground cube
    end
end
```

Algorithm 2 Adjusted RA cube identification.

Dif = density *D* of cube *i* in frame *f*

Pmin = Percentage *P* (15%)

$\triangle$D = Minimum density change

if *count(Di != 0)* $>$ *count(Di == 0)* **then**
  i is background cube
else if *count(Di != 0)* $>$ *(Pmin* $\cdot$ *count(Di))* **then**
  if *Di(f+1) - Dif* $>$ $\triangle D$ **then**
    i is a foreground cube
  else
    i is a background cube
  end
else
  i is a foreground cube
end

4.3.2 Clustering

When all background points are removed the remaining points need to be clustered into groups corresponding with objects. The effect of clustering is visualised in figure 4.7. Due to time constraints only one clustering method presented in section 3.2.2 is implemented, which is DBSCAN, for its proven functionality and general applicability.

4.3.2.1 Density-based spatial clustering Applications with noise

The clustering algorithm that is used is the Density-Based Spatial Clustering Applications with Noise (DBSCAN) algorithm [44] with improvements. This algorithm starts at a randomly selected point as well, and requires two parameters: epsilon (ϵ), which represents the radius around the initial point that will be considered, and *MinPts*, which represents the minimum number of points (including the starting point) that must be present within ϵ (called core points) for a cluster to be formed. Then, all points within a radius ϵ around the core points are also added to the cluster (called border points). This process is then also repeated for the border points until no new points are within range. After a cluster is complete, a new starting point is chosen and the process is repeated. This is done until all points are considered.

All points are projected on a plane parallel to the z-axis as it is assumed that traffic never vertically overlaps. This has the advantages that all points belonging to the same object are closer, and thus more likely to be clustered.

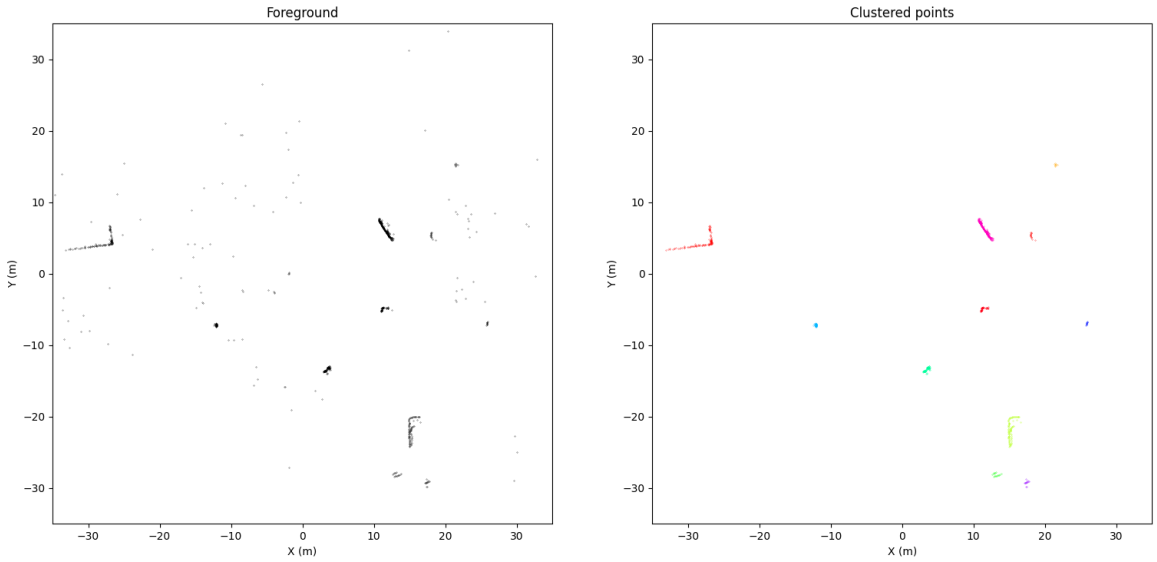


Figure 4.7: The goal of clustering is to assign closely located points to the same group to form objects and remove noise that does not belong to objects. Left shows the data before clustering and right shows the data after.

Originally, DBSCAN does not adjust for the sparsity over distance of a LiDAR generated point cloud. This is solved by making the parameters *MinPts* and ϵ dependant on the distance to the LiDAR. A separate *MinPts* function is fitted for both LiDAR sensors due to the increased resolution of the OS1-32. Formula 4.6 and 4.7 are for the OS1-16 and OS1-32 respectively. For ϵ a single function is defined as the horizontal resolution is identical, which is formula 4.8.

$$MinPts = \frac{500}{1.3d} - 12 \tag{4.6}$$

$$MinPts = \frac{500}{1.3d} - 5 \tag{4.7}$$

$$\epsilon = .03 * d \tag{4.8}$$

Where d is the distance to the LiDAR sensor.

4.3.3 Object tracking

After all objects in each frame have been detected, individual objects that appear in sequential frames need to be tracked as shown in figure 4.8. The tracking method Global Nearest Neighbour is chosen in combination with a Kalman filter. GNN associates objects with the smallest distance measure, in this case between a predicted

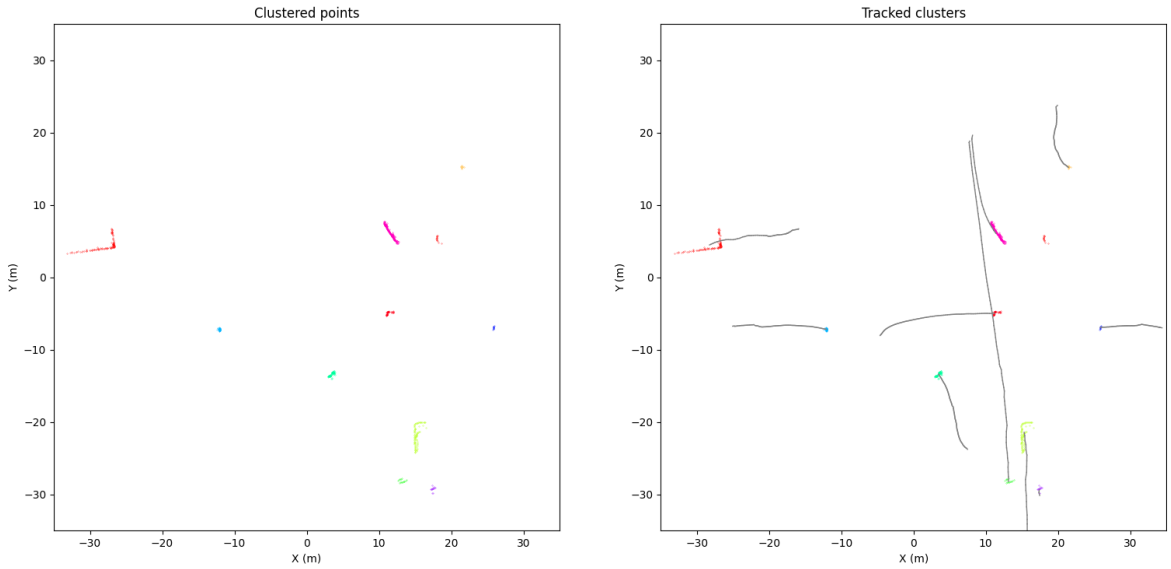


Figure 4.8: By drawing a line between the previous locations of a tracked object a 'path' is generated visualising the result of object tracking.

location and an actual measurement. The Kalman filter both predicts the location of an object based on previous states and updates the state using the prediction and the associated measurement. Both the object association using GNN and the Kalman filter are explained individually.

4.3.3.1 Object association

The GNN algorithm starts by considering all Detected Objects (DO)s in the first frame as new Actively Tracked Objects (ATO)s. In the following frame DOs are compared to the predicted locations of the ATOs (see next section for location predictions). Using the Euclidean Distance (ED) (equation 4.9) the distance between the location of all DOs and ATOs' predictions is calculated. The shortest distance is chosen, given that it falls within a predetermined range which is based on the maximum expected travel distance between frames. The maximum speed limit within city limits in the Netherlands is 50km/h (1.39 meter per frame), however, to ensure that all cars are properly tracked a velocity of 70km/h (1.94 meter per frame) is chosen as maximum travel distance between frames.

The ED is the length of a line segments between two points in Euclidean space and is calculated using the following formula for the distance $d(p, q)$ between points p and q:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2} \tag{4.9}$$

The are situations where the number of DOs is not equal to the number of ATOs or not all DOs are associated with the ATOs. In case an DO is not associated with any ATO, the DO is considered a new ATO. Any ATO needs to be recognised in at least 20 frames for it to be considered an actual object of interest. If an ATO is not associated with any DO, the ATO is updated as empty and not immediately removed. If an ATO is not associated for 10 consecutive frames it is removed. Each ATO has a unique ID and all ATOs with the required minimum amount of frames are stored.

4.3.3.2 Kalman Filter

To predict the next location and estimate the actual location of an object a Kalman filter is used. The GNN implementation as described in the previous section is combined with a pre-made two-dimensional Kalman filter by Sadli[6] to provide the objects' predicted location. An in-depth explanation of the provided Kalman filter is also provided by Sadli.

The state vector X_k is:

$$X_k = (p_x \;\; v_x \;\; p_y \;\; v_y)^T \tag{4.10}$$

Where p_x, v_x represent the position and velocity respectively in the x-direction and p_y, v_y in the y-direction.

The standard deviation of the measurements in both x- and y-direction are set to 0.15m based on the combination of the precision and accuracy as given in table 4.1. The magnitude of the process noise is set to one using the given value by Sadli and has not been tested further due to time limitations.

4.3.4 Orientation & dimension estimation

To determine the orientation and estimate the dimensions of objects four methods are implemented: Rotating calipers, Principal Component Analysis (PCA), Minimum Error Rectangle (MinErrorRect), and Nth percentile dimensions. The first three are bounding box methods that estimate the orientation of objects in each frame, while Nth percentile dimensions estimates the actual dimensions of an object based on all its observations. They are explained in the order as introduced.

[6] https://github.com/RahmadSadli/2-D-Kalman-Filter/blob/master/KalmanFilter.py

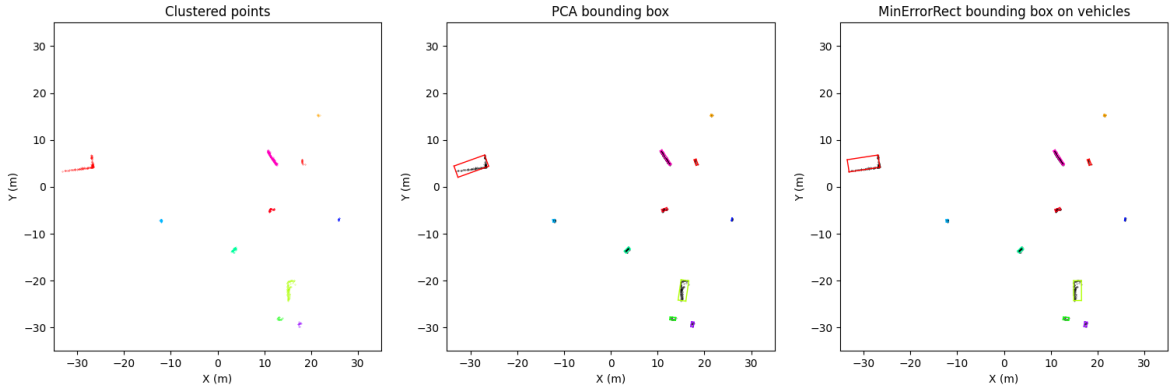


Figure 4.9: The left image shows the clustered points for which bounding boxes are calculated, the middle image shows the application of PCA for all objects and the right image shows the application of MinErrorRect on all vehicles and PCA for the remaining objects.

4.3.4.1 Rotating calipers

Even though Rotating calipers is already written of in the literature study, it is still included as a reference for evaluation. The method works by first computing the convex hull of the point cluster, creating a polygon. Then, a rectangular bounding box is aligned with each edge of the polygon, enveloping all points in the cluster. Finally, the rectangle with the smallest surface is chosen as the bounding box.

4.3.4.2 Principal component analysis

Principal Component Analysis (PCA) is not a bounding box estimation method by design, however, the first and second Principal Component (PC) of two-dimensional point clusters can be used to estimate the dimensions and orientation of objects. PCA starts by centering the point cluster of interest around the origin, subtracting the mean from the point coordinates but keeping the relative distances between the data. Then a line is drawn through the origin that best fits the data. The best fit is determined by rotating the line and minimises the sum of the squared distances from the projected points to the origin. The line with the lowest sum of squared distances is called PC1. Any additional PC also goes through the origin, is perpendicular to the previous PCs, and minimises the sum of squared distances.

Because the cluster points only have two variables: x and y, only two PCs can be made. This also means that, after PC1 is found, PC2 has only one possible orientation. Given that the shape of most vehicles and cyclists is that they are longer than they are wide and symmetric along the length, PC1 is also centered along the length of the object and lies in the direction of movement. The length of PC1

represents the length of the object and PC2 is perpendicular and thus its length represents the width. By aligning a rectangle around the endpoints of PC1 & 2 a bounding box is created.

4.3.4.3 Minimum Error Rectangle

The dimension estimate using edge angles from the Minimum Error Rectangle (MinErrorRect) method by Tarko et al. [15] is an alteration on the minimum area bounding box (determined using the Rotating calipers method). Tarko et al. [15] recognised that the minimum area bounding box is not always ideal and present an improved method by using a minimum error bounding box with respect to one of the edges of the object. The Euclidean distance between all the points and the specified edge are summed as the cost function, which is minimised. This ensures that the bounding box is parallel to at least one of the edges of the object.

In the related work chapter it is stated that both PCA and MinErrorRect are used as bounding box methods based on the object class. This means that the object class has to be known before a bounding box method can be applied. In the data processing this is solved by first applying PCA to all objects, and when later classified as vehicle, the bounding boxes are redrawn using MinErrorRect.

4.3.4.4 N^{th} percentile dimensions

The N^{th} percentile dimensions method makes a dimension estimation by combining all length or width measurements of one object and considers the N^{th} percentile as the closest approximation. The first step is to align all bounding boxes of an object to decide which measurements correspond to the width or length. This is done by measuring the angle between either parallel side of the bounding box and the tracked path between the current and previous frame as visualised in figure 4.10. Assuming all objects move forward, the side that aligns closest with the object's direction (α in figure 4.10) is the length. The second step is to take all length or width measurements and determine the N^{th} percentile. If no previous location is known, i.e. the first observation, than the longer side is chosen as the width. Even though objects are often longer than wide, the first observation is often at a relatively large distance and the object is facing towards the LiDAR on an intersection. This means that only the front, and thus the width, of an object is observed. The percentile that is initially chosen is 96^{th} based on the findings of Tarko et al. [15] for their enclosing percentage method.

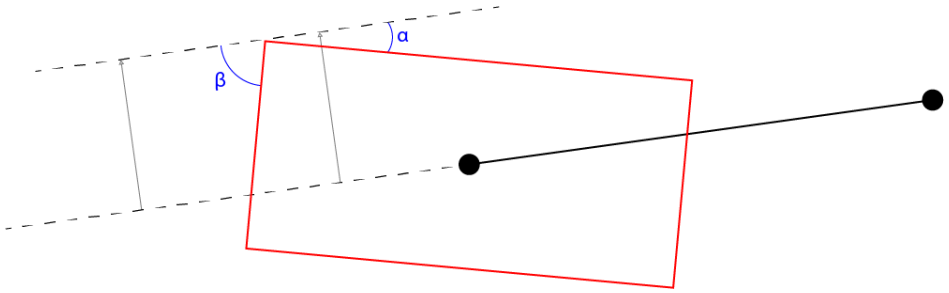


Figure 4.10: Calculating the angle between the parallel sides of the bounding box and the object's track. In this example α is smaller than β and thus the longer side of the bounding box is associated with the length of the object.

4.3.5 Object classification

The classification of objects is done using a manually defined decision tree based on an object's dimensions. The decision tree is shown in figure 4.11. The values chosen are based on the analyses in section 3.3.3.2 and of table 3.1.

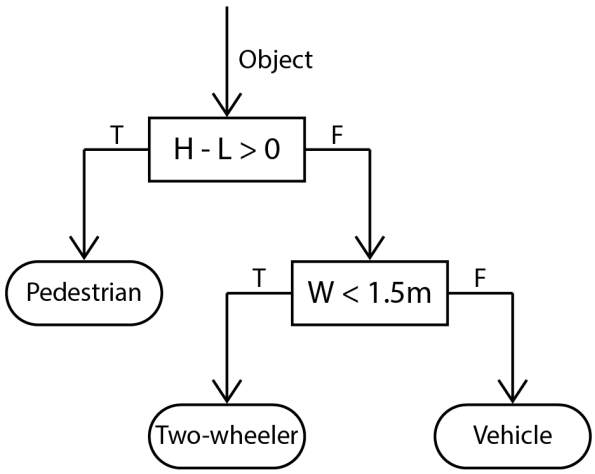


Figure 4.11: Visualisation of the decision tree used to classify each object, where H is height, L is length, and W is width.

Chapter 5

Method

The method describes what tests are done and what data is used in order to answer the research questions. The chapter is divided in two types of tests: system validation- and research tests. The research tests are designed to answer RQ1 & 2. In order to perform these tests a data processing pipeline is required, which is to be determined sufficient based on system validation tests. If the system is capable of outputting tracked and classified objects consistently, the main project goal is fulfilled. The combination of answering the RQs and implementing a functional pipeline are the backbone of answering the main research question.

Because the research tests require the system validation tests, all tests are done in order of data processing as described in the system chapter. The system validation also includes calibration of each processing step. Before the tests are described, first an overview of the used evaluation data is given.

5.1 Evaluation data

The implemented system is tested using a set of four recordings which are manually processed. The recordings are chosen based on the magnitude of movement from background objects and the amount of traffic during the recording. For each recording the date, location, duration, traffic composition, and background type is stated below in table 5.1. The traffic composition consists of pedestrians (P), bikes (B), cars (C), and trains (T). All two-wheeled vehicles are categorised as bikes.

All data processing has been limited to a range of 50m as beyond this distance the sparsity of the point cloud is too great to observe any objects with confidence. For the performance evaluation a couple of frames from each recording have been manually processed to identify all points belonging to traffic (object points). Table 5.2 shows the number of object- and background points for each frame. The frames

are visualised in appendix A, where red dots indicate object points and black dots indicate background points.

Table 5.1: Specifications of the data used for the evaluation of the implementation.

Name	Eikstraat	Kettingbrugweg
Sensor	OS1-16	OS1-16
Resolution	2048x16	2048x16
Frame rate	10	10
Date	29th of March 2021 at 12:31:44	29th of March 2021 at 11:37:57
Coordinates	52°16'07.0"N 6°47'40.3"E	52°14'52.6"N 6°49'17.8"E
Duration	∼15m (8997 frames)	∼15m (8998 frames)
Traffic	30P, 43B, 17C	11P, 35B, 5C, 4T
Background	Buildings only (mostly static)	Buildings and greenery (lightly dynamic)

Name	Holterbergweg	Torenstraat
Sensor	OS1-16	OS1-32
Resolution	2048x16	2048x32
Frame rate	10	10
Date	18th of April 2021 at 09:57:39	23rd of April 2021 at 10:06:40
Coordinates	52°17'55.1"N 6°25'11.7"E	53°06'24.5"N 6°06'03.3"E
Duration	∼15m (9001 frames)	∼14m 40s (8807 frames)
Traffic	4P, 53B, 30C	39P, 82B, 158C
Background	Greenery only (very dynamic)	Buildings only (mostly static)

Table 5.2: Specifications of the two frames per recording used for the evaluation of the implementation.

Dataset	Eikstraat		Kettingburgweg		Holterberg		Torenstraat	
Frame	1620	2623	1330	2998	384	1398	3748	4513
Total points <50m	32576	32574	32469	32469	32569	32501	64916	64858
Background points	32252	31324	31849	32238	32031	32159	62876	64816
Dynamic points	324	1250	620	231	538	342	2040	42
% Dynamic	0.99%	3.84%	1.91%	0.71%	1.65%	1.05%	3.14%	0.06%

5.2 Background identification & removal

The background identification & removal is the first segment of the implementation that is evaluated both with a validation- and research test. First, all BRMs are calibrated and validated. Then, the BRMs are compared in terms of performance and processing speed. The results of this test are used to answer RQ1. The best performing BRM is also used for any further testing.

5.2.1 Evaluation criteria

In order to evaluate the BRMs performance metrics are defined. The performance of the BRMs is tested similarly between existing literature with variances based on the use case. 3D-DSF is tested by measuring the exclusion percentage for both the background- and object points [30]. RA is tested based on the exclusion percentage of background points and observed objects, not mentioning what percentage of each object point is remaining. Another research by Wu et al. [45] using an alteration of 3D-DSF measures performance based on absolute numbers for the remaining background- and object points and a type 1 and type 2 error. To objectively evaluate the implemented BRMs four performance measures are calculated: precision, recall, and the type 1 and 2 errors.

The formulas for all performance measures are given by formula 5.1 to 5.4. Each frame has a number of Background Points (BP) and Object Points (OP) before filtering. The points after filtering are labelled Background Points Filtered (BPF) and Object Points Filtered (OPF) for background- and object points respectively. The precision (formula 5.1) calculates what percentage of the remaining points after filtering are actually a subset of OP. The recall (formula 5.2) calculates what percentage of OP remains after filtering. The type 1 error (formula 5.3) calculates what percentage of BP remains after filtering. Finally, the type 2 error (formula 5.4) calculates what percentage of OP is lost after filtering.

$$Precision = \frac{OPF}{OPF + BPF} \cdot 100\% \tag{5.1}$$

$$Recall = \frac{OPF}{OP} \cdot 100\% \tag{5.2}$$

$$Type\ 1\ error = \frac{BPF}{BP} \cdot 100\% \tag{5.3}$$

$$Type\ 2\ error = \frac{OP - OPF}{OP} \cdot 100\% \tag{5.4}$$

The processing speed evaluation is based on both a background model generation time and a background removal time. For each BRM a background model is generated using each data set five times using randomly selected frames, and each data set is fully processed to calculate the average background removal speed. The randomly selected frames are kept consistent between the BRMs. As the complete implemented system is not time critical, but does benefit from higher processing speeds, the evaluation of each BRM is biased towards the background removal performance. However, if multiple BRMs perform similar, the processing speed is a decisive factor.

5.2.2 Calibration & validation

The calibration is done to find optimal parameter values for each BRM that are used for further testing. Optimal parameters for TSC and RA are already given, however, due to hardware differences these values are validated. Each BRM has at least one tuneable parameter that determines the balance between precision and recall of the background removal.

In this research the calibrated BRM has the highest recall in combination with a manageable precision. Meaning that as much relevant data is kept as possible, while the following processing steps can deal with the noise. However, a precision percentage does not indicate the distribution of the noise throughout the FOV. I.e. a high precision in combination with dense noise clusters is more problematic than a low precision with evenly distributed noise. As it is currently unknown how a low precision will affect the results, the BRMs are calibrated at the 'elbow' of the recall curve where a decrease in precision does not yield a significant increase in recall.

All BRMs are calibrated using two manually labelled frames per recording (table 5.1 & 5.2) three times, using three different sample sets for the background generation. A visualisation of each frame can be found in appendix A. The sample sets used for the background generation are consistent between the BRMs. RA is the only method that must use consecutive frames, which are also varied between runs. The average precision, recall, and type 1 & 2 error are calculated for each parameter value for each BRM.

TSC The adjustable parameter of TSC is the Standard Deviation Multiplication (SDM), which is multiplied with the standard deviation and then subtracted from the mean distance measurement to determine the distance threshold for the background model. Further explanation on how the method works is described in section 4.3.1.1. The recommended multiplication value given by Tarko et al. [15] is three. A

calibration test is done to see whether three is also the optimal value in this research.

PD The adjustable parameter of PD is the percentage at which a peak in the distance measurement histogram is considered to be a relevant peak, called Relevant Peak Percentage (RPP). Further explanation on how the method works is described in section 4.3.1.2. As the method is developed during this research there is no recommended value for the RPP.

3D-DSF By adding a moving average filter to 3D-DSF the width of the filter is the adjustable parameter that is calibrated. Further explanation on how the method works is described in section 4.3.1.3. As the method originally does not include a moving average filter, there is no recommended value for the width of the filter.

RA The adjustable parameter of RA is the minimum density increase between consecutive frames for a cube to be considered non-background. Further explanation on how the method works is described in section 4.3.1.4. The recommended value given by Lv et al. [31] is three. A calibration test is done to see whether three is also the optimal value in this research.

The calibration of each BRM also acts as the validation. As a different data set is used it is difficult to compare the results against those of the original authors. However, based on the resulting performance metrics a substantiated claim can be made on whether the BRMs function as expected.

5.2.3 Comparison

The comparison of the BRMs is based on the performance as well as the processing speed. First, the optimal performing BRM configurations are used to determine the processing speed for both the background model generation and background removal times. After both tests are done, one BRM is chosen that is used for further testing. The choice is based on both results, however, it is biased towards background removal performance as the implementation is not time critical.

Independent two-sample t-tests are done between methods to determine whether the difference is statistically significant. All tests are done at a significance level of 0.05. One-tailed tests are done for testing whether one method is statistically better, using the following hypotheses:

$$H_0 : \mu_1 > \mu_2$$

$$H_a : \mu_1 \leq \mu_2$$

5.3 Clustering

The dynamic DBSCAN clustering method is both calibrated and validated using the four data sets in table 5.1. The calibration consists of fitting an ϵ and *MinPts* curve as a function of distance. Initially, an empirically determined value for ϵ is used as a starting point. The function for ϵ is:

$$\epsilon = .03 * d \tag{5.5}$$

Where d is the distance to the LiDAR sensor in meters. A minimum cluster size of five points is chosen to limit the noise coming through.

MinPts The function for *MinPts* is fitted to the data according to the cluster size of pedestrians. As pedestrians are the smallest object type that has to be detected in both length and width, the *MinPts* function may filter out all clusters that are smaller. A number of pedestrians are manually selected from the data sets and their number of points plotted against the distance to the LiDAR sensor. It is expected that this plot will form a relatively consistent curve to which a *MinPts* function can be fitted. After the implementation of the *MinPts* function a validation plot is made to determine whether it functions as expected.

Epsilon A similar process as determining the *MinPts* function is used to determine ϵ as a function of the distance to the LiDAR sensor. The goal of calibrating the dynamic ϵ is twofold. The first is to verify that a dynamic ϵ is an improvement over a static value. If so, the second is to calibrate the function for ϵ. To evaluate the results of multiple values of ϵ the distance between neighbouring points is calculated using Delaunay triangulation for each cluster extracted from the data sets in table 5.1. The 15th, 50th (mean), and 85th percentile of distances between points within a cluster over distance are plotted. the 15th and 85th percentile are included to represent the minimum and maximum distance within a cluster as the percentiles exclude

vertically overlapping and neighbouring border points respectively. There are no expected results for the point distances within clusters, thus the results are visually examined.

To validate if DBSCAN and the calibrated functions for ϵ and *MinPts* perform as expected, it is applied to the eight manually labelled frames from table 5.2. The validation is done based on the number of objects that are correctly clustered, merged, split, or not detected at all.

5.4 Tracking

The performance of the implemented Kalman filter tracking is validated by comparing the number of tracked objects to the number of manually determined objects within each data set. The goal is not to provide an in-depth analysis of the tracking algorithm, but to confirm that performance is sufficient for further testing. The tracking data is also visualised as a video and manually examined to substantiate why the number of tracked objects may not correspond to the expectations.

5.5 Classification

The goal of the classification of objects is not only to identify the object type, but also determines what orientation estimation method is applied. As time limitations did not allow for an extensive classification method, its primary goal is to differentiate between pedestrians & two-wheelers, and vehicles for the validation of the orientation estimation. Both pedestrians and two-wheelers use PCA as orientation estimation and vehicles use MinErrorRect. The validation of the classification method determines whether its performance is sufficient to evaluate the orientation estimation.

To validate the classification the processed data is visualised and manually checked. Each data set is rendered as a video and for all tracked objects the determined object class is compared against the actual object class. The tracked objects are the result of the implemented algorithm, meaning that objects are often split into multiple tracked objects. Therefore, the number of classified objects can be higher than the number of actual objects in the data sets. Tracked objects that do not represent traffic and grouped objects with multiple classes are ignored. Grouped objects of the same class are kept and labelled as their corresponding class. Besides a numerical validation the data is also observed for anecdotal evidence.

The results consist out of two parts: the performance of applying the correct ob-

ject class and of applying the correct orientation estimation method. The difference being that for the orientation estimation performance the pedestrian and two-wheeler classes are combined. The performance is measured based on precision and recall. The precision calculates what percentage of all objects labelled as class *X* also belong to class *X*. The recall calculates what percentage of all objects belonging to class *X* is also labelled as *X*. The performance of the orientation estimation dictates whether performance is sufficient for testing the orientation estimation.

5.6 Orientation estimation

To answer RQ2 on whether class-based bounding box methods perform better in estimating the orientation the orientation is compared against an object's next location. The assumption is that if the angle between the object's estimated orientation and the angle between the current and next location is equal, the estimation orientation is correct. This research test aims at evaluating how well the three implemented bounding box methods align with the next location of an object and whether using different methods per object class improves alignment.

To evaluate the performance of each bounding box method the angle between the orientation of the object and its next location is used as performance metric. This means that if the next location lays on the line of the orientation (if extended), the difference in angle is 0°and the orientation aligns perfectly. If the next location does not lay on the line, the angle difference indicates how many degrees the object orientation needs to be rotated in order to align them.

The method in section 4.3.4.4 is used to determine which side of the bounding box is the length and thus represents the direction. This is validated by plotting the angle difference between the chosen direction and the previous location, if functioning correctly, all angles fall within $\pm 45°$. Then, the angle difference between the orientation and the next location is calculated. The resulting angles lie in a range of $\pm 180°$. This calculation is done for each object in each frame, given that the object has both a previous and next location.

To determine the difference between the bounding box methods, each method is applied to every object class. Then, paired t-tests[1] are done between methods applied to the same object class at a 0.05 significance level. The paired t-test calculates the difference between the absolute angle differences of the tested methods for every sample and determines the average μ_d. This average is used for the one-tailed hypotheses:

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest$_rel.html$

$$H_0 : \mu_d = 0$$

$$H_a : \mu_d > 0$$

The absolute angle differences are also visualised in a histogram, including a mean and median for easier visual comparison. All measurements are divided into five ranges between 0 and 50m to find correlation between the angle difference and distance to the sensor. A lower average does not necessarily indicate better performance, as a few large angle differences can significantly increase the average. Therefore, the median is also of interest. The lower the average and the more right-skewed (median $<$ mean) the results the better.

Chapter 6

Results

6.1 Background identification & removal

6.1.1 TSC calibration

It is expected that a small Standard Deviation Multiplication (SDM) value results in a relatively mild filter as the range threshold lies close to the mean of the aggregated measurements. Whenever a list of aggregated measurements only contains measurements of a static background and the SDM is one, the standard deviation is smaller than the measurement precision of the LiDAR leading to false positive measurements and a low precision, but a high recall. A high SDM is expected to result in an aggressive filter that also filters out OP and thus gives a high precision, but a low recall.

To verify whether three is indeed the optimal value, multiplication values between five and one are tested and the results are shown in table 6.1 and figure 6.1. The values of which table 6.1 is derived can be found in appendix B. The turning point where the precision decreases more than the recall increases is at an SDM of three, thus agreeing with the recommended value of three.

Table 6.1: TSC calibration. Average performance results of applying TSC at eight different frames three times using a standard deviation between 5 and 1.

SD	5	4	3	2	1
Precision	81.31%	79.74%	72.89%	53.02%	27.95%
Recall	48.43%	55.70%	65.88%	73.80%	83.88%
Type 1 error	0.05%	0.09%	0.27%	1.37%	5.52%
Type 2 error	51.57%	44.30%	34.12%	26.20%	16.12%

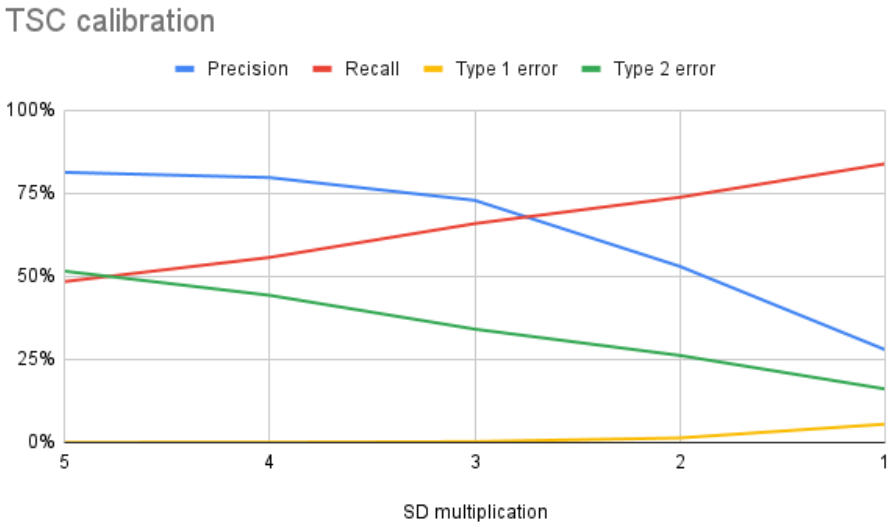


Figure 6.1: This graph is a visual representation of the performance metrics in table 6.1.

6.1.2 PD calibration

It is expected that both a low and high Relevant Peak Percentage (RPP) result in a low precision with high recall and that the inverse is true between those values. Starting with a high RPP, only the most certain background measurements are filtered out and thus most object points are kept in combination with many background points. This is also expected to be true for low RPP values as this results in many relevant peaks of which the furthest back is chosen.

For the calibration of PD 14 RPP values are tested and the results are shown in table 6.2 and figure 6.2. The values of which table 6.2 is derived can be found in appendix C. As can be seen in figure 6.2, the elbow of the recall curve lies at an RPP of 15%, which is adopted as the optimal value.

6.1.3 3D-DSF calibration

It is expected that by increasing the width of the moving average filter the threshold for a cube to be considered background decreases. Which in turn results in fewer background cubes and thus a higher recall and lower precision. Because the moving average smooths out the curve on which the threshold is based, the threshold calculation is less affected by unexpected peaks in the curve.

For the calibration of 3D-DSF five moving average widths are tested and the results are shown in table 6.3 and figure 6.3. A width of one means that both neigh-

Table 6.2: PD calibration. Average performance results of applying PD at eight different frames three times using a minimum peak percentage between 1% and 50%

RPP (%)	1	2	3	4	5	10	15
Precision	84.80%	81.66%	79.90%	78.95%	77.36%	71.19%	67.37%
Recall	72.93%	75.61%	77.59%	80.52%	81.73%	86.50%	93.09%
Type 1 error	0.07%	0.12%	0.17%	0.20%	0.23%	0.37%	0.49%
Type 2 error	27.07%	24.39%	22.41%	19.48%	18.27%	13.50%	6.91%

RPP (%)	20	25	30	35	40	45	50
Precision	64.88%	62.66%	61.15%	59.72%	58.59%	57.80%	57.62%
Recall	93.03%	92.66%	92.72%	92.02%	91.52%	92.23%	92.66%
Type 1 error	0.57%	0.63%	0.69%	0.73%	0.77%	0.81%	0.82%
Type 2 error	6.97%	7.34%	7.28%	7.98%	8.48%	7.77%	7.34%

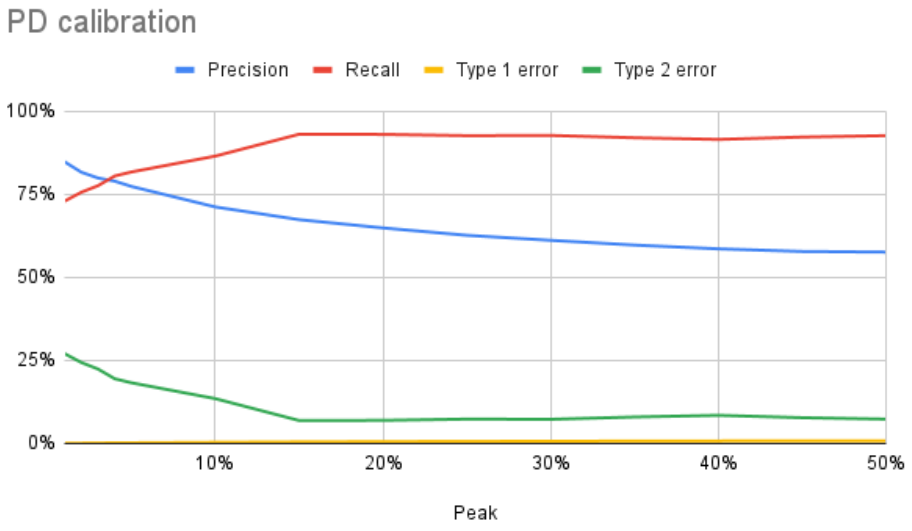


Figure 6.2: This graph is a visual representation of the performance metrics in table 6.2

Table 6.3: 3D-DSF calibration. Average performance results of applying 3D-DSF at eight different frames three times without and with a moving average filter width from 0 to 4.

MA filter width	0	1	2	3	4
Precision	74.72%	70.16%	66.69%	64.68%	63.48%
Recall	85.30%	92.77%	93.97%	94.34%	94.70%
Type 1 error	0.31%	0.44%	0.55%	0.61%	0.65%
Type 2 error	14.70%	7.23%	6.03%	5.66%	5.30%

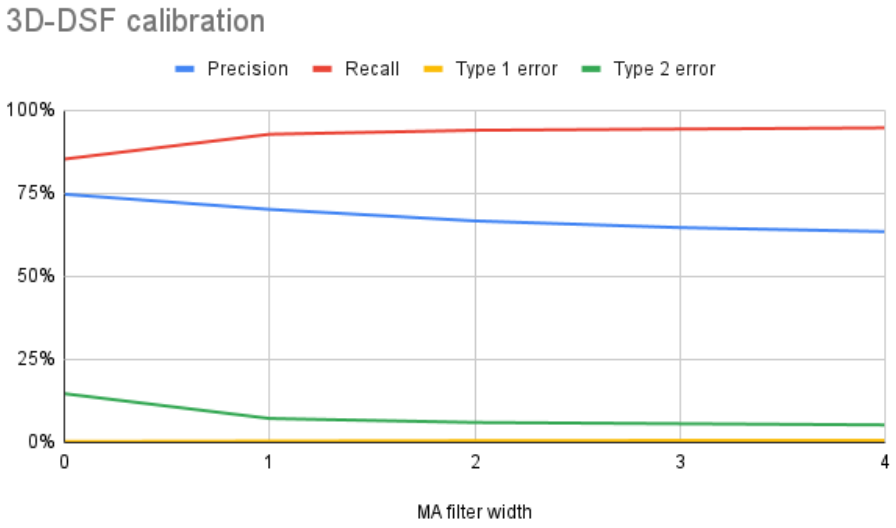


Figure 6.3: This graph is a visual representation of the performance metrics in table 6.3

bours of a value are included in the average. The values of which table 6.3 is derived can be found in appendix D. The elbow of the recall curve lies at a moving average filter width of one, thus chosen as optimal value.

6.1.4 Raster-based algorithm

The implementation of the RA method does not perform as expected and will therefore also not be calibrated for further use. An example of the application of algorithm 1 and its result are shown in figure 6.4. As mentioned in 4.3.1.4 it is possible that the order of processing steps is interpreted incorrectly and thus the RA method is adjusted accordingly as stated in algorithm 2. However, even after these adjustments there are fundamental problems with RA.

The most prominent problem with RA is that it does not account for the sparsity of a LiDAR generated point cloud. Lv et al. [31] show that the density of background cubes does not change more than two points between consecutive frames. However, figure 6.5 shows that increasing the minimum density change for a cube to be foreground decreases the number of background points left in the resulting point cloud. This proves that the density of background cubes (especially close to the LiDAR sensor) does change more than two.

Figure 6.5 also shows that increasing the minimum density change decreases the number of remaining object points (*dp* in figure titles) as it becomes increasingly more improbable for cubes to have a density change higher than the minimum and

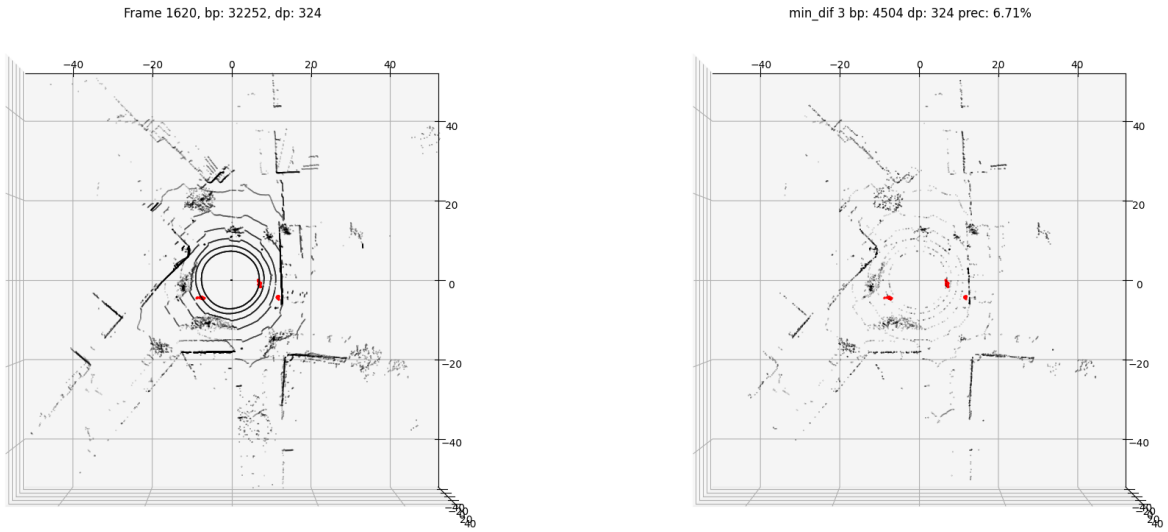


Figure 6.4: The left image shows a raw point cloud and the right image shows the result after applying the original RA method.

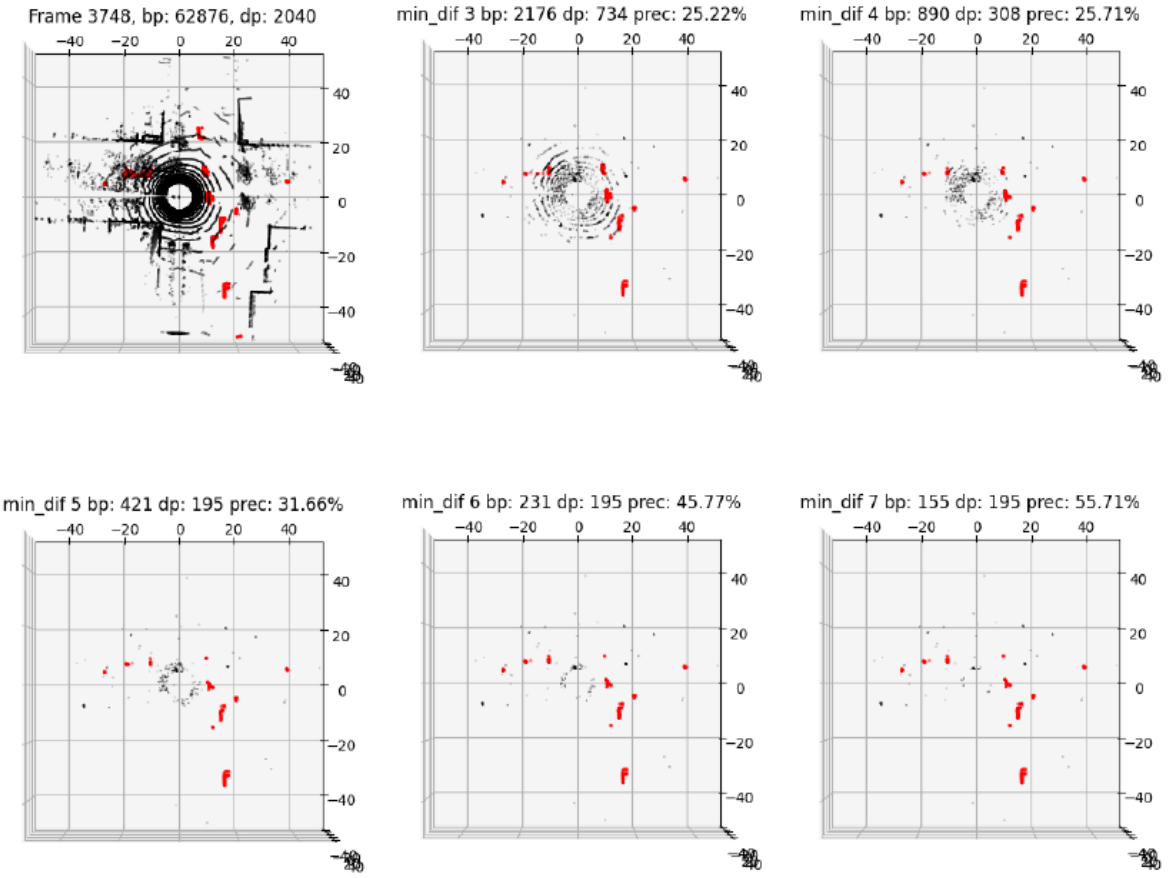


Figure 6.5: Here the effect of increasing RA's minimum density increase parameter is shown. The top-left image shows the raw point cloud while the remaining images show a minimum density increase value between 3 and 7 from top-middle to bottom-right.

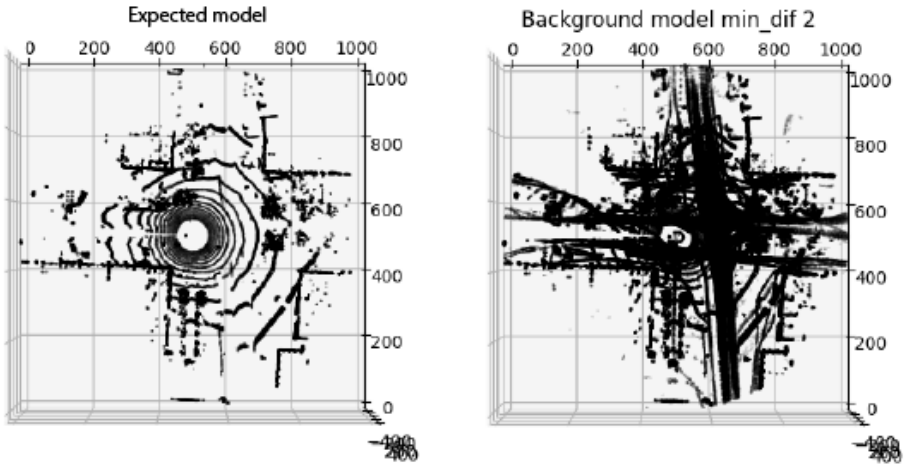


Figure 6.6: The left image shows an estimation of the expected background model using the RA method as each black dot is a cube that corresponds to a background object. The right image shows the actual model generated by the RA method and shows that all moving traffic is also labelled as background.

thus be labelled as foreground. This is also clearly visible when visualising the background model instead of the resulting point cloud as shown in figure 6.6. Here the thick black lines indicate where traffic is passing and should be excluded from the background model based on the density increase, however, as the density increase does not exceed the threshold the corresponding cubes are labelled as background.

Using geometry the maximum distance for a cube to have at least a density of 3 is calculated. Based on the specifications of the OS1-32 in table 4.1 the angle between vertical measurements is 0.176° and 1.406° between horizontal measurements. For a cube with 10cm sides the vertical and horizontal diagonal are 14.142cm. Meaning that the distance between two horizontal points can be at most 7.071cm to fit 3 horizontal points in 1 cube. This can only happen within a distance of 23.019m. The other option is 14.142cm between horizontal- and vertical points to fit 2 horizontal- and 1 vertical point, giving a maximum distance of 5.763m due to the larger vertical angle. Doing the same calculations for the sensor used by Lv et al. [31] (Velodyne VLP-32c) results in a maximum distance of 40.514m for 3 vertical points. Due to the limitations of the LiDAR sensor used in this research the RA method will not be considered from this point.

6.1.5 Processing speed

In this section the results of the processing speed are presented for both PD and 3D-DSF. Due to time limitations, PD being better than TSC across the board, and expected similar processing times, TSC has not been evaluated in terms of processing speed.

It is expected that for the PD method the background model generation may differ slightly between data sets as the algorithm benefits from angles that observe stationary objects or no objects at all. The more the measurements of one angle are distributed over the range, the more processor cycles are required to determine the peaks within the range. Angles that did not receive a significant amount of measurements are skipped entirely. The same is expected for the 3D-DSF method as a more dynamic environment results in more cubes within the background matrix having density measures and thus requiring processor cycles. All cubes within the matrix that do not have any density measurements are skipped. Thus, more dynamic scenes are expected to result in longer processing times for both background model generation methods. The background removal duration for a single frame is expected to be dependent on the number of measurements that fall within the defined ROI for both BRMs. All points within a new frame that do not fall within the ROI are immediately removed and thus speed up the process.

Between PD and 3D-DSF it is expected that PD is the faster method in both background model generation and background removal. This hypothesis is based on the fact that PD generates a 2D matrix with the dimensions of the LiDAR's resolution, while 3D-DSF generates a 3D matrix. Given the OS1-32 and a ROI of 50m horizontally and 4m vertically, PD and 3D-DSF generate a matrix containing 65,536 and up to 10,000,000 values respectively. Both generating and using the larger matrix is expected to be slower.

The results of the processing speed evaluation are shown in table 6.4. The average model generation speed for PD is 28.00s and 55.55s for the OS1-16 and OS1-32 respectively. For 3D-DSF this is 50.61s and 95.85s respectively. The background removal times are on average 13.5ms & 27.2ms for PD and 111.6ms & 204.2ms for 3D-DSF for the OS1-16 and OS1-32 respectively.

Table 6.4: Average processing speeds of PD and 3D-DSF for the background model generation using 3000 frames and the background removal of one frame.

	PD				3D-DSF			
	Model generation (s)		Background removal (ms)		Model generation (s)		Background removal (ms)	
Dataset	μ	σ	μ	σ	μ	σ	μ	σ
Eikstraat	35.33	0.19	13.4	0.4	52.70	0.39	116.8	3.3
Kettingbrugweg	25.37	0.13	13.6	0.4	44.49	0.40	98.7	0.9
Holterbergweg	23.31	0.65	13.5	0.4	54.63	0.48	119.5	1.0
Torenstraat	55.55	1.46	27.2	1.1	95.85	0.60	204.2	3.0

6.1.6 Comparison

The results of the performance and processing speed tests are summarised in table 6.5. For each BRM the optimal performing parameters are used. The processing speeds are the averages of table 6.4.

The comparison is done between PD and 3D-DSF. Between TSC and PD, PD always outperforms TSC based on the trade-off between recall and precision. TSC has a higher precision than PD at 79.31%. However, looking at table 6.2 PD with an RPP of 5% has both a higher precision (82.85%) and a higher recall (86.13%), making TSC irrelevant.

In terms of background removal performance PD and 3D-DSF perform fairly similar. 3D-DSF has a 4.06% higher precision, while PD has a 0.34% higher recall. The difference between the precision and recall of both BRMs is tested at a 0.05 significance level. 3D-DSF's precision is not significantly higher ($t(46) = 0.46,\ p = 0.32$). PD's recall is also not significantly higher ($t(44) = 0.07,\ p = 0.47$)

On average the model generation- and background removal times for PD are 43.82% and 87.53% faster than 3D-DSF respectively. For both PD and 3D-DSF the difference between using a vertical resolution of 16 and 32 is approximately a factor two for model generation- and background removal time. The difference between the background model generation and -removal speeds are tested at a 0.05 significance level with the hypothesis that PD is faster than 3D-DSF. The difference in processing speed is significant in the following four situations:

- Background model generation OS1-16: $t(26) = -0.46,\ p < 1.00 \cdot 10^{-5}$
- Background model generation OS1-32: $t(5) = -57.09,\ p < 1.00 \cdot 10^{-5}$
- Background removal OS1-16: $t(29975) = -9229.30,\ p < 1.00 \cdot 10^{-5}$
- Background removal OS1-32: $t(12756) = -4978.61,\ p < 1.00 \cdot 10^{-5}$

Table 6.5: Performance comparison between the calibrated background removal methods.

Method	TSC	PD	3D-DSF
Precision	72.89%	67.37%	70.16%
Recall	65.88%	93.09%	92.77%
Type 1 error	0.27%	0.49%	0.44%
Type 2 error	34.12%	6.91%	7.23%
Model generation	-	34.89s	61.92s
Background removal	-	16.93ms	134.80ms

6.2 Clustering

The clustering method is applied to frames processed using the PD method with an RPP of 15% for background identification & removal.

6.2.1 Dynamic MinPts

The results for the data set(s) using the OS1-16 and OS1-32 are shown in figure 6.7. To understand which clusters are of interest three pedestrians are selected and plotted in the same manner as shown in figure 6.8. As pedestrians are the smallest objects of interest, any clusters smaller than a pedestrian can be filtered out. Based on the results of the pedestrians' clusters threshold functions for both LiDAR sensors are determined, which are stated in formula 4.6 and 4.7 for the OS1-16 and OS1-32 respectively.

The threshold function that is fitted filters out all clusters that are too small considering the distance to the LiDAR sensor. The results after the implementation of both filter functions is shown in figure 6.9. Both thresholds filter the clusters that are considered too small as expected.

6.2.2 Dynamic epsilon

Figure 6.10 shows the application of multiple functions for ϵ. The 15^{th}, 50^{th} (average), and 85^{th} percentile of distances between points within a cluster over distance are plotted to exclude the distances between overlapping points and outer-shell points. The used function for ϵ is plotted as a black line and a red line indicates the minimum distance possible between two horizontally adjacent LiDAR angles.

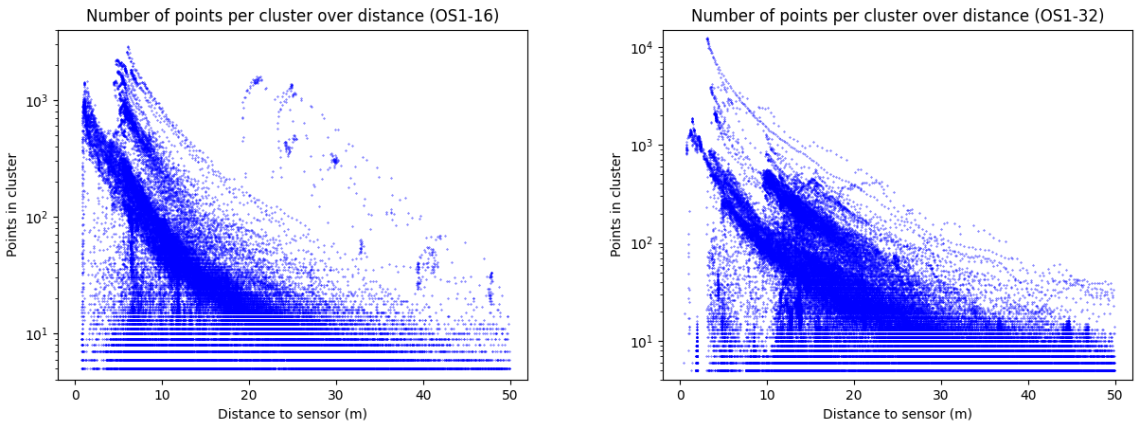
Whether the addition of a dynamic value for ϵ is an improvement is evaluated by first using static values. Figure 6.10a & 6.10b show an ϵ of 0.2 and 1.0 respectively.

Having a low static ϵ of 0.2 clearly shows that the maximum range is decreased, while a high value of 1.0 introduces peaks with higher average point distance.

As mentioned in the previous section the empirically determined function for ϵ is given by formula 5.5, which resulted in figure 6.10d. The 15^{th} percentile aligns with the minimum horizontal point distance. The 15^{th} percentile point distance can be lower due to the stacking of vertical LiDAR angles. Distances can also be higher than ϵ due to neighbouring border points. It seems like the ϵ function prevents less dense clusters from forming due to the average distance lying relatively close to the corresponding ϵ.

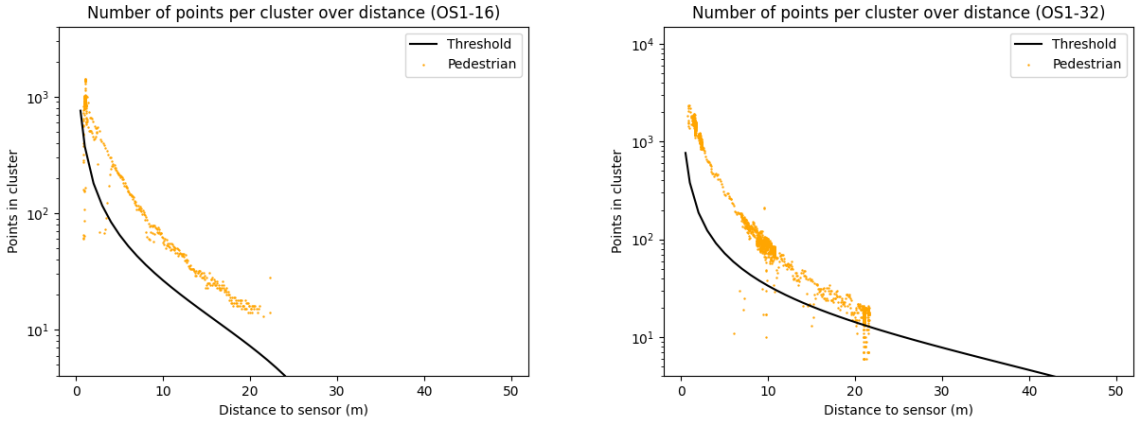
Varying the function of ϵ (figures 6.10c-6.10e) shows that the average point distance may be significantly influenced by neighbouring border points. E.g. a crescent-shaped cluster can be formed by the proximity of the points, however, calculating the average point distance using Delaunay triangulation will also include the distances within the concave part of the crescent, which are larger. If neighbouring border points do have influence, the average point distance will be relatively unresponsive to a change in ϵ. This is tested by changing the multiplier of ϵ to 0.015 and 0.06, as shown in figure 6.10c and 6.10e respectively.

The results of varying the function for ϵ are similar, apart from the additional peaks in 6.10e. As expected, the average point distance does not necessarily increase after increasing the ϵ multiplier. This can best be seen by the distance between ϵ and the average point distance, which increases when ϵ increases. The new peaks in figure 6.10e are the same as in figure 6.10b. Which function for ϵ is optimal is determined based on the eight manually labelled frames of table 5.2.



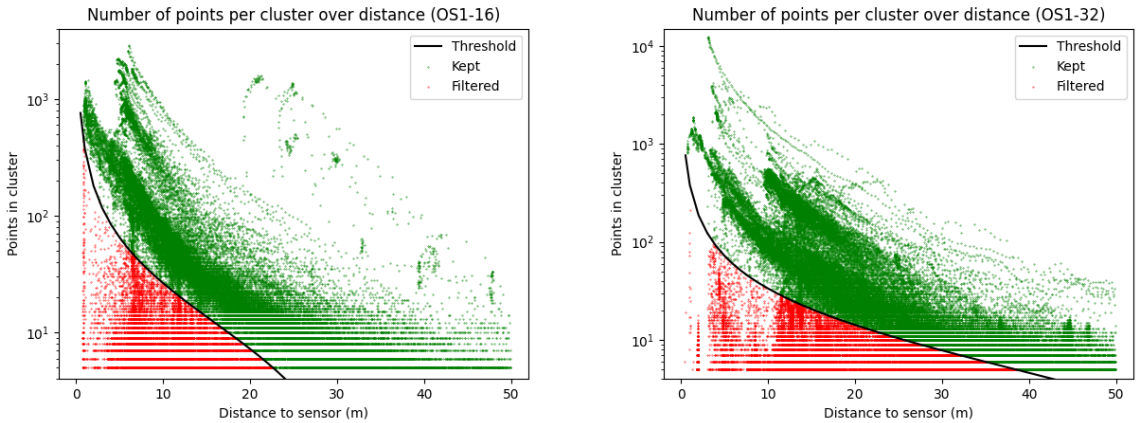
(a) OS1-16 *(b)* OS1-32

Figure 6.7: Scatter plot of the number of points in each cluster against the distance of the cluster's centroid to the LiDAR sensor.



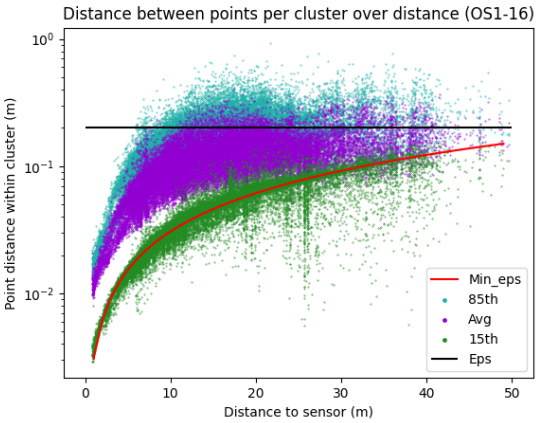
(a) OS1-16 *(b)* OS1-32

Figure 6.8: Scatter plot of the number of points in each cluster of 3 pedestrians against the distance of each cluster's centroid to the LiDAR sensor.

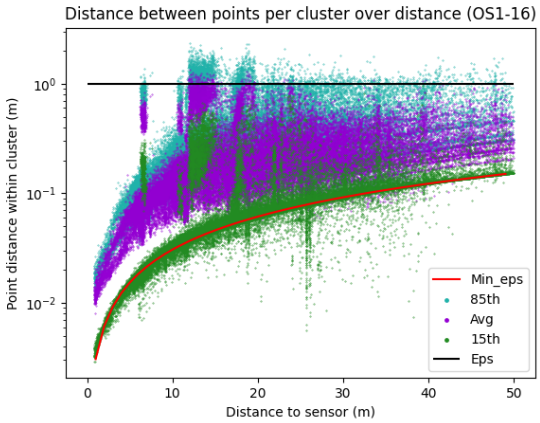


(a) OS1-16 *(b)* OS1-32

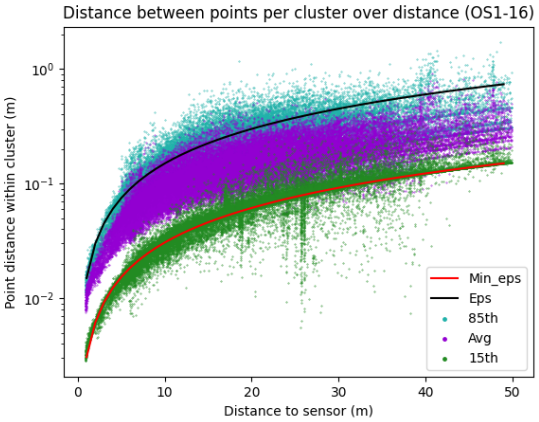
Figure 6.9: Scatter plot of the number of points in each cluster against the distance of the cluster's centroid to the LiDAR sensor, including a filter. The black line indicates the division, green points are kept, and red points are filtered out.



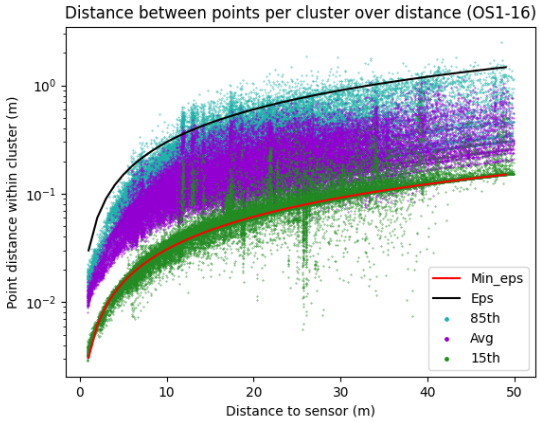
(a) $\epsilon = 0.2$



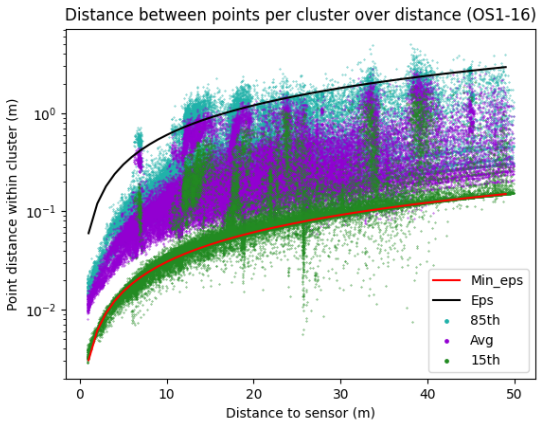
(b) $\epsilon = 1.0$



(c) $\epsilon = 0.015 \cdot d$



(d) $\epsilon = 0.03 \cdot d$



(e) $\epsilon = 0.06 \cdot d$

Figure 6.10: Scatter plot of the distance between points within a cluster. The black line shows the function of ϵ and the red line shows the minimum distance between horizontally adjacent LiDAR angles.

Table 6.6: Comparison between the clustering performance of three functions for ϵ using table 5.2.

ϵ	Ground truth	$0.015 \cdot d$	$0.030 \cdot d$	$0.060 \cdot d$
Clustered points	5387	5055	5276	5374
Clustered objects	46	44	40	37
Missing objects	0	5	5	5
Merged objects	0	0	1	6
Seperated objects	0	3	0	0

Table 6.7: Comparison between how many objects passed through the FOV of the LiDAR and how many objects are identified and tracked using data processing.

Data set	**Expected**	**Tracked**
Eikstraat	90	166
Kettingbrugweg	55	85
Holterbergweg	87	78
Torenstraat	279	527

The three functions for ϵ presented are tested against the eight manually labelled frames from table 5.2. The results are shown in table 6.6 and display expected behaviour in that a too small ϵ results in cluster separation and a too large ϵ in cluster merging. Notable is that all three functions have five missing objects, which corresponds to five objects that contain fewer than five points and thus fewer than the lower limit of *MinPts*. This makes $\epsilon = 0.03 \cdot d$ the preferred function and is used for the remainder of the evaluation.

6.3 Tracking

Given ideal circumstances the tracking performs as expected, however, the imperfect background removal and clustering introduce additional tracked objects. The results of the number of tracked objects are shown in table 6.7 and in most cases the number of tracked objects is significantly higher than expected. Visual examination reveals three situations that account for a significant part of the increase in the number of tracked objects: incorrect clustering, distance to the LiDAR sensor, and occlusion.

The two ways that incorrect clustering results in an increase in the number of tracked objects are the separation of a single object into multiple objects and the

(a) Incorrect clustering *(b)* Double tracking

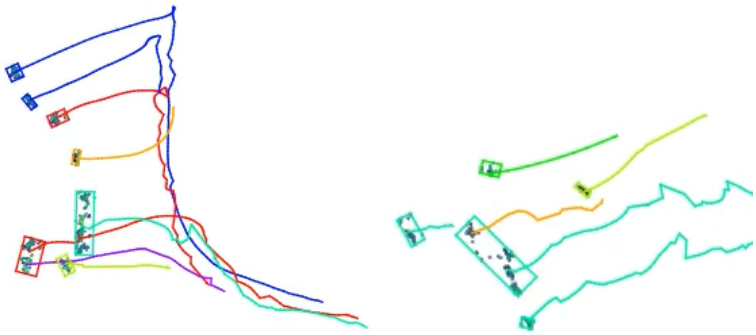
Figure 6.11: Incorrect clustering of objects (6.11a) can lead to two tracks representing the same object (6.11b).

grouping of multiple objects into a single cluster. Whenever an object is split into multiple clusters an additional possible association is created. Whenever the object stays separated, multiple objects are tracked. If it happens only once, the tracking algorithm can alternate between the two created tracked objects as an object that disappears is not immediately removed. Resulting in an individual object represented by two tracks as shown in figure 6.11. When multiple objects are grouped the track for one of the objects can be lost, and when they are separated again the object is considered a new object.

Objects both too close or too far away from the LiDAR sensor can also result in an increased number of tracked objects as they cannot be detected. The LiDAR sensors have a so-called dead zone close to the sensor in which no measurements can be done. Objects that stay within the dead zone too long are registered as new tracked objects when they are observed again. Objects that move at the edge of the LiDAR's FOV are most difficult to identify due to sparsity and may not be consistently detected and thus incorrectly tracked.

Occlusion is a well-known issue within the area of computer vision and also applies to LiDAR-based data by covering (part of) an object. Occlusion can result in both incorrect clustering and lost tracks. When an object is partly occluded, e.g. passing a road sign, it can be split into multiple clusters representing the same object. When an object is completely occluded, e.g. passing behind a truck, the object's track can be lost and is considered a new object when coming into view again.

Most objects are tracked correctly, but the increase in the number of observed objects is largely due to groups of objects that are incorrectly tracked. An example of such a situation is shown in figure 6.12, where a group of pedestrians move tightly packed. Incorrect clustering, occlusion, and incorrect association all occur within this group, rapidly increasing the number of observed objects. Due to occlusion



(a) *(b)*

Figure 6.12: Incorrect clustering of a group of pedestrians creating multiple short tracks. Both images show the same group of pedestrians at two different moments in time.

fewer pedestrians are visible in 6.12b and in both images multiple pedestrians are grouped.

6.4 Classification

To evaluate the classification the processed data is visualised and manually checked. The combined results are shown in table 6.8. Tracked objects that did not represent traffic and grouped objects with multiple classes were ignored. Grouped objects of the same class were kept and labelled as their corresponding class. In total 59 objects are ignored, which is 6.89%. Anecdotal evidence shows that object tracks that keep a relatively large distance to the LiDAR sensor are often classified incorrectly.

The weighted precision of the classifier for distinguishing between pedestrians, two-wheelers, and vehicles is 81.30%. However, when considering the goal of orientation estimation only the classification between pedestrians/two-wheelers and vehicles is relevant. Table 6.9 shows the results when splitting the object classes between PCA and MinErrorRect as the orientation estimation method. In this case the weighted average precision is 93.35%.

Table 6.8: Classification results of the evaluation using all data sets.

Combined		**Classified as**			
		Pedestrian	Two-wheeler	Vehicle	Recall
Class	Pedestrian	**257**	42	9	83.44%
	Two-wheeler	54	**159**	7	72.27%
	Vehicle	11	26	**232**	86.25%
	Precision	79.81%	70.04%	93.55%	

Table 6.9: Results of the evaluation of the applied method for the dimension estimation.

Combined		**Applied method**		
		PCA	MinErrorRect	Recall
Expected	PCA	**512**	16	96.97%
method	MinErrorRect	37	**232**	86.25%
	Precision	93.26%	93.55%	

6.5 Orientation estimation

Preceding the results o the correlation between an object's orientation and next location, is the validation of the selection of the bounding box side representing the object's direction. As mentioned in the method, the selection is correct if all values fall within $\pm 45^{\circ}$. The result is shown in figure 6.13. The results are separated per bounding box methods per object class for later comparison. All measurements do fall within the expected $\pm 45^{\circ}$ range.

It is expected that both PCA and MinErrorRect outperform Rotating calipers in all situations. Based on the initial results of figure 3.3 PCA is expected to perform well on two-wheelers and MinErrorRect on vehicles. For pedestrians none of the methods is expected to perform well. However, as pedestrians do have some consistency in shape PCA is expected to align slightly better than the others. Reasoning for these hypotheses can be found in section 3.3.1.

The results of testing the orientation estimation against an object's next location are shown figure 6.14. Each histogram shows the distribution of angle difference between the orientation and next location split between five distance segments. The mean and median of the absolute angle differences are visualised in figure 6.14b and noted in table 6.10. The more the distribution in figure 6.14b is skewed to the right, the better the orientation corresponds with the next location.

For pedestrians all three bounding box methods performed similar with a relatively high mean and median compared to the other object classes. This is in line with the results in figure 6.13, where the variation of the alignment between the previous location and the direction is high. PCA does perform best when comparing mean and median values. This is also confirmed using one-tailed paired t-tests:

- **Rot Calp - PCA**: With $t(49087) = 15.24$ and $p < 1.00 \cdot 10^{-5}$ the average angle difference of Rotating calipers is statistically larger than PCA.
- **PCA - MinErrorRect**: With $t(49087) = -16.50$ and $p < 1.00 \cdot 10^{-5}$ the average angle difference of PCA is statistically smaller than MinErrorRect.
- **Rot Calp - MinErrorRect**: With $t(49087) = -1.32$ and $p = 9.40 \cdot 10^{-2}$ the average angle difference of Rotating calipers is not statistically smaller than MinErrorRect.

For two-wheelers PCA performed on average almost 2° better than both Rotating calipers and MinErrorRect. Again, the differences between the methods are tested:

- **Rot Calp - PCA**: With $t(23336) = 29.73$ and $p < 1.00 \cdot 10^{-5}$ the average angle difference of Rotating calipers is statistically larger than PCA.
- **PCA - MinErrorRect**: With $t(23336) = -28.63$ and $p < 1.00 \cdot 10^{-5}$ the average angle difference of PCA is statistically smaller than MinErrorRect.
- **Rot Calp - MinErrorRect**: With $t(23336) = -1.42$ and $p = 1.56 \cdot 10^{-1}$ the average angle difference of Rotating calipers is not statistically smaller than MinErrorRect.

These results again show that PCA is a significant improvement over both Rotating calipers and MinErrorRect. Both the lower mean and median, and the larger difference between the mean and median substantiate the improvement.

Finally, for vehicles MinErrorRect performs best and shows an improvement of almost 5° on average. Paired t-tests results:

- **Rot Calp - PCA**: With $t(21820) = 32.43$ and $p < 1.00 \cdot 10^{-5}$ the average angle difference of Rotating calipers is statistically larger than PCA.
- **PCA - MinErrorRect**: With $t(21820) = 47.53$ and $p < 1.00 \cdot 10^{-5}$ the average angle difference of PCA is statistically larger than MinErrorRect.
- **Rot Calp - MinErrorRect**: With $t(21820) = 64.86$ and $p < 1.00 \cdot 10^{-5}$ the average angle difference of Rotating calipers is statistically larger than MinErrorRect.

As expected, MinErrorRect performs best on vehicles and shows the largest improvement over the other methods between object classes. Especially the low median reflects how right-skewed the absolute histogram is. Also here, the lower mean and median, and the larger difference between the mean and median substantiate the improvement.

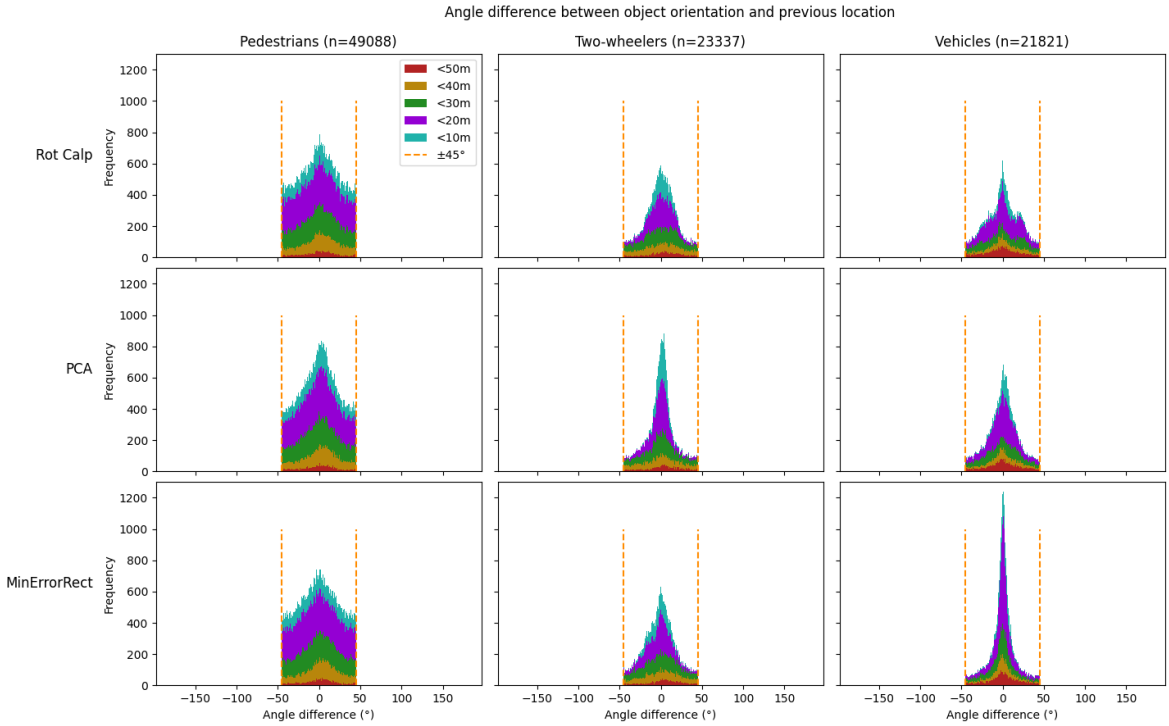
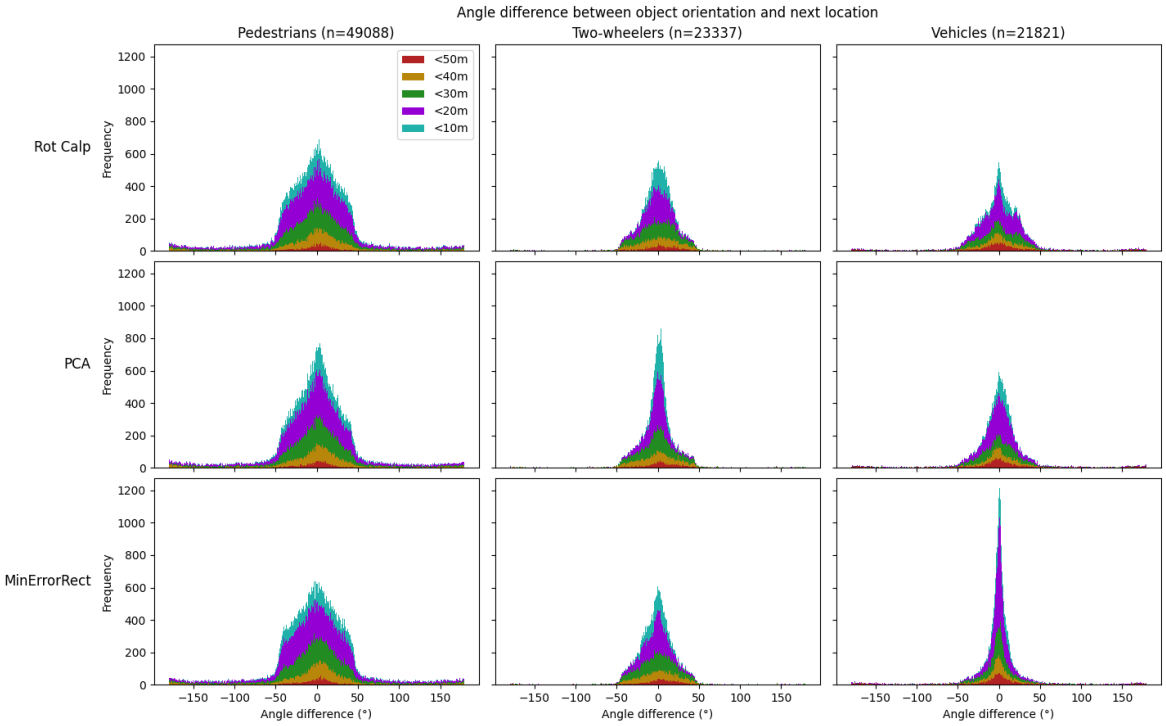


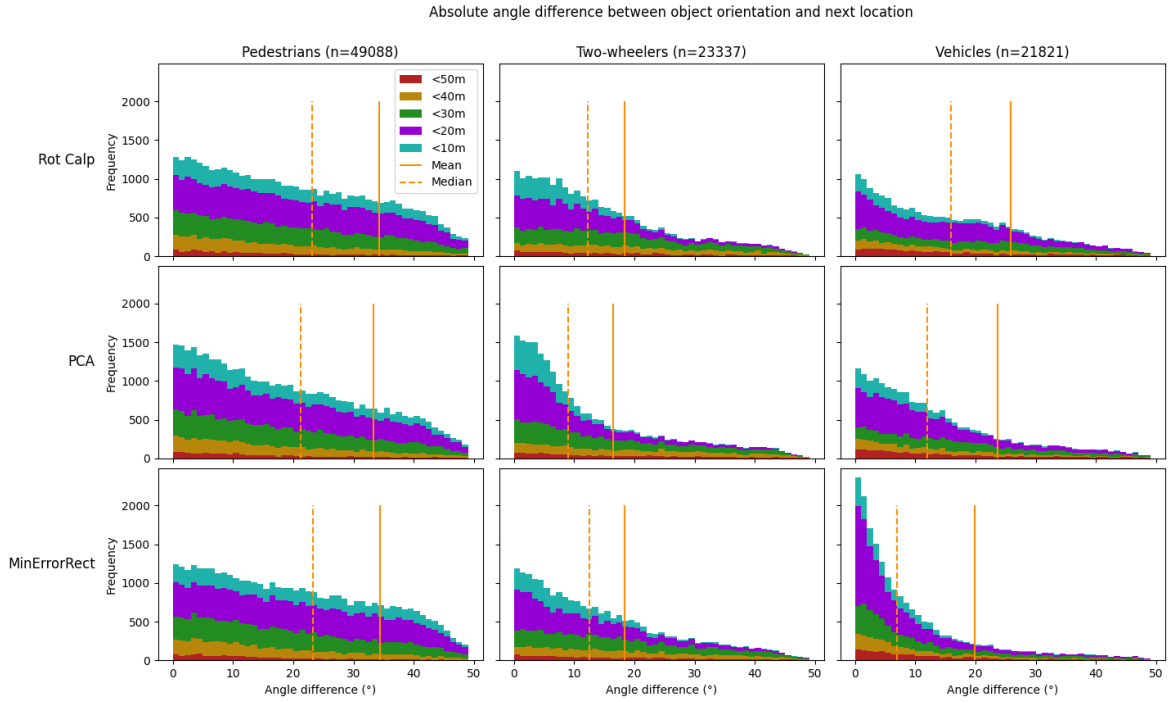
Figure 6.13: Histograms of the angle difference between the selected side of an object's bounding box determining its direction and its previous location.

Table 6.10: Mean and median values of the absolute angle difference between the object orientation and the next location per bounding box method per object class between 0 and 180°.

Class	**Pedestrian**		**Two-wheeler**		**Vehicle**	
Metric	Mean	Median	Mean	Median	Mean	Median
Rot Calp	34.31	23.07	18.33	12.30	25.82	15.88
PCA	33.26	21.17	16.50	9.00	23.69	11.96
MinErrorRect	34.40	23.19	18.43	12.57	19.90	6.93



(a) All values



(b) Absolute values between 0 - 50°

Figure 6.14: Overview of histograms that show the angle difference between the estimated orientation and the next location of an object using multiple bounding box methods and object classes. The results per method/class combination are split into five equal ranges between 0 and 50m.

Chapter 7

Discussion

In this chapter the results are discussed and how they provide an answer to the overall goal and research questions. The discussion chapter is divided into *system validation*, *background identification & removal*, and *orientation estimation*.

7.1 System validation

To answer the main research question on autonomous object detection and tracking using LiDAR a full data processing pipeline is required, resulting in the main project goal. Using knowledge and experience from existing work and by developing new methods a system is designed and implemented that processes point clouds into tracked objects. Whether this system is sufficient for answering the main- and sub research questions is determined using validation tests for each processing step. All processing steps are discussed apart from *background identification & removal* and *orientation estimation* as they are discussed in-depth later in this chapter.

7.1.1 Clustering

The implemented DBSCAN method including a dynamic function for both *MinPts* and ϵ functions correctly and is capable of clustering points that represent objects and discarding noise. First, the alignment of the *MinPts* functions function as expected as shown in figure 6.9. All clusters below the threshold are filtered. Due to limited time the functions are aligned based on a limited number of manually selected pedestrians and also include a margin to prevent unwanted exclusion of clusters. The margin is also included in order to prevent partially occluded objects from being filtered out. More testing is required in order to get a better fitting *MinPts* function for both LiDARs.

The calibration of ϵ shows that using a distance-dependant function results in an improvement over using a static value. Because the point cloud is projected parallel to the ground plane before clustering it, the distance between points could have been decreased enough for a static ϵ to be sufficient. However, figure 6.10 indicates that this is not the case. First, a too low static ϵ cuts the usable range with respect to the LiDAR sensor (figure 6.10a). Second, the hypothesis is that a too large ϵ merges or creates new clusters incorrectly, as displayed in figures 6.10b and 6.10e. This hypothesis is backed by the results from table 6.6 where the largest function for ϵ results in merged clusters. Table 6.6 also provides evidence that using a dynamic function results in improvements as $\epsilon = 0.03 \cdot d$ results in a limited amount of merged clusters while not limiting the usable range.

From the three tested functions for ϵ the best performing is $\epsilon = 0.03 \cdot d$ based on the results of table 6.6. The smallest value for $\epsilon = 0.015 \cdot d$ resulted in the least amount of remaining points and cars being split into multiple objects. $\epsilon = 0.06 \cdot d$ resulted in the most remaining points, however, did cluster groups of objects. $\epsilon = 0.03 \cdot d$ strikes a balance between point retention and correct clustering. Later testing does show that this function for ϵ still struggles with groups as shown by figure 6.12. The results of using DBSCAN in combination with dynamic functions for *MinPts* and ϵ are considered sufficient for further use.

7.1.2 Tracking

The results of the tracking method (section 6.3) show that in an ideal situation it is capable of correctly tracking an object, however, in more challenging situations it often results in an increased number of objects. For the evaluation of the orientation estimation it is not a problem if objects are split into multiple tracks, as long as the tracking itself and classification are correct. It is also better to have more objects than expected instead of too few, because this means most objects are detected and need better association instead of objects not being detected. As the object classification is dependent on all dimension estimations of an object, split tracks can also lead to incorrect classification and in turn incorrect orientation estimation. The test results of the classification already showed this not to be a problem, as is discussed in the next section.

The lack of rotational velocity in the Kalman filter increases the difficulty of tracking grouped objects. The prediction for an object's next location is based on the continuation in a straight line from the last known location. When a group of objects starts turning this often results in incorrect association when the path of object *A* crosses the predicted path of object *B*.

The tracking is considered sufficient for the evaluation of the orientation estimation, however, for automatic traffic analysis anomalies can be problematic. As the path of objects is tracked, may it not be in one go, and the correct bounding box method can be applied based on the observed clusters within most tracks, the tracking does suffice for object orientation estimation. Whenever the path of an object is represented by multiple tracks it becomes more difficult to analyse the full path of that object without manual input.

7.1.3 Classification

The implemented classification method performs reasonably well for distinguishing between object classes and is sufficient for evaluating the orientation estimation. Due to time limitations the relatively basic classification method is mostly aimed at the evaluation of the orientation estimation. One of the requirements is to differentiate between pedestrians, two-wheelers, and vehicle and thus this performance is also discussed.

Anecdotal evidence shows that objects that are split in multiple tracks are often incorrectly classified due to the distance to the sensor. Because of the distance no accurate dimension estimation can be made and thus no correct classification. Especially the vertical resolution is a limiting factor at range, conflicting with classification of pedestrians which is fully based on the height of an object. This explains why pedestrians are often classified as two-wheelers at range. Improvement of the tracking algorithm will likely lead to improved classification.

The classification of two-wheelers is always a challenging situation given its dimensions and the used method. On city bikes where the rider is sitting straight up the difference between the length and the height of bicycle, including rider, is small. This can lead to two-wheelers being classified as pedestrians, especially when the bicycle is always observed from an angle. The resolution of the LiDAR does remedy this problem somewhat, because the vertical resolution is significantly lower than the horizontal, meaning it is more likely to observe the full length of the bike over the full height.

Combining the results of table 6.8 and the anecdotal evidence that distant objects are often incorrectly classified due to split tracks, the classification method performs as expected. For determining whether to apply PCA or MinErrorRect the classification works well at a precision of 93.35%, as most incorrect classifications are between pedestrians and two-wheelers.

7.2 Background identification & removal

The results in section 6.1 show that the newly introduced PD method has a significant computational advantage over 3D-DSF at an insignificant performance loss. The PD method originated by trying to improve TSC as the potential processing speed was promising, but initial results were unreliable. RA is determined unusable in combination with the used hardware and not discussed further.

TSC was not expected to outperform either PD or 3D-DSF and it did not. The most significant problem with TSC is that it is heavily affected by passing traffic and dynamic background. Both situations increase the standard deviation significantly, often placing the threshold too close to the sensor and filtering out relevant data.

Both PD and 3D-DSF are capable of filtering out passing traffic, but struggle with traffic that is stationary for extended periods of time. In frame 3748 of the *Torenstraat* data set a truck is present, which is unloading for the majority of the recording. Both BRMs filter out a significant number of points associated to the truck and also all other objects moving through the space when there is no truck. This is also a situation where the difference in dimensionality shows, as 3D-DSF labels the space occluded by the truck as foreground and PD labels it background. There is no clear advantage to either situation as PD removes relevant info, while 3D-DSF may keep irrelevant data.

The use of a a two-dimensional background model has a clear computational benefit over a three-dimensional model when using a rotational LiDAR, without significant performance loss. This improvement is not only a result of the reduced dimensionality, but also of not having to convert coordinates from spherical to Cartesian. The number of entries in the background model of 3D-DSF is potentially 300 times larger than that of PD, which does slow down both background model generation and -removal. This also means that the 3D-DSF model is more detailed and may perform better in more specialised situations.

Between PD and 3D-DSF it was expected that the former would outperform the latter in terms of processing speed, while the question remained if and, if so, by how much the background identification performance would suffer. In section 6.1.6 is determined that the performance difference between the BRMs is not significant and the speed increase of PD over 3D-DSF is significant in all scenarios. With an average processing speed of 14ms against 112ms for PD and 3D-DSF using the OS1-16 respectively, PD leaves significantly more room for further processing at a LiDAR output of 10Hz. Therefore, PD with an RPP of 15% is chosen to be used for the implementation.

7.3 Orientation estimation

The results in figure 6.14 and table 6.10 show that for all object classes different bounding box methods align better to an object's next location. This confirms what was expected based on the results of figure 3.3.

For pedestrians none of the bounding box methods showed a clear improvement initially, however, PCA is determined to be statistically better. Even though the difference is significant, the alignment error is still almost double that of the two-wheelers. Meaning that the direction based on the estimated orientation may still be very unreliable.

The overall poor orientation estimation results for pedestrians is expected to be caused by their continuously changing shape and 360° mobility. When viewed from above, a walking pedestrian is moving arms and legs in an alternating motion. A stationary pedestrian with its arms along its sides is often wider than it is long, while it is the other way around when moving. Due to the ever changing shape it is difficult to find one bounding box method that aligns best with all potential shapes. The second problem is that pedestrians can move in any direction, may it be forwards, backwards, or sideways. By not being able to consistently estimate the orientation of a pedestrian and knowing the actual orientation does not necessarily represent its direction, the results are explainable.

The improvement of PCA over MinErrorRect when applied to two-wheelers is statistically significant and shows from the shape of the histogram in figure 6.14b. It is more right-skewed than the other two. Numerically the mean and median are 10% and 27% lower respectively than the second best method Rotating calipers. Meaning a larger amount of measurements have a small angle difference and thus align better with an object's next location.

A similar difference between methods is found for vehicles, with in this case MinErrorRect being the better performer. It shows a statistically significant improvement over both other methods. Compared to PCA, MinErrorRect has a 16% lower mean and 42% lower median.

Interesting it that MinErrorRect applied to vehicles has a higher mean and lower median than PCA applied to two-wheelers. This is visible in figure 6.14b as MinErrorRect has an almost exponential decrease in number of measurements per angle difference. Looking closely at the histograms of both PCA and MinErrorRect in 6.14a shows that MinErrorRect has substantially more measurements outside $\pm 50°$, probably explaining the larger mean.

A possible cause of large angle differences for vehicles with is noise. If enough noise passes the background removal in roughly the same location it can result

in a tracked object. This false object moves randomly and thus results in large angle differences. These false objects have been observed in the processed data and were often at large range or around very dynamic backgrounds like bushes. A case can also be made for why more noise is labelled as vehicle than two-wheeler. Because the movement of the noise is random, the dimension estimation does also not work properly. This can easily result in an object wider than 1.5m, acquiring the vehicle label.

The results of section 6.5 prove that using class-dependant bounding box methods improves the ability to estimate the orientation of all object types. The shape of pedestrians is too inconsistent and the mobility too unpredictable for any of the used methods to be as reliable as for the other classes. For two-wheelers and vehicles PCA and MinErrorRect align more consistent than other methods respectively.

Chapter 8

Conclusion

The presented research meets all of its goals and provides answers to the research questions stated in the introduction. A completely autonomous data processing pipeline is built that extracts information on a per object basis from a sequence of point clouds. A new background identification and removal method is developed that significantly improves the processing speed. And, it is shown to be an improvement to use different bounding box methods per object class to estimate the orientation.

Each object is detected, tracked, and classified, and for each object the location, velocity, and direction are known within each frame. From the processed data a video can be generated which shows the point cluster of each object surrounded by a bounding box and a trail is drawn based on the travelled path. The object data allows for numerous types of analysis like heat maps, directionality maps, per class velocities, et cetera.

The validation of each part of the data processing pipeline shows that the overall performance is sufficient for answering all RQs. The background of each frame is removed with an average precision and recall of 67% and 93% respectively. The parameters of the clustering method are dynamically altered based on the distance to the LiDAR, resulting in consistent object detection throughout the LiDAR's FOV. However, closely grouped objects and occlusion do result in incorrect clustering. Given an ideal situation the tracking method performs as intended, but with errors in preceding processing steps objects are often split into multiple tracks. For the classification of objects a manually defined decision tree was able to correctly classify object 81% of the time, while applying the correct bounding box method in 93% of the cases.

During the implementation of the data processing pipeline a new Background Removal Method (BRM) is developed called Peak Detection (PD). Four BRMs are compared to answer the sub question whether a two-dimensional (2D) background model can improve performance over a three-dimensional (3D) model. Two meth-

ods generating a 2D model and two generating a 3D model are compared. Results show that using PD, a 2D model, does not reduce detection and removal performance significantly compared against state of the art 3D methods. On average PD does increase model generation and removal speed with 44% and 88% against 3D models respectively. Thus, 2D background identification and -removal methods do outperform 3D alternatives given a rotational LiDAR.

Based on an initial comparison of bounding box methods it was hypothesised that selecting a method on a per object class basis would improve orientation estimation performance. Using almost 100,000 measurements it is proven that applying a PCA-based bounding box significantly improves alignment with both pedestrians' and two-wheelers' next location, while MinErrorRect aligns better with vehicles. Thus, confirming that selecting a bounding box method based on the object class for the orientation estimation can improve performance.

Chapter 9

Future work

In this chapter recommendations are made for points of improvement in future work. The chapter is divided into three sections. The first section describes a number of improvements that would have been implemented given more time. These are not discussed in detail as they are already discussed earlier in this research. The second section presents multiple directions for improving the PD background detection and -removal method. And the final section discusses the potential improvement of clustering based on an object's expected dimensions.

9.1 General improvements

There are multiple adjustments that are recommended for evaluation as they are expected to improve overall performance.

Clustering Presented in section 3.2.2.2 is a clustering method called DTSCAN. The related work chapter already discusses that DTSCAN is expected to be an improvement over DBSCAN, however, due to time constraints it has not been implemented. The ability to separate merged objects due to bridging points will help separate grouped objects like in figure 6.12b.

Tracking Even though Kalman filter tracking is a capable tracking method, some situations have been found during the evaluation indicating potential performance gains. The exclusive association of existing objects with newly detected objects makes association in groups difficult due to occlusion, especially closely located groups like pedestrians. The addition of probability, like the Joint Probabilistic Data Association can help in these situations. Another potential improvement for the

tracking method is the addition of an Interactive Multiple Model method, that can better predict an object's future location using multiple motion modes.

Classification For the initial implementation of the data processing pipeline a manually defined classifier proved to be usable. However, now that classified data is available classification models can be trained. Especially between the limited number of classes defined in this research a higher classification precision is achievable. A second improvement would be to introduce more specified object classes for more detailed object information.

Hardware Tuning the vertical FOV of the LiDAR will increase the number of usable points and thus the number of points representing objects. As with all technology, LiDAR sensors also improve over time and increase their vertical resolution. However, in case of the used OS1-XX sensors, they both have a vertical FOV that aims equally upwards as downwards. At the height the sensor is mounted on the pole and car the upward facing LiDAR angles are often measuring outside the area of interest. Lowering the overall vertical FOV by reducing the upwards facing angle will increase vertical point density in the area of interest without significantly affecting usability.

9.2 Improving PD

The PD background identification and removal method presented in this research already performs well, but there are ideas on how to improve it further. In this section multiple ideas are discussed which may lead to increased detection and removal performance.

9.2.1 Peak selection

The implementation of PD (section 4.3.1.2) states that whenever multiple peaks between the RPP and 50% are found, the last peak is chosen as background. This choice is made based on the idea that no detection is possible behind a background object. However, this is not true in case of dynamic background. In this research no testing is done to evaluate the effect of choosing the last peak, potentially leading to unfiltered background points. Therefore, it is recommended to investigate whether the increase in noise is worth the potential increase in recall.

9.2.2 Multiple peak model

The aforementioned choice on which peak should be selected if multiple peaks are present can be resolved by allowing multiple peaks to be selected. Using multiple range threshold per LiDAR angle does change PD from a two-dimensional into a three-dimensional model. However, it is expected that this will still result in a significantly smaller model than e.g. 3D-DSF. With a multiple peak model all peaks that are larger than the RPP are selected as background, and the background model contains sets of exclusion ranges instead of a maximum detection range.

Using the multiple peak model allows for the detection of objects between and beyond ranges identified as background. If a flag is positioned in front of a house, the space between them should still be usable when the flag is not occluding it. This is one of the advantages the implemented three-dimensional methods have over PD.

9.2.3 Peak characteristics

In the current implementation of PD only the sum of measurements of each detected peak is used to label it, even though more information is available. This section notes two more characteristics that could be used to determine whether a peak represents background or foreground. Taking it a step further, the characteristics can also be combined and having a trained classifier determine what peaks are background.

Peak shape When observing the histogram of a single LiDAR angle after aggregating the frames used to determine the background, all kinds of peak shapes are shown. The wall of a building will generate a tall and narrow peak due to the consistency of the measurements, while a road will generate a lower and wider peak due to the varying positions of traffic. Understanding the causation of peak shapes can help classifying which are background.

Measurement intensity Each measurement made by the LiDAR also includes the intensity of the light that is reflected back. E.g. road signs are manufactured to have high reflectivity and reflect more light than a curtain made for blocking out the light. It is difficult to differentiate background from foreground based on the intensity value alone, however, the consistency can provide more information. A stationary object is likely to have a consistent intensity value, while passing traffic will vary because of different materials and colors.

9.2.4 Real-time model adjustments

Using a recording-based background model in a real-time situation introduces the problem of not being able to adapt to a changing environment. Parked vehicles or weather conditions like snow can deviate the generated model from the physical situation. By utilising the ever increasing number of processing cores available on modern processing units, the background model can be updated in parallel to the background removal process.

One implementation is that for every X minutes a percentage of frames is aggregated and compared against the background model. If during the initial background generation a vehicle is parked, the background model will exclude all measurements at the vehicle and beyond. If at a later time the vehicle moves and the newly collected frames do not show any sign of the vehicle, the model may be updated to find a new background threshold.

9.3 Dimensions-based clustering

An issue that remains, no matter how well the clustering parameters are tuned, is the incorrect merging and splitting of object clusters. Objects can be split due to occluding road signage or other traffic, buses show interior clusters due to large windows, et cetera. Similar processing steps are already researched and implemented in related work. Tarko et al. [15] implemented an iteration process, not unlike the one presented in this section, and loop over the clusters until the number of objects corresponds to the number of tracked objects.

The clustering performance can be improved by introducing a dimension check on objects that have already been tracked for multiple frames. Whenever multiple clusters fit within the dimensions (and orientation) of a tracked object, it is likely they belong to the same vehicle. By aligning a bounding box using the expected dimensions of an object any corresponding clusters are expected to fall within it. This will in turn improve the tracking performance as less confusion is created by suddenly emerging new clusters.

The expected dimensions bounding box can also be used to split any merging objects. If two cyclists ride side by side, their steering wheels may form a bridging point that results in one large cluster. Projecting an expected bounding box on the large cluster shows a too large object. This can be solved by either re-clustering the points with finer parameters, or allowing multiple active tracks to be associated with the cluster so that the tracking continues correctly.

Bibliography

[1] "Verkeersdoden in Nederland. SWOV-Factsheet, april 2020. SWOV, Den Haag." [Online]. Available: https://www.swov.nl/feiten-cijfers/factsheet/verkeersdoden-nederland

[2] "Ernstig verkeersgewonden in Nederland. SWOV-factsheet, december 2019, SWOV Den Haag." [Online]. Available: https://www.swov.nl/feiten-cijfers/factsheet/ernstig-verkeersgewonden-nederland

[3] "Kosten van verkeersongevallen. SWOV-factsheet, maart 2020, SWOV Den Haag." [Online]. Available: https://www.swov.nl/feiten-cijfers/factsheet/kosten-van-verkeersongevallen

[4] CBS, "11 procent meer verkeersdoden in 2018," 2019. [Online]. Available: https://www.cbs.nl/nl-nl/nieuws/2019/16/11-procent-meer-verkeersdoden-in-2018

[5] M. Reurings, W. Vlakveld, D. Twisk, A. Dijkstra, W. Wijnen, B. Reurings, P. Vlakveld, M. Twisk, A. Dijkstra, and W. Wijnen, "Van fietsongeval naar maatregelen: kennis en hiaten," p. 203, 2012.

[6] WHO, *Global status report on road safety 2018*, 2018.

[7] C.-H. Wei, C.-T. Chen, J.-C. Chen, and S.-T. Wu, "Using ground-based lidar data to measure standing trees in a red cypress plantation," *Taiwan Journal of Forest Science*, vol. 29, pp. 169–178, 09 2014.

[8] Federal Highway Administration, "Traffic Monitoring Guide FHWA," *Fhwa*, no. October, p. 462, 2016. [Online]. Available: http://www.fhwa.dot.gov/policyinformation/tmguide/

[9] F. Heintz, P. Rudol, and P. Doherty, "From images to traffic behavior - A UAV tracking and monitoring application," *FUSION 2007 - 2007 10th International Conference on Information Fusion*, no. June 2014, 2007.

[10] L. Wang, F. Chen, and H. Yin, "Detecting and tracking vehicles in traffic by unmanned aerial vehicles," *Automation in Construction*, vol. 72, no. May, pp. 294–308, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.autcon.2016.05.008

[11] G. Lira, Z. Kokkinogenis, R. J. Rossetti, D. C. Moura, and T. Rúbio, "A computer-vision approach to traffic analysis over intersections," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 47–53, 2016.

[12] T. Akita and S. Mita, "Object Tracking and Classification Using Millimeter-Wave Radar Based on LSTM," *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*, pp. 1110–1115, 2019.

[13] P. Zhao, C. X. Lu, J. Wang, C. Chen, W. Wang, N. Trigoni, and A. Markham, "MID: Tracking and identifying people with millimeter wave radar," *Proceedings - 15th Annual International Conference on Distributed Computing in Sensor Systems, DCOSS 2019*, pp. 33–40, 2019.

[14] J. Ryde and N. Hillier, "Performance of laser and radar ranging devices in adverse environmental conditions," *Journal of Field Robotics*, vol. 26, no. 9, pp. 712–727, 2009.

[15] A. P. Tarko, K. B. Ariyur, M. A. Romero, V. K. Bandaru, and C. G. Lizarazo, *JOINT TRANSPORTATION TScan : Stationary LiDAR for Traffic and Safety Studies — Object Detection and Tracking*, 2016.

[16] J. Wu, "An Automatic Procedure for Vehicle Tracking with a Roadside LiDAR Sensor," *ITE journal*, pp. 32–37, 2018.

[17] J. Zhao, H. Xu, H. Liu, J. Wu, Y. Zheng, and D. Wu, "Detection and tracking of pedestrians and vehicles using roadside LiDAR sensors," *Transportation Research Part C: Emerging Technologies*, vol. 100, no. December 2017, pp. 68–87, 2019. [Online]. Available: https://doi.org/10.1016/j.trc.2019.01.007

[18] Y. Cui, H. Xu, J. Wu, Y. Sun, and J. Zhao, "Automatic Vehicle Tracking with Roadside LiDAR Data for the Connected-Vehicles System," *IEEE Intelligent Systems*, vol. PP, no. c, p. 1, 2019.

[19] J. Chen, H. Xu, J. Wu, R. Yue, C. Yuan, and L. Wang, "Deer Crossing Road Detection with Roadside LiDAR Sensor," *IEEE Access*, vol. 7, pp. 65 944–65 954, 2019.

[20] Y. Zhou and O. Tuzel, “VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4490–4499, 2018.

[21] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, “Pv-Rcnn,” *Cvpr*, pp. 10 529–10 538, 2020. [Online]. Available: http://arxiv.org/abs/1912.13192

[22] C. He, H. Zeng, J. Huang, X.-s. Hua, and L. Zhang, “Sa-Ssd,” vol. 1, 2020.

[23] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, “Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks BT - Robotics and Automation (ICRA), 2017 IEEE International Conference on,” *IEEE Int. Conf. on Robotics and Autom. (ICRA)*, pp. 1355–1361, 2017.

[24] E. A. Hakim, “3D YOLO: End-to-End 3D Object Detection Using Point Clouds,” *Degree Project Computer Science and Engineering*, p. 47, 2018. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-234242

[25] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *International Journal of Robotics Research (IJRR)*, 2013.

[26] K. Kidono, T. Miyasaka, A. Watanabe, T. Naito, and J. Miura, “Pedestrian recognition using high-definition lidar,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 405–410.

[27] K. Liu, W. Wang, and J. Wang, “Pedestrian detection with lidar point clouds based on single template matching,” *Electronics (Switzerland)*, vol. 8, no. 7, 2019.

[28] Z. Y. Zhang, J. Zheng, X. Wang, and X. Fan, “Background Filtering and Vehicle Detection with Roadside Lidar Based on Point Association,” *Chinese Control Conference, CCC*, vol. 2018-July, pp. 7938–7943, 2018.

[29] C. Lin, H. Liu, D. Wu, and B. Gong, “Background Point Filtering of Low-Channel Infrastructure-Based LiDAR Data Using a Slice-Based Projection Filtering Algorithm,” *Sensors*, vol. 20, no. 11, p. 3054, may 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/11/3054

[30] J. Wu, H. Xu, Y. Sun, J. Zheng, and R. Yue, “Automatic Background Filtering Method for Roadside LiDAR Data,” *Transportation Research Record*, vol. 2672, no. 45, pp. 106–114, 2018.

[31] B. Lv, H. Xu, J. Wu, Y. Tian, and C. Yuan, “Raster-Based Background Filtering for Roadside LiDAR Data,” *IEEE Access*, vol. 7, pp. 76 779–76 788, 2019.

[32] J. Kim and J. Cho, "Delaunay triangulation-based spatial clustering technique for enhanced adjacent boundary detection and segmentation of lidar 3d point clouds," *Sensors (Switzerland)*, vol. 19, no. 18, 2019.

[33] M. Shamos, *Computational Geometry*, ser. THESIS (Ph.D) - YALE UNIVERSITY, 1978. Yale University, 1978. [Online]. Available: https://books.google.nl/books?id=Mf50SQAACAAJ

[34] S. Mishra, U. Sarkar, S. Taraphder, S. Datta, D. Swain, R. Saikhom, S. Panda, and M. Laishram, "Principal component analysis," *International Journal of Livestock Research*, p. 1, 01 2017.

[35] M. Yoshioka, N. Suganuma, K. Yoneda, and M. Aldibaja, "Real-time object classification for autonomous vehicle using LIDAR," *ICIIBMS 2017 - 2nd International Conference on Intelligent Informatics and Biomedical Sciences*, vol. 2018-January, pp. 210–211, 2018.

[36] "Lengte en gewicht van personen, ondergewicht en overgewicht; vanaf 1981." [Online]. Available: https://www.cbs.nl/nl-nl/cijfers/detail/81565NED?dl=2AA3B

[37] "Regeling voertuigen lengte voertuigen en aanhangwagens." [Online]. Available: https://auto-en-vervoer.infonu.nl/verkeer/78919-regeling-voertuigen-lengte-voertuigen-en-aanhangwagens.html

[38] "Average car length." [Online]. Available: https://anewwayforward.org/average-car-length/

[39] "Will a motorcycle fit in a cargo van." [Online]. Available: https://vandimensions.com/info/will-a-motorcycle-fit-in-a-cargo-van

[40] "Fiets afmetingen." [Online]. Available: https://www.verderfietsen.nl/fiets-afmetingen/

[41] "Average car height width and weight." [Online]. Available: https://bovagrai.info/auto/2019/en/2-registrations/2-5-average-car-height-width-and-weight/

[42] M. Potkány, M. Debnár, M. Hitka, and M. Gejdoš, "Requirements for the Internal Layout of Wooden House From the Point of View of Ergonomics Changes," *Quality Production Improvement*, vol. 09, no. September, pp. 43–70, 2018.

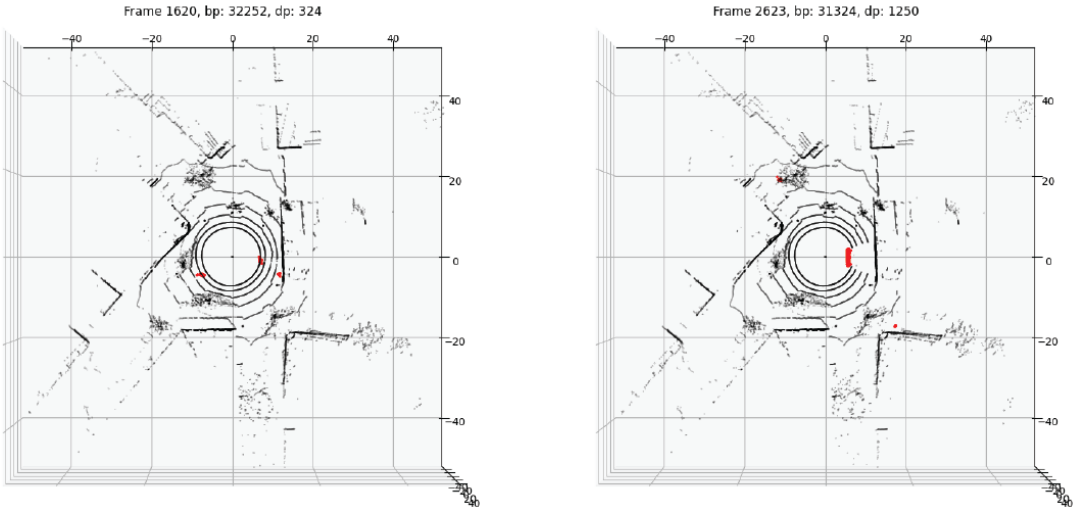
[43] D. Freedman and P. Diaconis, "On the histogram as a density estimator:L2 theory," *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, vol. 57, no. 4, pp. 453–476, 1981.

[44] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD’96. AAAI Press, 1996, p. 226–231.

[45] J. Wu, H. Xu, Y. Tian, R. Pi, and R. Yue, “Vehicle detection under adverse weather from roadside LiDAR data,” *Sensors (Switzerland)*, vol. 20, no. 12, pp. 1–17, 2020.

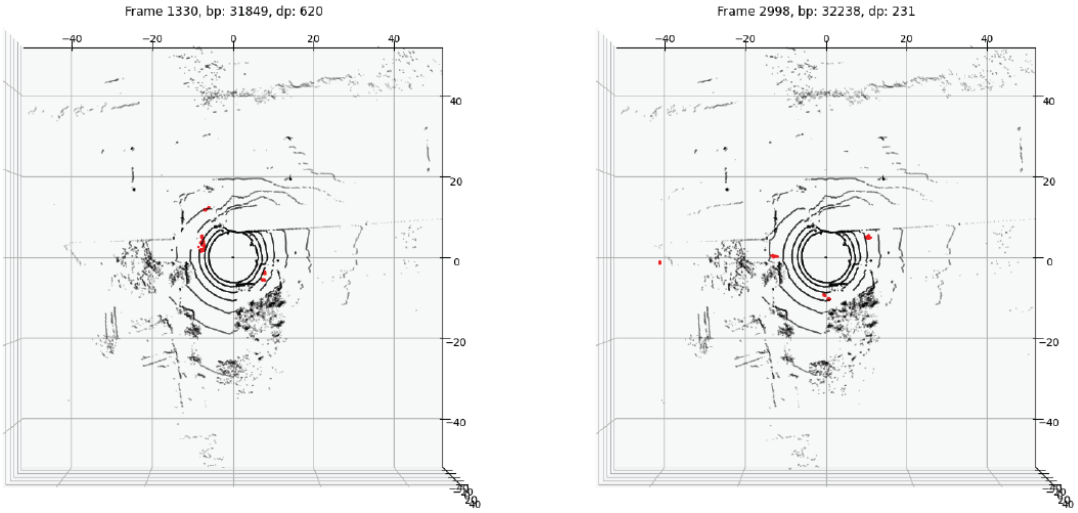
Appendix A

Evaluation frames images



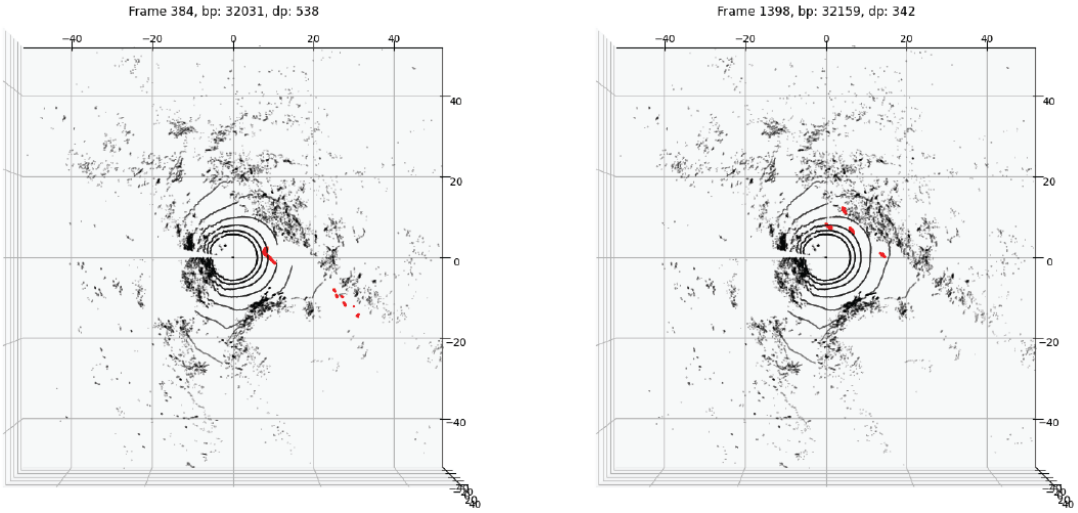
(a) *(b)*

Figure A.1: Evaluation frames for the *Eikstraat* recording.



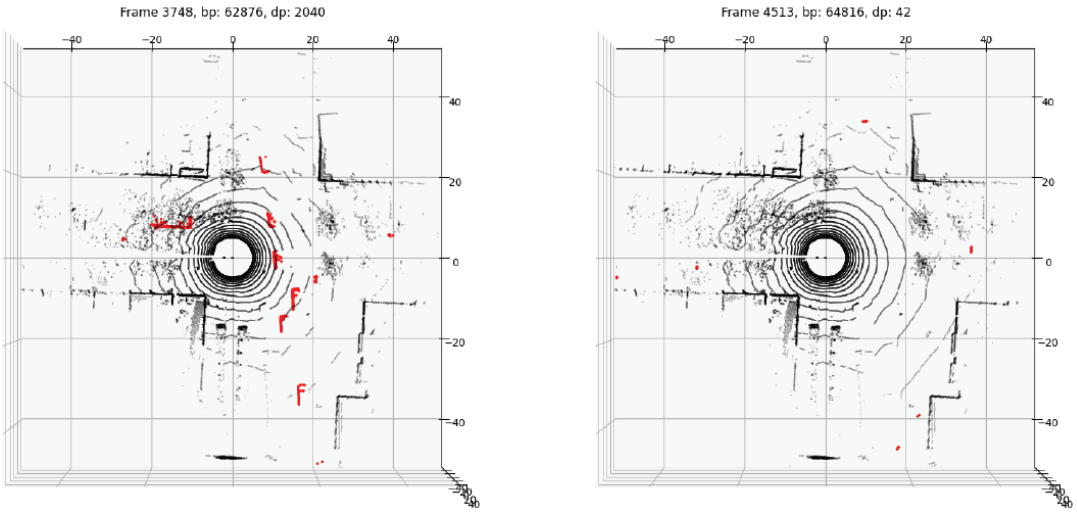
(a) *(b)*

Figure A.2: Evaluation frames for the *Kettingbrugweg* recording.



(a) *(b)*

Figure A.3: Evaluation frames for the *Holterbergweg* recording.



(a) *(b)*

Figure A.4: Evaluation frames for the *Torenstraat* recording.

Appendix B

TSC calibration

	Recording	Eikstraat						Kettingbrugweg					
	Frame	1620			2623			1330			2998		
	Total points	32576			32574			32469			32469		
	Background points	32252			31324			31849			32238		
	Dynamic points	324			1250			620			231		
	Run	1	2	3	1	2	3	1	2	3	1	2	3
	Total points	2037	1982	1995	2564	2530	2555	1391	1399	1353	482	493	481
	Background points	1722	1667	1679	1320	1286	1312	808	813	763	281	290	278
	Dynamic points	315	315	316	1244	1244	1243	583	586	590	201	203	203
std 1	Precision	15.46%	15.89%	15.84%	48.52%	49.17%	48.65%	41.91%	41.89%	43.61%	41.70%	41.18%	42.20%
	Recall	97.22%	97.22%	97.53%	99.52%	99.52%	99.44%	94.03%	94.52%	95.16%	87.01%	87.88%	87.88%
	Type 1 error	5.34%	5.17%	5.21%	4.21%	4.11%	4.19%	2.54%	2.55%	2.40%	0.87%	0.90%	0.86%
	Type 2 error	2.78%	2.78%	2.47%	0.48%	0.48%	0.56%	5.97%	5.48%	4.84%	12.99%	12.12%	12.12%
	Total points	599	590	610	1449	1436	1438	832	843	829	222	223	222
	Background points	307	303	319	211	199	201	284	294	276	54	52	52
	Dynamic points	292	287	291	1238	1237	1237	548	549	553	168	171	170
std 2	Precision	48.75%	48.64%	47.70%	85.44%	86.14%	86.02%	65.87%	65.12%	66.71%	75.68%	76.68%	76.58%
	Recall	90.12%	88.58%	89.81%	99.04%	98.96%	98.96%	88.39%	88.55%	89.19%	72.73%	74.03%	73.59%
	Type 1 error	0.95%	0.94%	0.99%	0.67%	0.64%	0.64%	0.89%	0.92%	0.87%	0.17%	0.16%	0.16%
	Type 2 error	9.88%	11.42%	10.19%	0.96%	1.04%	1.04%	11.61%	11.45%	10.81%	27.27%	25.97%	26.41%
	Total points	324	322	326	1264	1262	1265	624	617	609	161	166	161
	Background points	52	53	53	33	31	34	126	118	121	17	17	16
	Dynamic points	272	269	273	1231	1231	1231	498	499	488	144	149	145
std 3	Precision	83.95%	83.54%	83.74%	97.39%	97.54%	97.31%	79.81%	80.88%	80.13%	89.44%	89.76%	90.06%
	Recall	83.95%	83.02%	84.26%	98.48%	98.48%	98.48%	80.32%	80.48%	78.71%	62.34%	64.50%	62.77%
	Type 1 error	0.16%	0.16%	0.16%	0.11%	0.10%	0.11%	0.40%	0.37%	0.38%	0.05%	0.05%	0.05%
	Type 2 error	16.05%	16.98%	15.74%	1.52%	1.52%	1.52%	19.68%	19.52%	21.29%	37.66%	35.50%	37.23%
	Total points	249	254	249	1243	1243	1244	492	489	494	116	136	114
	Background points	20	20	20	12	12	13	64	64	65	7	8	6
	Dynamic points	229	234	229	1231	1231	1231	428	425	429	109	128	108
std 4	Precision	91.97%	92.13%	91.97%	99.03%	99.03%	98.95%	86.99%	86.91%	86.84%	93.97%	94.12%	94.74%
	Recall	70.68%	72.22%	70.68%	98.48%	98.48%	98.48%	69.03%	68.55%	69.19%	47.19%	55.41%	46.75%
	Type 1 error	0.06%	0.06%	0.06%	0.04%	0.04%	0.04%	0.20%	0.20%	0.20%	0.02%	0.02%	0.02%
	Type 2 error	29.32%	27.78%	29.32%	1.52%	1.52%	1.52%	30.97%	31.45%	30.81%	52.81%	44.59%	53.25%
	Total points	199	216	199	1234	1238	1237	432	419	422	86	107	82
	Background points	9	12	10	9	10	9	40	39	35	4	3	3
	Dynamic points	190	204	189	1225	1228	1228	392	380	387	82	104	79
std 5	Precision	95.48%	94.44%	94.97%	99.27%	99.19%	99.27%	90.74%	90.69%	91.71%	95.35%	97.20%	96.34%
	Recall	58.64%	62.96%	58.33%	98.00%	98.24%	98.24%	63.23%	61.29%	62.42%	35.50%	45.02%	34.20%
	Type 1 error	0.03%	0.04%	0.03%	0.03%	0.03%	0.03%	0.13%	0.12%	0.11%	0.01%	0.01%	0.01%
	Type 2 error	41.36%	37.04%	41.67%	2.00%	1.76%	1.76%	36.77%	38.71%	37.58%	64.50%	54.98%	65.80%

	Recording	Holterbergweg						Torenstraat						
	Frame	384			1398			3748			4513			
	Total points	32569			32501			64916			64858			
	Background points	32031			32159			62876			64816			
	Dynamic points	538			342			2040			42			
	Run	1	2	3	1	2	3	1	2	3	1	2	3	Average
	Total points	5466	5988	5952	3489	3877	3845	2913	2885	2851	1529	1459	1536	
	Background points	4935	5455	5419	3149	3537	3505	1285	1235	1267	1523	1453	1530	
	Dynamic points	531	533	533	340	340	340	1628	1650	1584	6	6	6	
std 1	Precision	9.71%	8.90%	8.95%	9.74%	8.77%	8.84%	55.89%	57.19%	55.56%	0.39%	0.41%	0.39%	30.45%
	Recall	98.70%	99.07%	99.07%	99.42%	99.42%	99.42%	79.80%	80.88%	77.65%	14.29%	14.29%	14.29%	90.21%
	Type 1 error	15.41%	17.03%	16.92%	9.79%	11.00%	10.90%	2.04%	1.96%	2.02%	2.35%	2.24%	2.36%	5.81%
	Type 2 error	1.30%	0.93%	0.93%	0.58%	0.58%	0.58%	20.20%	19.12%	22.35%	85.71%	85.71%	85.71%	9.79%
	Total points	2364	2593	2584	785	851	841	1065	1052	1080	169	164	169	
	Background points	1865	2091	2086	453	519	510	145	144	147	167	162	167	
	Dynamic points	499	502	498	332	332	331	920	908	933	2	2	2	
std 2	Precision	21.11%	19.36%	19.27%	42.29%	39.01%	39.36%	86.38%	86.31%	86.39%	1.18%	1.22%	1.18%	57.73%
	Recall	92.75%	93.31%	92.57%	97.08%	97.08%	96.78%	45.10%	44.51%	45.74%	4.76%	4.76%	4.76%	80.07%
	Type 1 error	5.82%	6.53%	6.51%	1.41%	1.61%	1.59%	0.23%	0.23%	0.23%	0.26%	0.25%	0.26%	1.47%
	Type 2 error	7.25%	6.69%	7.43%	2.92%	2.92%	3.22%	54.90%	55.49%	54.26%	95.24%	95.24%	95.24%	19.93%
	Total points	771	843	840	389	374	380	608	614	559	46	44	45	
	Background points	315	374	381	86	90	89	23	21	20	45	43	44	
	Dynamic points	456	469	459	303	284	291	585	593	539	1	1	1	
std 3	Precision	59.14%	55.63%	54.64%	77.89%	75.94%	76.58%	96.22%	96.58%	96.42%	2.17%	2.27%	2.22%	79.31%
	Recall	84.76%	87.17%	85.32%	88.60%	83.04%	85.09%	28.68%	29.07%	26.42%	2.38%	2.38%	2.38%	71.65%
	Type 1 error	0.98%	1.17%	1.19%	0.27%	0.28%	0.28%	0.04%	0.03%	0.03%	0.07%	0.07%	0.07%	0.29%
	Type 2 error	15.24%	12.83%	14.68%	11.40%	16.96%	14.91%	71.32%	70.93%	73.58%	97.62%	97.62%	97.62%	28.35%
	Total points	502	519	519	275	274	271	183	137	132	23	22	23	
	Background points	85	86	95	31	34	31	10	10	10	23	22	22	
	Dynamic points	417	433	424	244	240	240	173	127	122	0	0	1	
std 4	Precision	83.07%	83.43%	81.70%	88.73%	87.59%	88.56%	94.54%	92.70%	92.42%	0.00%	0.00%	4.35%	86.79%
	Recall	77.51%	80.48%	78.81%	71.35%	70.18%	70.18%	8.48%	6.23%	5.98%	0.00%	0.00%	2.38%	60.65%
	Type 1 error	0.27%	0.27%	0.30%	0.10%	0.11%	0.10%	0.02%	0.02%	0.02%	0.04%	0.03%	0.03%	0.10%
	Type 2 error	22.49%	19.52%	21.19%	28.65%	29.82%	29.82%	91.52%	93.77%	94.02%	100.00%	100.00%	97.62%	39.35%
	Total points	393	402	409	223	209	220	62	59	58	16	17	17	
	Background points	37	36	37	18	18	16	9	8	8	16	17	17	
	Dynamic points	356	366	372	205	191	204	53	51	50	0	0	0	
std 5	Precision	90.59%	91.04%	90.95%	91.93%	91.39%	92.73%	85.48%	86.44%	86.21%	0.00%	0.00%	0.00%	88.70%
	Recall	66.17%	68.03%	69.14%	59.94%	55.85%	59.65%	2.60%	2.50%	2.45%	0.00%	0.00%	0.00%	52.84%
	Type 1 error	0.12%	0.11%	0.12%	0.06%	0.06%	0.05%	0.01%	0.01%	0.01%	0.02%	0.03%	0.03%	0.05%
	Type 2 error	33.83%	31.97%	30.86%	40.06%	44.15%	40.35%	97.40%	97.50%	97.55%	100.00%	100.00%	100.00%	47.16%

Appendix C

PD calibration

	Recording	Eikstraat						Kettingbrugweg					
	Frame	1620	1620	1620	2623	2623	2623	1330	1330	1330	2998	2998	2998
	Total points	32576	32576	32576	32574	32574	32574	32469	32469	32469	32469	32469	32469
	Background points	32252	32252	32252	31324	31324	31324	31849	31849	31849	32238	32238	32238
	Dynamic points	324	324	324	1250	1250	1250	620	620	620	231	231	231
	Run	1	2	3	1	2	3	1	2	3	1	2	3
	Total points	344	345	342	1262	1265	1262	642	650	648	173	178	175
	Background points	23	25	23	15	15	14	53	57	55	7	8	6
	Dynamic points	321	320	319	1247	1250	1248	589	593	593	166	170	169
Peak% .01	Precision	93.31%	92.75%	93.27%	98.81%	98.81%	98.89%	91.74%	91.23%	91.51%	95.95%	95.51%	96.57%
	Recall	99.07%	98.77%	98.46%	99.76%	100.00%	99.84%	95.00%	95.65%	95.65%	71.86%	73.59%	73.16%
	Type 1 error	0.07%	0.08%	0.07%	0.05%	0.05%	0.04%	0.17%	0.18%	0.17%	0.02%	0.02%	0.02%
	Type 2 error	0.93%	1.23%	1.54%	0.24%	0.00%	0.16%	5.00%	4.35%	4.35%	28.14%	26.41%	26.84%
	Total points	354	353	353	1273	1273	1273	691	698	699	209	210	208
	Background points	33	33	34	23	23	23	87	93	94	10	12	9
	Dynamic points	321	320	319	1250	1250	1250	604	605	605	199	198	199
Peak% .02	Precision	90.68%	90.65%	90.37%	98.19%	98.19%	98.19%	87.41%	86.68%	86.55%	95.22%	94.29%	95.67%
	Recall	99.07%	98.77%	98.46%	100.00%	100.00%	100.00%	97.42%	97.58%	97.58%	86.15%	85.71%	86.15%
	Type 1 error	0.10%	0.10%	0.11%	0.07%	0.07%	0.07%	0.27%	0.29%	0.30%	0.03%	0.04%	0.03%
	Type 2 error	0.93%	1.23%	1.54%	0.00%	0.00%	0.00%	2.58%	2.42%	2.42%	13.85%	14.29%	13.85%
	Total points	364	365	362	1279	1279	1278	723	725	727	215	214	213
	Background points	43	45	43	29	29	28	115	116	118	15	15	13
	Dynamic points	321	320	319	1250	1250	1250	608	609	609	200	199	200
Peak% .03	Precision	88.19%	87.67%	88.12%	97.73%	97.73%	97.81%	84.09%	84.00%	83.77%	93.02%	92.99%	93.90%
	Recall	99.07%	98.77%	98.46%	100.00%	100.00%	100.00%	98.06%	98.23%	98.23%	86.58%	86.15%	86.58%
	Type 1 error	0.13%	0.14%	0.13%	0.09%	0.09%	0.09%	0.36%	0.36%	0.37%	0.05%	0.05%	0.04%
	Type 2 error	0.93%	1.23%	1.54%	0.00%	0.00%	0.00%	1.94%	1.77%	1.77%	13.42%	13.85%	13.42%
	Total points	372	372	369	1283	1282	1282	750	755	759	222	224	225
	Background points	51	52	50	33	32	32	140	144	148	20	20	21
	Dynamic points	321	320	319	1250	1250	1250	610	611	611	202	204	204
Peak% .04	Precision	86.29%	86.02%	86.45%	97.43%	97.50%	97.50%	81.33%	80.93%	80.50%	90.99%	91.07%	90.67%
	Recall	99.07%	98.77%	98.46%	100.00%	100.00%	100.00%	98.39%	98.55%	98.55%	87.45%	88.31%	88.31%
	Type 1 error	0.16%	0.16%	0.16%	0.11%	0.10%	0.10%	0.44%	0.45%	0.46%	0.06%	0.06%	0.07%
	Type 2 error	0.93%	1.23%	1.54%	0.00%	0.00%	0.00%	1.61%	1.45%	1.45%	12.55%	11.69%	11.69%
	Total points	377	376	375	1283	1283	1282	767	768	775	238	238	239
	Background points	56	56	56	33	33	32	156	157	164	30	30	30
	Dynamic points	321	320	319	1250	1250	1250	611	611	611	208	208	209
Peak% .05	Precision	85.15%	85.11%	85.07%	97.43%	97.43%	97.50%	79.66%	79.56%	78.84%	87.39%	87.39%	87.45%
	Recall	99.07%	98.77%	98.46%	100.00%	100.00%	100.00%	98.55%	98.55%	98.55%	90.04%	90.04%	90.48%
	Type 1 error	0.17%	0.17%	0.17%	0.11%	0.11%	0.10%	0.49%	0.49%	0.51%	0.09%	0.09%	0.09%
	Type 2 error	0.93%	1.23%	1.54%	0.00%	0.00%	0.00%	1.45%	1.45%	1.45%	9.96%	9.96%	9.52%

	Recording	Holterbergweg						Torenstraat						
	Frame	384	384	384	1398	1398	1398	3748	3748	3748	4513	4513	4513	
	Total points	32569	32569	32569	32501	32501	32501	64916	64916	64916	64858	64858	64858	
	Background points	32031	32031	32031	32159	32159	32159	62876	62876	62876	64816	64816	64816	
	Dynamic points	538	538	538	342	342	342	2040	2040	2040	42	42	42	
	Run	1	2	3	1	2	3	1	2	3	1	2	3	**Average**
	Total points	534	537	539	337	336	336	247	237	243	35	38	34	
	Background points	28	37	33	17	18	16	31	31	28	27	30	26	
	Dynamic points	506	500	506	320	318	320	216	206	215	8	8	8	
Peak% .01	**Precision**	94.76%	93.11%	93.88%	94.96%	94.64%	95.24%	87.45%	86.92%	88.48%	22.86%	21.05%	23.53%	90.48%
	Recall	94.05%	92.94%	94.05%	93.57%	92.98%	93.57%	10.59%	10.10%	10.54%	19.05%	19.05%	19.05%	77.83%
	Type 1 error	0.09%	0.12%	0.10%	0.05%	0.06%	0.05%	0.05%	0.05%	0.04%	0.04%	0.05%	0.04%	0.07%
	Type 2 error	5.95%	7.06%	5.95%	6.43%	7.02%	6.43%	89.41%	89.90%	89.46%	80.95%	80.95%	80.95%	22.17%
	Total points	566	590	589	354	350	351	312	311	317	47	51	46	
	Background points	49	73	74	30	29	27	48	47	44	39	43	38	
	Dynamic points	517	517	515	324	321	324	264	264	273	8	8	8	
Peak% .02	**Precision**	91.34%	87.63%	87.44%	91.53%	91.71%	92.31%	84.62%	84.89%	86.12%	17.02%	15.69%	17.39%	87.58%
	Recall	96.10%	96.10%	95.72%	94.74%	93.86%	94.74%	12.94%	12.94%	13.38%	19.05%	19.05%	19.05%	80.75%
	Type 1 error	0.15%	0.23%	0.23%	0.09%	0.09%	0.08%	0.08%	0.07%	0.07%	0.06%	0.07%	0.06%	0.12%
	Type 2 error	3.90%	3.90%	4.28%	5.26%	6.14%	5.26%	87.06%	87.06%	86.62%	80.95%	80.95%	80.95%	19.25%
	Total points	639	662	661	361	362	362	467	476	484	56	59	59	
	Background points	120	141	143	35	36	34	55	56	51	46	48	49	
	Dynamic points	519	521	518	326	326	328	412	420	433	10	11	10	
Peak% .03	**Precision**	81.22%	78.70%	78.37%	90.30%	90.06%	90.61%	88.22%	88.24%	89.46%	17.86%	18.64%	16.95%	85.55%
	Recall	96.47%	96.84%	96.28%	95.32%	95.32%	95.91%	20.20%	20.59%	21.23%	23.81%	26.19%	23.81%	82.37%
	Type 1 error	0.37%	0.44%	0.45%	0.11%	0.11%	0.11%	0.09%	0.09%	0.08%	0.07%	0.07%	0.08%	0.17%
	Type 2 error	3.53%	3.16%	3.72%	4.68%	4.68%	4.09%	79.80%	79.41%	78.77%	76.19%	73.81%	76.19%	17.63%
	Total points	679	688	683	371	372	370	709	736	734	72	71	71	
	Background points	153	160	158	43	44	41	64	63	61	58	57	58	
	Dynamic points	526	528	525	328	328	329	645	673	673	14	14	13	
Peak% .04	**Precision**	77.47%	76.74%	76.87%	88.41%	88.17%	88.92%	90.97%	91.44%	91.69%	19.44%	19.72%	18.31%	84.40%
	Recall	97.77%	98.14%	97.58%	95.91%	95.91%	96.20%	31.62%	32.99%	32.99%	33.33%	33.33%	30.95%	84.92%
	Type 1 error	0.48%	0.50%	0.49%	0.13%	0.14%	0.13%	0.10%	0.10%	0.10%	0.09%	0.09%	0.09%	0.21%
	Type 2 error	2.23%	1.86%	2.42%	4.09%	4.09%	3.80%	68.38%	67.01%	67.01%	66.67%	66.67%	69.05%	15.08%
	Total points	706	708	704	379	383	382	847	865	846	84	84	81	
	Background points	172	176	174	51	55	53	79	76	70	70	70	67	
	Dynamic points	534	532	530	328	328	329	768	789	776	14	14	14	
Peak% .05	**Precision**	75.64%	75.14%	75.28%	86.54%	85.64%	86.13%	90.67%	91.21%	91.73%	16.67%	16.67%	17.28%	82.85%
	Recall	99.26%	98.88%	98.51%	95.91%	95.91%	96.20%	37.65%	38.68%	38.04%	33.33%	33.33%	33.33%	86.13%
	Type 1 error	0.54%	0.55%	0.54%	0.16%	0.17%	0.16%	0.13%	0.12%	0.11%	0.11%	0.11%	0.10%	0.24%
	Type 2 error	0.74%	1.12%	1.49%	4.09%	4.09%	3.80%	62.35%	61.32%	61.96%	66.67%	66.67%	66.67%	13.87%

	Recording	Eikstraat						Kettingbrugweg					
	Frame	1620	1620	1620	2623	2623	2623	1330	1330	1330	2998	2998	2998
	Total points	408	406	404	1309	1309	1309	856	850	856	285	283	284
	Background points	87	86	85	59	59	59	244	237	243	66	64	65
	Dynamic points	321	320	319	1250	1250	1250	612	613	613	219	219	219
Peak% .1	Precision	78.68%	78.82%	78.96%	95.49%	95.49%	95.49%	71.50%	72.12%	71.61%	76.84%	77.39%	77.11%
	Recall	99.07%	98.77%	98.46%	100.00%	100.00%	100.00%	98.71%	98.87%	98.87%	94.81%	94.81%	94.81%
	Type 1 error	0.27%	0.27%	0.26%	0.19%	0.19%	0.19%	0.77%	0.74%	0.76%	0.20%	0.20%	0.20%
	Type 2 error	0.93%	1.23%	1.54%	0.00%	0.00%	0.00%	1.29%	1.13%	1.13%	5.19%	5.19%	5.19%
	Total points	435	432	434	1324	1325	1326	926	924	920	316	313	312
	Background points	114	112	115	74	75	76	315	312	308	97	94	93
	Dynamic points	321	320	319	1250	1250	1250	611	612	612	219	219	219
Peak% .15	Precision	73.79%	74.07%	73.50%	94.41%	94.34%	94.27%	65.98%	66.23%	66.52%	69.30%	69.97%	70.19%
	Recall	99.07%	98.77%	98.46%	100.00%	100.00%	100.00%	98.55%	98.71%	98.71%	94.81%	94.81%	94.81%
	Type 1 error	0.35%	0.35%	0.36%	0.24%	0.24%	0.24%	0.99%	0.98%	0.97%	0.30%	0.29%	0.29%
	Type 2 error	0.93%	1.23%	1.54%	0.00%	0.00%	0.00%	1.45%	1.29%	1.29%	5.19%	5.19%	5.19%
	Total points	445	442	446	1330	1333	1335	963	963	957	328	324	324
	Background points	124	122	127	80	83	85	353	352	346	109	105	105
	Dynamic points	321	320	319	1250	1250	1250	610	611	611	219	219	219
Peak% .2	Precision	72.13%	72.40%	71.52%	93.98%	93.77%	93.63%	63.34%	63.45%	63.85%	66.77%	67.59%	67.59%
	Recall	99.07%	98.77%	98.46%	100.00%	100.00%	100.00%	98.39%	98.55%	98.55%	94.81%	94.81%	94.81%
	Type 1 error	0.38%	0.38%	0.39%	0.26%	0.26%	0.27%	1.11%	1.11%	1.09%	0.34%	0.33%	0.33%
	Type 2 error	0.93%	1.23%	1.54%	0.00%	0.00%	0.00%	1.61%	1.45%	1.45%	5.19%	5.19%	5.19%
	Total points	452	449	450	1338	1337	1341	983	984	980	341	339	337
	Background points	131	129	131	88	88	92	375	375	371	122	120	118
	Dynamic points	321	320	319	1250	1249	1249	608	609	609	219	219	219
Peak% .25	Precision	71.02%	71.27%	70.89%	93.42%	93.42%	93.14%	61.85%	61.89%	62.14%	64.22%	64.60%	64.99%
	Recall	99.07%	98.77%	98.46%	100.00%	99.92%	99.92%	98.06%	98.23%	98.23%	94.81%	94.81%	94.81%
	Type 1 error	0.41%	0.40%	0.41%	0.28%	0.28%	0.29%	1.18%	1.18%	1.16%	0.38%	0.37%	0.37%
	Type 2 error	0.93%	1.23%	1.54%	0.00%	0.08%	0.08%	1.94%	1.77%	1.77%	5.19%	5.19%	5.19%
	Total points	460	457	460	1340	1341	1343	1006	1006	1005	356	352	351
	Background points	139	137	141	91	92	94	398	397	396	137	133	132
	Dynamic points	321	320	319	1249	1249	1249	608	609	609	219	219	219
Peak% .3	Precision	69.78%	70.02%	69.35%	93.21%	93.14%	93.00%	60.44%	60.54%	60.60%	61.52%	62.22%	62.39%
	Recall	99.07%	98.77%	98.46%	99.92%	99.92%	99.92%	98.06%	98.23%	98.23%	94.81%	94.81%	94.81%
	Type 1 error	0.43%	0.42%	0.44%	0.29%	0.29%	0.30%	1.25%	1.25%	1.24%	0.42%	0.41%	0.41%
	Type 2 error	0.93%	1.23%	1.54%	0.08%	0.08%	0.08%	1.94%	1.77%	1.77%	5.19%	5.19%	5.19%

	Recording	Holterbergweg						Torenstraat						
	Frame	384	384	384	1398	1398	1398	3748	3748	3748	4513	4513	4513	
	Total points	778	783	783	439	444	444	1269	1287	1272	148	146	151	
	Background points	241	247	248	111	116	115	127	128	125	128	128	129	
	Dynamic points	537	536	535	328	328	329	1142	1159	1147	20	18	22	
Peak% .1	Precision	69.02%	68.45%	68.33%	74.72%	73.87%	74.10%	89.99%	90.05%	90.17%	13.51%	12.33%	14.57%	76.44%
	Recall	99.81%	99.63%	99.44%	95.91%	95.91%	96.20%	55.98%	56.81%	56.23%	47.62%	42.86%	52.38%	90.03%
	Type 1 error	0.75%	0.77%	0.77%	0.35%	0.36%	0.36%	0.20%	0.20%	0.20%	0.20%	0.20%	0.20%	0.38%
	Type 2 error	0.19%	0.37%	0.56%	4.09%	4.09%	3.80%	44.02%	43.19%	43.77%	52.38%	57.14%	47.62%	9.97%
	Total points	849	854	851	504	502	506	1379	1368	1372	219	218	225	
	Background points	312	318	316	176	174	177	163	165	164	178	177	184	
	Dynamic points	537	536	535	328	328	329	1216	1203	1208	41	41	41	
Peak% .15	Precision	63.25%	62.76%	62.87%	65.08%	65.34%	65.02%	88.18%	87.94%	88.05%	18.72%	18.81%	18.22%	71.81%
	Recall	99.81%	99.63%	99.44%	95.91%	95.91%	96.20%	59.61%	58.97%	59.22%	97.62%	97.62%	97.62%	92.68%
	Type 1 error	0.97%	0.99%	0.99%	0.55%	0.54%	0.55%	0.26%	0.26%	0.26%	0.27%	0.27%	0.28%	0.51%
	Type 2 error	0.19%	0.37%	0.56%	4.09%	4.09%	3.80%	40.39%	41.03%	40.78%	2.38%	2.38%	2.38%	7.32%
	Total points	900	900	901	551	548	548	1402	1410	1394	237	237	244	
	Background points	364	365	367	223	220	219	190	209	190	196	196	203	
	Dynamic points	536	535	534	328	328	329	1212	1201	1204	41	41	41	
Peak% .2	Precision	59.56%	59.44%	59.27%	59.53%	59.85%	60.04%	86.45%	85.18%	86.37%	17.30%	17.30%	16.80%	69.23%
	Recall	99.63%	99.44%	99.26%	95.91%	95.91%	96.20%	59.41%	58.87%	59.02%	97.62%	97.62%	97.62%	92.61%
	Type 1 error	1.14%	1.14%	1.15%	0.69%	0.68%	0.68%	0.30%	0.33%	0.30%	0.30%	0.30%	0.31%	0.59%
	Type 2 error	0.37%	0.56%	0.74%	4.09%	4.09%	3.80%	40.59%	41.13%	40.98%	2.38%	2.38%	2.38%	7.39%
	Total points	943	945	941	594	596	594	1467	1457	1456	249	251	257	
	Background points	408	411	408	266	268	266	252	254	253	209	211	217	
	Dynamic points	535	534	533	328	328	328	1215	1203	1203	40	40	40	
Peak% .25	Precision	56.73%	56.51%	56.64%	55.22%	55.03%	55.22%	82.82%	82.57%	82.62%	16.06%	15.94%	15.56%	66.92%
	Recall	99.44%	99.26%	99.07%	95.91%	95.91%	95.91%	59.56%	58.97%	58.97%	95.24%	95.24%	95.24%	92.42%
	Type 1 error	1.27%	1.28%	1.27%	0.83%	0.83%	0.83%	0.40%	0.40%	0.40%	0.32%	0.33%	0.33%	0.66%
	Type 2 error	0.56%	0.74%	0.93%	4.09%	4.09%	4.09%	40.44%	41.03%	41.03%	4.76%	4.76%	4.76%	7.58%
	Total points	978	980	977	627	626	622	1485	1477	1475	271	269	278	
	Background points	443	446	444	298	297	293	268	267	268	231	229	238	
	Dynamic points	535	534	533	329	329	329	1217	1210	1207	40	40	40	
Peak% .3	Precision	54.70%	54.49%	54.55%	52.47%	52.56%	52.89%	81.95%	81.92%	81.83%	14.76%	14.87%	14.39%	65.38%
	Recall	99.44%	99.26%	99.07%	96.20%	96.20%	96.20%	59.66%	59.31%	59.17%	95.24%	95.24%	95.24%	92.49%
	Type 1 error	1.38%	1.39%	1.39%	0.93%	0.92%	0.91%	0.43%	0.42%	0.43%	0.36%	0.35%	0.37%	0.71%
	Type 2 error	0.56%	0.74%	0.93%	3.80%	3.80%	3.80%	40.34%	40.69%	40.83%	4.76%	4.76%	4.76%	7.51%

	Recording	Eikstraat						Kettingbrugweg					
	Frame	1620	1620	1620	2623	2623	2623	1330	1330	1330	2998	2998	2998
	Total points	470	466	469	1347	1346	1348	1028	1029	1029	367	366	363
	Background points	149	146	150	98	97	99	422	422	422	149	147	145
	Dynamic points	321	320	319	1249	1249	1249	606	607	607	218	219	218
Peak% .35	Precision	68.30%	68.67%	68.02%	92.72%	92.79%	92.66%	58.95%	58.99%	58.99%	59.40%	59.84%	60.06%
	Recall	99.07%	98.77%	98.46%	99.92%	99.92%	99.92%	97.74%	97.90%	97.90%	94.37%	94.81%	94.37%
	Type 1 error	0.46%	0.45%	0.47%	0.31%	0.31%	0.32%	1.33%	1.33%	1.33%	0.46%	0.46%	0.45%
	Type 2 error	0.93%	1.23%	1.54%	0.08%	0.08%	0.08%	2.26%	2.10%	2.10%	5.63%	5.19%	5.63%
	Total points	478	474	477	1354	1352	1355	1041	1042	1041	379	381	380
	Background points	157	154	158	105	103	106	435	435	434	161	163	162
	Dynamic points	321	320	319	1249	1249	1249	606	607	607	218	218	218
Peak% .4	Precision	67.15%	67.51%	66.88%	92.25%	92.38%	92.18%	58.21%	58.25%	58.31%	57.52%	57.22%	57.37%
	Recall	99.07%	98.77%	98.46%	99.92%	99.92%	99.92%	97.74%	97.90%	97.90%	94.37%	94.37%	94.37%
	Type 1 error	0.49%	0.48%	0.49%	0.34%	0.33%	0.34%	1.37%	1.37%	1.36%	0.50%	0.51%	0.50%
	Type 2 error	0.93%	1.23%	1.54%	0.08%	0.08%	0.08%	2.26%	2.10%	2.10%	5.63%	5.63%	5.63%
	Total points	488	485	483	1364	1362	1362	1050	1052	1050	393	395	396
	Background points	167	165	164	115	113	113	445	446	444	175	177	178
	Dynamic points	321	320	319	1249	1249	1249	605	606	606	218	218	218
Peak% .45	Precision	65.78%	65.98%	66.05%	91.57%	91.70%	91.70%	57.62%	57.60%	57.71%	55.47%	55.19%	55.05%
	Recall	99.07%	98.77%	98.46%	99.92%	99.92%	99.92%	97.58%	97.74%	97.74%	94.37%	94.37%	94.37%
	Type 1 error	0.52%	0.51%	0.51%	0.37%	0.36%	0.36%	1.40%	1.40%	1.39%	0.54%	0.55%	0.55%
	Type 2 error	0.93%	1.23%	1.54%	0.08%	0.08%	0.08%	2.42%	2.26%	2.26%	5.63%	5.63%	5.63%
	Total points	494	489	490	1368	1366	1369	1048	1054	1051	399	400	402
	Background points	173	169	171	119	117	120	444	449	446	181	182	184
	Dynamic points	321	320	319	1249	1249	1249	604	605	605	218	218	218
Peak% .5	Precision	64.98%	65.44%	65.10%	91.30%	91.43%	91.23%	57.63%	57.40%	57.56%	54.64%	54.50%	54.23%
	Recall	99.07%	98.77%	98.46%	99.92%	99.92%	99.92%	97.42%	97.58%	97.58%	94.37%	94.37%	94.37%
	Type 1 error	0.54%	0.52%	0.53%	0.38%	0.37%	0.38%	1.39%	1.41%	1.40%	0.56%	0.56%	0.57%
	Type 2 error	0.93%	1.23%	1.54%	0.08%	0.08%	0.08%	2.58%	2.42%	2.42%	5.63%	5.63%	5.63%

	Recording	Holterbergweg						Torenstraat						
	Frame	384	384	384	1398	1398	1398	3748	3748	3748	4513	4513	4513	
	Total points	998	1001	998	654	656	658	1507	1495	1494	280	282	286	
	Background points	466	470	467	325	327	328	286	283	281	242	244	248	
	Dynamic points	532	531	531	329	329	330	1221	1212	1213	38	38	38	
Peak% .35	**Precision**	53.31%	53.05%	53.21%	50.31%	50.15%	50.15%	81.02%	81.07%	81.19%	13.57%	13.48%	13.29%	63.93%
	Recall	98.88%	98.70%	98.70%	96.20%	96.20%	96.49%	59.85%	59.41%	59.46%	90.48%	90.48%	90.48%	92.16%
	Type 1 error	1.45%	1.47%	1.46%	1.01%	1.02%	1.02%	0.45%	0.45%	0.45%	0.37%	0.38%	0.38%	0.76%
	Type 2 error	1.12%	1.30%	1.30%	3.80%	3.80%	3.51%	40.15%	40.59%	40.54%	9.52%	9.52%	9.52%	7.84%
	Total points	1019	1023	1020	679	680	679	1507	1497	1501	285	284	295	
	Background points	491	495	491	349	350	349	295	289	293	248	247	259	
	Dynamic points	528	528	529	330	330	330	1212	1208	1208	37	37	36	
Peak% .4	**Precision**	51.82%	51.61%	51.86%	48.60%	48.53%	48.60%	80.42%	80.69%	80.48%	12.98%	13.03%	12.20%	62.77%
	Recall	98.14%	98.14%	98.33%	96.49%	96.49%	96.49%	59.41%	59.22%	59.22%	88.10%	88.10%	85.71%	91.94%
	Type 1 error	1.53%	1.55%	1.53%	1.09%	1.09%	1.09%	0.47%	0.46%	0.47%	0.38%	0.38%	0.40%	0.80%
	Type 2 error	1.86%	1.86%	1.67%	3.51%	3.51%	3.51%	40.59%	40.78%	40.78%	11.90%	11.90%	14.29%	8.06%
	Total points	1041	1046	1042	695	702	698	1649	1821	1516	295	291	299	
	Background points	514	519	515	365	372	367	299	298	299	259	255	263	
	Dynamic points	527	527	527	330	330	331	1350	1523	1217	36	36	36	
Peak% .45	**Precision**	50.62%	50.38%	50.58%	47.48%	47.01%	47.42%	81.87%	83.64%	80.28%	12.20%	12.37%	12.04%	61.95%
	Recall	97.96%	97.96%	97.96%	96.49%	96.49%	96.78%	66.18%	74.66%	59.66%	85.71%	85.71%	85.71%	92.82%
	Type 1 error	1.60%	1.62%	1.61%	1.13%	1.16%	1.14%	0.48%	0.47%	0.48%	0.40%	0.39%	0.41%	0.84%
	Type 2 error	2.04%	2.04%	2.04%	3.51%	3.51%	3.22%	33.82%	25.34%	40.34%	14.29%	14.29%	14.29%	7.18%
	Total points	1035	1041	1041	703	706	706	1769	1777	1743	288	284	296	
	Background points	512	520	519	374	377	378	296	299	296	252	248	260	
	Dynamic points	523	521	522	329	329	328	1473	1478	1447	36	36	36	
Peak% .5	**Precision**	50.53%	50.05%	50.14%	46.80%	46.60%	46.46%	83.27%	83.17%	83.02%	12.50%	12.68%	12.16%	61.73%
	Recall	97.21%	96.84%	97.03%	96.20%	96.20%	95.91%	72.21%	72.45%	70.93%	85.71%	85.71%	85.71%	93.29%
	Type 1 error	1.60%	1.62%	1.62%	1.16%	1.17%	1.18%	0.47%	0.48%	0.47%	0.39%	0.38%	0.40%	0.85%
	Type 2 error	2.79%	3.16%	2.97%	3.80%	3.80%	4.09%	27.79%	27.55%	29.07%	14.29%	14.29%	14.29%	6.71%

Appendix D

3D-DSF calibration

	Recording	Eikstraat						Kettingbrugweg					
	Frame	1620	1620	1620	2623	2623	2623	1330	1330	1330	2998	2998	2998
	Total points	32576	32576	32576	32574	32574	32574	32469	32469	32469	32469	32469	32469
	Background points	32252	32252	32252	31324	31324	31324	31849	31849	31849	32238	32238	32238
	Dynamic points	324	324	324	1250	1250	1250	620	620	620	231	231	231
	Run	1	2	3	1	2	3	1	2	3	1	2	3
	Total points	377	379	376	437	981	738	788	859	888	272	271	282
	Background points	55	57	53	43	48	46	236	240	269	52	46	55
	Dynamic points	322	322	323	394	933	692	552	619	619	220	225	227
No filter	Precision	85.41%	84.96%	85.90%	90.16%	95.11%	93.77%	70.05%	72.06%	69.71%	80.88%	83.03%	80.50%
	Recall	99.38%	99.38%	99.69%	31.52%	74.64%	55.36%	89.03%	99.84%	99.84%	95.24%	97.40%	98.27%
	Type 1 error	0.17%	0.18%	0.16%	0.14%	0.15%	0.15%	0.74%	0.75%	0.84%	0.16%	0.14%	0.17%
	Type 2 error	0.62%	0.62%	0.31%	68.48%	25.36%	44.64%	10.97%	0.16%	0.16%	4.76%	2.60%	1.73%
	Total points	398	403	395	1153	1284	1212	985	971	1005	305	310	308
	Background points	74	79	71	56	59	53	369	352	386	78	83	81
	Dynamic points	324	324	324	1097	1225	1159	616	619	619	227	227	227
MA 1	Precision	81.41%	80.40%	82.03%	95.14%	95.40%	95.63%	62.54%	63.75%	61.59%	74.43%	73.23%	73.70%
	Recall	100.00%	100.00%	100.00%	87.76%	98.00%	92.72%	99.35%	99.84%	99.84%	98.27%	98.27%	98.27%
	Type 1 error	0.23%	0.24%	0.22%	0.18%	0.19%	0.17%	1.16%	1.11%	1.21%	0.24%	0.26%	0.25%
	Type 2 error	0.00%	0.00%	0.00%	12.24%	2.00%	7.28%	0.65%	0.16%	0.16%	1.73%	1.73%	1.73%
	Total points	408	421	409	1253	1315	1245	1068	1057	1016	328	334	325
	Background points	84	97	85	64	71	66	449	438	397	100	106	98
	Dynamic points	324	324	324	1189	1244	1179	619	619	619	228	228	227
MA 2	Precision	79.41%	76.96%	79.22%	94.89%	94.60%	94.70%	57.96%	58.56%	60.93%	69.51%	68.26%	69.85%
	Recall	100.00%	100.00%	100.00%	95.12%	99.52%	94.32%	99.84%	99.84%	99.84%	98.70%	98.70%	98.27%
	Type 1 error	0.26%	0.30%	0.26%	0.20%	0.23%	0.21%	1.41%	1.38%	1.25%	0.31%	0.33%	0.30%
	Type 2 error	0.00%	0.00%	0.00%	4.88%	0.48%	5.68%	0.16%	0.16%	0.16%	1.30%	1.30%	1.73%
	Total points	419	425	418	1254	1319	1304	1099	1079	1105	356	355	346
	Background points	95	101	94	68	69	71	480	460	486	128	127	119
	Dynamic points	324	324	324	1186	1250	1233	619	619	619	228	228	227
MA 3	Precision	77.33%	76.24%	77.51%	94.58%	94.77%	94.56%	56.32%	57.37%	56.02%	64.04%	64.23%	65.61%
	Recall	100.00%	100.00%	100.00%	94.88%	100.00%	98.84%	99.84%	99.84%	99.84%	98.70%	98.70%	98.27%
	Type 1 error	0.29%	0.31%	0.29%	0.22%	0.22%	0.23%	1.51%	1.44%	1.53%	0.40%	0.39%	0.37%
	Type 2 error	0.00%	0.00%	0.00%	5.12%	0.00%	1.36%	0.16%	0.16%	0.16%	1.30%	1.30%	1.73%
	Total points	421	431	429	1304	1328	1309	1113	1076	1108	369	357	352
	Background points	97	107	105	73	80	76	494	457	489	141	129	124
	Dynamic points	324	324	324	1231	1248	1233	619	619	619	228	228	228
	Precision	76.96%	75.17%	75.52%	94.40%	93.98%	94.19%	55.62%	57.53%	55.87%	61.79%	63.87%	64.77%
	Recall	100.00%	100.00%	100.00%	98.48%	99.84%	98.64%	99.84%	99.84%	99.84%	98.70%	98.70%	98.70%
	Type 1 error	0.30%	0.33%	0.33%	0.23%	0.26%	0.24%	1.55%	1.43%	1.54%	0.44%	0.40%	0.38%
MA 4	Type 2 error	0.00%	0.00%	0.00%	1.52%	0.16%	1.36%	0.16%	0.16%	0.16%	1.30%	1.30%	1.30%

	Recording	Holterbergweg						Torenstraat						
	Frame	384	384	384	1398	1398	1398	3748	3748	3748	4513	4513	4513	
	Total points	32569	32569	32569	32501	32501	32501	64916	64916	64916	64858	64858	64858	
	Background points	32031	32031	32031	32159	32159	32159	62876	62876	62876	64816	64816	64816	
	Dynamic points	538	538	538	342	342	342	2040	2040	2040	42	42	42	
	Run	1	2	3	1	2	3	1	2	3	1	2	3	**Average**
	Total points	695	725	711	465	446	445	926	867	876	136	143	136	
	Background points	177	187	179	123	106	106	122	109	116	94	101	94	
	Dynamic points	518	538	532	342	340	339	804	758	760	42	42	42	
No filter	Precision	74.53%	74.21%	74.82%	73.55%	76.23%	76.18%	86.83%	87.43%	86.76%	30.88%	29.37%	30.88%	78.77%
	Recall	96.28%	100.00%	98.88%	100.00%	99.42%	99.12%	39.41%	37.16%	37.25%	100.00%	100.00%	100.00%	83.96%
	Type 1 error	0.55%	0.58%	0.56%	0.38%	0.33%	0.33%	0.19%	0.17%	0.18%	0.15%	0.16%	0.15%	0.33%
	Type 2 error	3.72%	0.00%	1.12%	0.00%	0.58%	0.88%	60.59%	62.84%	62.75%	0.00%	0.00%	0.00%	16.04%
	Total points	786	786	824	486	486	526	1299	1232	1060	178	168	166	
	Background points	248	248	286	144	144	184	152	144	148	136	126	124	
	Dynamic points	538	538	538	342	342	342	1147	1088	912	42	42	42	
MA 1	Precision	68.45%	68.45%	65.29%	70.37%	70.37%	65.02%	88.30%	88.31%	86.04%	23.60%	25.00%	25.30%	74.25%
	Recall	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	56.23%	53.33%	44.71%	100.00%	100.00%	100.00%	92.12%
	Type 1 error	0.77%	0.77%	0.89%	0.45%	0.45%	0.57%	0.24%	0.23%	0.24%	0.21%	0.19%	0.19%	0.47%
	Type 2 error	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	43.77%	46.67%	55.29%	0.00%	0.00%	0.00%	7.88%
	Total points	845	893	912	536	558	572	1319	1306	1366	188	206	184	
	Background points	307	355	374	194	216	230	164	173	163	146	164	142	
	Dynamic points	538	538	538	342	342	342	1155	1133	1203	42	42	42	
MA 2	Precision	63.67%	60.25%	58.99%	63.81%	61.29%	59.79%	87.57%	86.75%	88.07%	22.34%	20.39%	22.83%	70.79%
	Recall	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	56.62%	55.54%	58.97%	100.00%	100.00%	100.00%	93.42%
	Type 1 error	0.96%	1.11%	1.17%	0.60%	0.67%	0.72%	0.26%	0.28%	0.26%	0.23%	0.25%	0.22%	0.58%
	Type 2 error	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	43.38%	44.46%	41.03%	0.00%	0.00%	0.00%	6.58%
	Total points	903	903	936	568	568	590	1426	1319	1389	216	206	220	
	Background points	365	365	398	226	226	248	192	171	192	174	164	178	
	Dynamic points	538	538	538	342	342	342	1234	1148	1197	42	42	42	
MA 3	Precision	59.58%	59.58%	57.48%	60.21%	60.21%	57.97%	86.54%	87.04%	86.18%	19.44%	20.39%	19.09%	68.76%
	Recall	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	60.49%	56.27%	58.68%	100.00%	100.00%	100.00%	93.83%
	Type 1 error	1.14%	1.14%	1.24%	0.70%	0.70%	0.77%	0.31%	0.27%	0.31%	0.27%	0.25%	0.27%	0.64%
	Type 2 error	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	39.51%	43.73%	41.32%	0.00%	0.00%	0.00%	6.18%
	Total points	970	951	962	619	590	601	1435	1397	1424	214	245	212	
	Background points	432	413	424	277	248	259	191	198	190	172	203	170	
	Dynamic points	538	538	538	342	342	342	1244	1199	1234	42	42	42	
	Precision	55.46%	56.57%	55.93%	55.25%	57.97%	56.91%	86.69%	85.83%	86.66%	19.63%	17.14%	19.81%	67.57%
	Recall	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	60.98%	58.77%	60.49%	100.00%	100.00%	100.00%	94.22%
	Type 1 error	1.35%	1.29%	1.32%	0.86%	0.77%	0.81%	0.30%	0.31%	0.30%	0.27%	0.31%	0.26%	0.68%
MA 4	Type 2 error	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	39.02%	41.23%	39.51%	0.00%	0.00%	0.00%	5.78%