



Master's Thesis in Robotics, Cognition, Intelligence

# Unsupervised LiDAR-based 3D Object Detection Using Infrastructure Sensors

Unüberwachte LiDAR-basierte 3D-Objekterkennung mit  
Infrastruktursensoren

<b>Supervisor</b>	Prof. Dr.-Ing. habil. Alois C. Knoll
<b>Advisor</b>	Walter Zimmer, M.Sc.
<b>Author</b>	Marcel Brucker
<b>Date</b>	September 15, 2022 in Garching



# Disclaimer

I confirm that this Master's Thesis is my own work and I have documented all sources and material used.

Garching, September 15, 2022

---

(Marcel Brucker)

## **Abstract**

As humans cause most severe car crashes [71], lawmakers have been enforcing mandatory driver assistance systems for vehicles such as electronic stability control. Advanced driver assistance requires reliable environmental perception through cameras, radar, and LiDAR sensors. When installed on road infrastructure, in addition to vehicles, these sensors facilitate autonomous transport because they allow foresight of traffic events from a more convenient perspective. Provided that vehicles receive complete environment information externally from road infrastructure via wireless communication, car manufacturers could save on some costly on-board sensors.

To obtain complete environment information, the sensor data must be automatically analyzed by intelligent object detection software. This research work produces a modular unsupervised approach that does not require training on sample data and processes LiDAR data at near 50 Hz while extracting objects and their 3D information. The presented solution is a suitable code base, which can be further improved by related research works, examined in a comprehensive literature review.

## **Zusammenfassung**

Da der Mensch die meisten schweren Verkehrsunfälle verursacht [71], hat der Gesetzgeber Fahrerassistenzsysteme wie die elektronische Stabilitätskontrolle für Autos erzwungen. Fortgeschrittene Fahrerassistenz erfordert eine zuverlässige Umgebungswahrnehmung durch Kameras, Radar- und LiDAR-Sensoren. Wenn diese Sensoren nicht nur an Fahrzeugen, sondern auch an der Verkehrsinfrastruktur angebracht sind, erleichtern sie die Realisierung des autonomen Verkehrs, da sie aus einer geeigneteren Perspektive eine Vorausschau auf das Verkehrsgeschehen ermöglichen. Wenn Fahrzeuge vollständige Umgebungsinformationen extern von der Verkehrsinfrastruktur über drahtlose Kommunikation erhalten, können Automobilhersteller einzelne teure Bordsensoren einsparen.

Um vollständige Umgebungsinformationen zu erlangen, müssen Sensordaten automatisch von einer intelligenten Objekterkennungssoftware ausgewertet werden. In dieser Forschungsarbeit wird ein modularer Ansatz entwickelt, der kein Training auf Beispieldaten erfordert und LiDAR-Daten mit nahezu 50 Hz verarbeitet, wobei er Objekte und deren 3D-Informationen extrahiert. Die vorgestellte Lösung ist eine geeignete Codebasis, die durch verwandte Forschungsarbeiten, welche in einer umfassenden Literaturrecherche beleuchtet werden, weiter verbessert werden kann.

# Acknowledgement

I would like to thank my advisor Walter Zimmer for entrusting me with this exciting research topic and for his constant interest in assisting me over the past six months. Furthermore, I show my gratitude that he willingly mentored me during my research stay abroad.

Abroad, I would like to express my special thanks to Prof. Branch Bedoya as well as Prof. Sánchez Torres, who warmly welcomed me to their research group for artificial intelligence GIDIA and allowed me to spend an eventful research stay at the Medellín campus of the National University of Colombia. The insights into academia provided me with valuable experiences. I greatly appreciated the trust placed in me and the scientific freedom provided.

Finally, I would like to thank the German Academic Exchange Service DAAD for choosing to make me a scholarship holder, making my stay in Colombia and this research work financially possible.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Problem Statement . . . . .	2
1.3	Contribution . . . . .	3
<b>2</b>	<b>Background</b>	<b>5</b>
2.1	Sensor Hardware . . . . .	5
2.2	LiDAR Output . . . . .	8
2.3	Computer Vision Tasks . . . . .	10
2.4	Paradigms of Machine Learning . . . . .	11
2.5	Intelligent Infrastructure . . . . .	12
2.6	Intelligent Infrastructure Projects . . . . .	14
2.7	Datasets . . . . .	15
2.8	Point Cloud Libraries . . . . .	16
2.9	Evaluation Metrics . . . . .	16
<b>3</b>	<b>Related Work</b>	<b>21</b>
3.1	3D Object Detection . . . . .	21
3.1.1	Supervised 3D Object Detection . . . . .	21
3.1.2	Unsupervised 3D Object Detection . . . . .	24
3.2	Ground Segmentation . . . . .	25
3.3	Background Filtering . . . . .	30
3.4	Outlier Removal . . . . .	36
3.5	Point Generation . . . . .	37
3.6	Clustering . . . . .	38
3.7	Bounding Box Fitting . . . . .	40
3.8	Classification . . . . .	42
<b>4</b>	<b>Methodology</b>	<b>47</b>
4.1	Region of Interest Selection . . . . .	48
4.2	Ground Segmentation . . . . .	50
4.3	Background Filtering . . . . .	52
4.4	Outlier Removal . . . . .	54
4.5	Clustering . . . . .	55
4.6	Bounding Box Fitting . . . . .	56
4.7	Classification . . . . .	57
<b>5</b>	<b>Evaluation</b>	<b>61</b>
5.1	Analysis of Missed LiDAR Reflections . . . . .	62
5.2	Point Reduction . . . . .	63
5.3	Average Precision and Average Recall . . . . .	63

5.4	Runtime Measurements . . . . .	65
5.5	Mean Average Precision through Intersection over Union in 3D . . . . .	66
5.6	Visual Results . . . . .	68
<b>6</b>	<b>Future Work</b>	<b>73</b>
<b>7</b>	<b>Conclusion</b>	<b>77</b>
<b>A</b>	<b>Appendix 1</b>	<b>79</b>
	<b>Bibliography</b>	<b>85</b>



# Chapter 1

## Introduction

### 1.1 Motivation

Autonomous vehicles are beneficial for passengers and society as a whole [103] because they will significantly reduce the number of traffic accidents as a recent survey reasons after noting 94% of severe car crashes are due to human failure [71]. Drunk driving, speeding, reacting emotionally, losing focus, and tiredness constitute the inferiority of human drivers to autopilots or computers. In addition to the consistent performance, autopilots base their decisions on more sensory stimuli than humans, and they react several orders of magnitude faster [39].

Increased safety is not the only benefit of autonomous or self-driving vehicles. These vehicles allow drivers to use the journey time for other activities and drivers can save time by activating self-parking after disembarking at their destination. Ultimately, self-driving vehicles facilitate individual mobility even for children, incapacitated seniors, or disabled people.

Autonomous vehicles incorporate software for (1) perception, (2) decision making, (3) path planning, and (4) control [135][27]. Since perception affects subsequent software modules [41], the precision with which autonomous systems perceive their environment is crucial. Perceiving requires sensors such as cameras or laser scanners referred to as LiDAR. The data that these sensors generate is processed by the perception software module aiming to detect objects in the scanned environment, including the positions, extents, and orientations of objects. Acquiring this 3D information about surrounding objects is a computer vision task targeted by scientists that study 3D object detection. Many surrounding objects represent obstacles in the vehicle's path, so perceiving them enables an autonomous vehicle to plan collision-free paths [48].

Installing sensors on autonomous vehicles has drawbacks regarding the limited sensor range, occlusions, and costs [3][50][28][25]. The limited visibility of these on-board sensors is problematic at intersections [99], where researchers find that almost half of the injuries and fatalities in urban areas occur [75]. Looking around corners or through other vehicles like large trucks could reduce hazards at intersections [92]. To extend the sensor range for vehicles and to address occlusions, the scientific community has been employing static roadside sensors in recent years that they install above the road to enable recordings from an elevated perspective. This way, roadside sensors capture a larger scene than on-board sensors [137] and complement the perception of vehicles while necessitating the processed sensor data to be sent wirelessly to affected road users. Attaching sensors to road infrastructure constitutes a key milestone for the establishment of intelligent infrastructure systems, whose collaboration with intelligent self-driving vehicles is the most promising path towards autonomous transport [15][81][134][55].

When intelligent infrastructure transmits traffic information to vehicles or people's smartphones, scientists refer to cooperative driving automation applications [3]. Aside from obstacle detection, the acquired roadside sensor data is applicable to optimize traffic flow, which decreases travel times and reduces emissions [132][106][81][28].

Sensors based on laser beams are called LiDAR and facilitate acquiring 3D information for object detection in the scanned environment. Research distinguishes between object detection with mobile, on-board LiDAR and object detection with static roadside LiDAR as both produce distinct data and pose separate challenges. Although 3D object detection with roadside LiDAR data has emerged later, it is usually implemented deploying traditional unsupervised algorithms instead of deep learning based on artificial neural networks. The dominant position of traditional algorithms in the field arises from several drawbacks that deep learning entails and that are not present in traditional algorithms.

## 1.2 Problem Statement

Ever since scientists began field-testing intelligent infrastructure in recent years, roadside LiDAR 3D object detection emerged as a new field of research. This new field of research has not yet yielded a prevalent dataset including a benchmark that all scientists use to compare results when publishing their research. In contrast, 3D object detection using on-board sensors employs the KITTI Vision Benchmark Suite [31] since 2012 to compare results. The fact that no comparable standard has yet become established in roadside 3D object detection is a result of most public datasets being released only from 2021 and thus hardly used in previous publications.

Employing new datasets to address 3D object detection based on roadside LiDAR is the objective of this study. The object detection task implemented in software adopts well-tested algorithmic techniques showcased in related research. These traditional algorithmic techniques are divided into seven modules that process their respective data input independently of each other and that developers can substitute individually. By following a modular implementation, the software differs from entangled deep learning implementations acting as end-to-end detection solutions. Another benefit of traditional algorithms is their detection performance independent of large available sample data, causing the description "unsupervised". While the newly released datasets are precious to evaluate and compare results, they are not needed to develop traditional algorithms reliably detecting objects in data. In comparison, deep learning approaches are dependent on large datasets during development.

Another difference to deep learning approaches concerns the availability of code bases. Computer scientists that develop powerful deep learning models usually publish their model to enable other scientist to use it for their research. As scientists in this field code in a common programming language and use common libraries, their software is quickly deployable. This collaborative research practice is less common for traditional algorithms as their advancement is mostly described theoretically without sharing source code to test for yourself. The few published code bases available are implemented in disparate programming languages and with varying libraries. This circumstance complicates the development of a software prototype that masters object detection.

Detecting objects precisely is already a challenging computer vision task itself. The task becomes even more challenging under real-time requirements. Real-time requirements arise from the desire of being able to broadcast warnings about new obstacles to road users within milliseconds to seconds so that they have time to adjust their planned path. To meet this desire, this study emphasizes processing speed.

## 1.3 Contribution

This research work is the first to pursue an unsupervised approach to 3D object detection within the Providentia++ project [70], a continuous intelligent infrastructure project close to Munich, Germany. As a result, this work conducts an extensive literature review and provides a comprehensive overview of the state of the art, but also outlines some early developments. Since no prior work exists within Providentia++ and few source code is publicly available in this field of research, a detection software is implemented from scratch.

Beyond the scope of object detection, an easy-to-use software with a graphical user interface emerged during the writing of this thesis, a software that can track detected objects across LiDAR frames. This software is based on code from the Aalborg University and called PyLidarTracker. The PyLidarTracker has been supplemented with the detection software mentioned above, which adds new functionalities. The software can be accessed through a link<sup>1</sup>.

Since it is crucial to quantitatively evaluate developed object detection software to enable comparisons with other solutions, an evaluation kit is contributed to the Providentia++ project.

---

<sup>1</sup>[www.github.com/marcelbrucker/Unsupervised-3D-Object-Detection-Using-Roadside-LiDAR-Sensors](http://www.github.com/marcelbrucker/Unsupervised-3D-Object-Detection-Using-Roadside-LiDAR-Sensors)



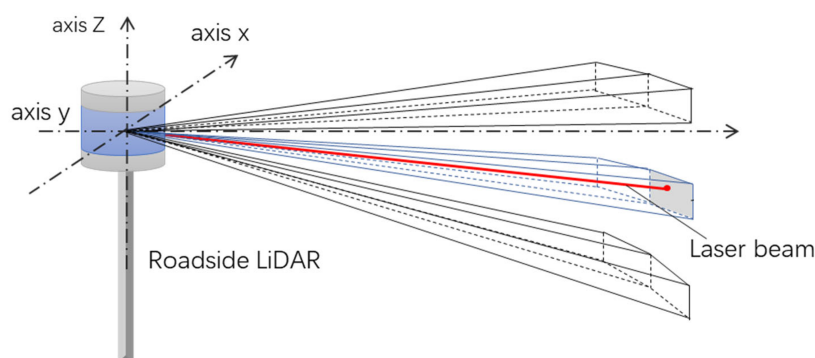
# Chapter 2

## Background

### 2.1 Sensor Hardware

Autonomous vehicles as well as other robots perceive their environment through sensors. One commonly used sensor is light detection and ranging (LiDAR) [82][98][45][72][90][53][135][41], which emits invisible laser beams and analyzes the beams that are reflected by the surfaces in the sensor surrounding. By emitting laser beams, LiDAR acts on its environment, which is why scientists classify it as an active sensor [45][71]. Scientists and the industry utilize it in various technical applications because LiDAR has several favorable properties, described in more detail in the remainder of the section. These applications include urban planning, 3D city modeling, 3D reconstruction, agricultural development, environmental monitoring, simultaneous localization and mapping (SLAM), and intelligent infrastructure systems to enhance traffic safety and flow, especially at intersections [98][53][90][126][125][55][112][116].

Self-driving vehicles use two different kinds of LiDAR: 2D and 3D LiDAR [116]. 2D LiDAR employs a single channel while 3D LiDAR has multiple channels stacked above each other, typically ranging from 16 to 128 [40]. As illustrated in Figure 2.1, these channels scan at different vertical angles to measure the height of objects in addition to their planar position and planar extent [133][62]. 3D LiDAR is superior if not required for certain applications stated above. On the other hand, it is more expensive, which is the reason researchers use their 2D counterpart for object detection as well, taking into account the very narrow field of view [116][40][125].

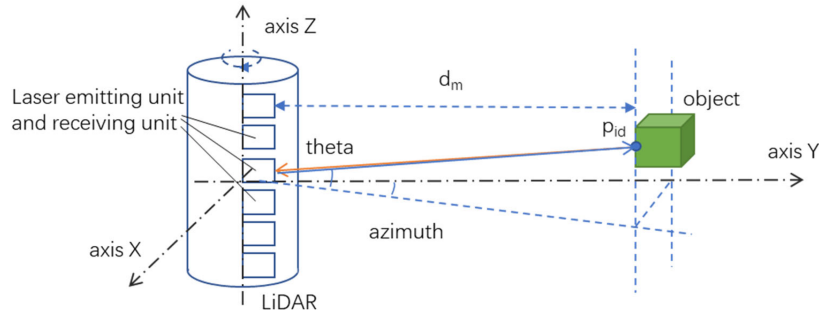


**Figure 2.1:** Stacked laser channels in a 3D LiDAR sensor scanning at different vertical angles [102].

To better understand the advantages and disadvantages of the different types of LiDAR, as well as to compare and evaluate the technology as a whole with other sensor technologies, it is crucial to know how LiDAR functions.

### LiDAR Functionality

The principal components of a LiDAR sensor are a laser transmitter and a laser receiver [126]. The transmitter emits short laser pulses into the environment, all-around in the case of a 360° horizontal field of view. The surrounding objects reflect these laser pulses back to the sensor, where the receiver detects them [86][134]. Figure 2.2 illustrates this concept. The sensor measures the time difference between the emission and the reception of each laser beam and, since they fly at the speed of light [71], one can easily compute the relative distance to the obstacle; it is the speed of light times the measured time difference divided by two as each beam traveled the distance there and back. This computation is called the time of flight (ToF) method [116][86].



**Figure 2.2:** A laser beam being emitted, reflected, and received [102].

Scientists distinguish between solid-state LiDAR and spinning LiDAR [40][71]. The solid-state LiDAR only covers a partial horizontal field of view as it never rotates, while the second kind spins around the vertical axis [134][121], enabling a wider field of view (often 360°) [92]. The typical rotational speed for spinning LiDAR is 5 to 20 Hz [137], meaning in case of 10 Hz, the sensor provides a complete measurement frame every 10 ms [40][134]. The selected rotational speed also affects the angular resolution of each frame, which is often 0.2 to 0.8° [134].

Most manufacturers state a maximum operating range of 50 to 500 m for their LiDAR systems [40], while we have to consider that the point density decreases exponentially as we move away from the sensor due to the sensor's emission of all laser beams from one single point with different vertical angles [55][106][41][137]. At some point, the vertical spacing is larger than the height of an object, meaning it is either sensed by one LiDAR channel or completely missed and thus invisible to the sensor [134].

Another important characteristic of LiDAR systems is that they only measure the distance of object surfaces facing the sensor [129][6][45][116][135]. They cannot sense what is behind the first surface blocking a laser beam's path as the light does not penetrate it but rather is reflected by the surface. Consequently, the sensor does not perceive the entire shape of an object, a phenomenon known as occlusion. Contradictory to this is the comment that laser beams do partially penetrate in case of glass and windows, respectively, causing the absence of reflections in the concerned spot.

A last characteristic to consider is laser light can also scatter when it hits the edge of e.g. walls, leaves, or branches of trees [134].

### LiDAR in Comparison

The biggest advantage of LiDAR sensing is the possibility to obtain accurate distance measurements to surrounding objects [82][135][72][106][41][129][92][99]. The sense of depth [71][92][133][126] is measured in a direct way [22][121][106], one of the reasons why the measurement is precise [101][133][62], that is, a repeated measurement of the same dis-

tance has low variance and always shows a very similar result. In addition to distance, LiDAR provides shape and scale information [34], enabling the acquisition of complete 3D information of the scene [121][6][3], even over a large field of view [135][72][6][106][62][92]. While some researchers argue LiDAR data is obtained in high resolution [133][45][40][126], others counter its sparsity causes difficulties in detection [142][134][71][50] and classification [22][126][55]. In fact, Hu et al. [41] consider the point density variations across distance the primary drawback of LiDAR, because objects farther away feature less points than closer objects. Shen et al. [82] recall that laser cannot perceive the entire object boundary as it merely detects contours facing the sensor [135], a property that requires us to infer the object pose [82]. Occlusion makes it worse since these contours are not even observed entirely when one object is hidden by another [135][41], a problem Zhang et al. [126] see as the biggest challenge in detection.

Apart from geometrical observations, LiDAR is lighting invariant, meaning the technology works just as well at night as during the day [135][72][40][133][126][62][92][141] given its self-generated light pulses [71], and overexposure due to direct sunlight does not impair perception [6][86]. However, it lacks color information as a consequence of functioning outside of visible light in the electromagnetic spectrum [22] as well as texture attributes [71]. LiDAR may compensate for that with other useful features though, features such as intensity, reflectivity, and ambient [19] that Börzs et al. [6] summarize under the term reflection properties. Moreover, missing visual features eliminate privacy concerns from the outset, since laser sensing cannot capture license plates or faces in a usable manner, for what reason we do not need to disguise [126].

Even though Sun et al. [92] allege LiDAR was not sensitive to changes in weather, other scientists reason that rain, fog, and snow do affect LiDAR data because of scattering and refraction [15][45][71][92]. Qian et al. [71] advance the argument that variations in temperature influence the wavelength stability.

Opinions on the costs of LiDAR differ; there are researchers stating that the costs are high [22][71], while others say they are relatively low [72][40] and have dropped [121][34][133][133][92], respectively.

Aside from light ranging, players in the autonomous driving domain like to use vision-based perception systems, systems with monocular or stereo video cameras [136][40]. Cameras output a dense representation of the scene, equipped with color and texture attributes and to an extent where we can recognize texts from road signs [71][116][92][3]. Similar to LiDAR, many cameras have a wide field of view [116], though not 360°. However, cameras cannot directly measure depth [126]. While stereo cameras can employ matching algorithms to align images from two different perspectives for depth recovery, the only remedy for conventional monocular cameras is depth estimation, two methods still being unsatisfactory [71][22]. Sun et al. bring up that there are depth or RGB-D cameras, but at the same time admit that their 3D data is noisy and redundant [90]. Despite the dense representation, distant objects are difficult to see which hampers detection [50][55].

Another big disadvantage of cameras is their sensitiveness to illumination [40]. At night or in the rain, their performance severely degrades, which is why cameras are mostly insufficient as standalone perception system [71][116][137][86][34][133].

On the bright side, they have a high frame rate and much lower costs [71].

Another commonly used sensor system is radio-waves based ranging or radar (radio detection and ranging) [40]. Radio waves make it possible to determine, range, angle, and velocity of objects [116][92], the latter being a unique feature among the three perception systems. Compared to LiDAR, radar has a farther scanning range [141], whereas the field of view is smaller [116] and the resolution lower [126] [92]. On top of that, radar measure-

ments are noisier, so users have to filter extensively [116]. Yet, the sensor is in demand for safety-critical applications in the automotive industry, because it is less affected by changes in weather and illumination [3][116][141].

We can summarize the main points of the passage in Table 2.1 according to Liu et al. [55].

**Table 2.1:** Performance matrix for different perception sensors [55].

Capabilities	LiDAR	Camera	Radar
Accurately localizes objects	***	**	*
Accurately classifies objects	**	***	*
Accurately measures object speed	**	*	***
Extensive field of view	***	*	**
Reliability across changes in lighting, sun, temperature	***	**	***
Ability to read signs and differentiate color	**	***	*
Privacy-secure data	***	*	***

Given many benefits, LiDAR sensing is emerging as leading sensor solution in future road transport [116][133][126][132][55][112]. In fact, Waymo, Baidu, and General Motors consider 3D LiDAR their primary sensor for perception [99]. Yet, utilizing multiple sensor modalities complementarily is convenient to ensure safety from a redundancy point of view [71], so we will briefly address how different sensors can benefit from each other.

### Sensor Collaboration

To face sparsity in LiDAR data, computer scientists can employ point cloud registration, a process stitching together data from multiple LiDAR sensors to increase point density in the scene [126]. Another potential solution is to fuse data from different sensor modalities, a solution that combines their individual strengths [136][22][71] and thus may enhance detection accuracy and prediction certainty [126][71].

## 2.2 LiDAR Output

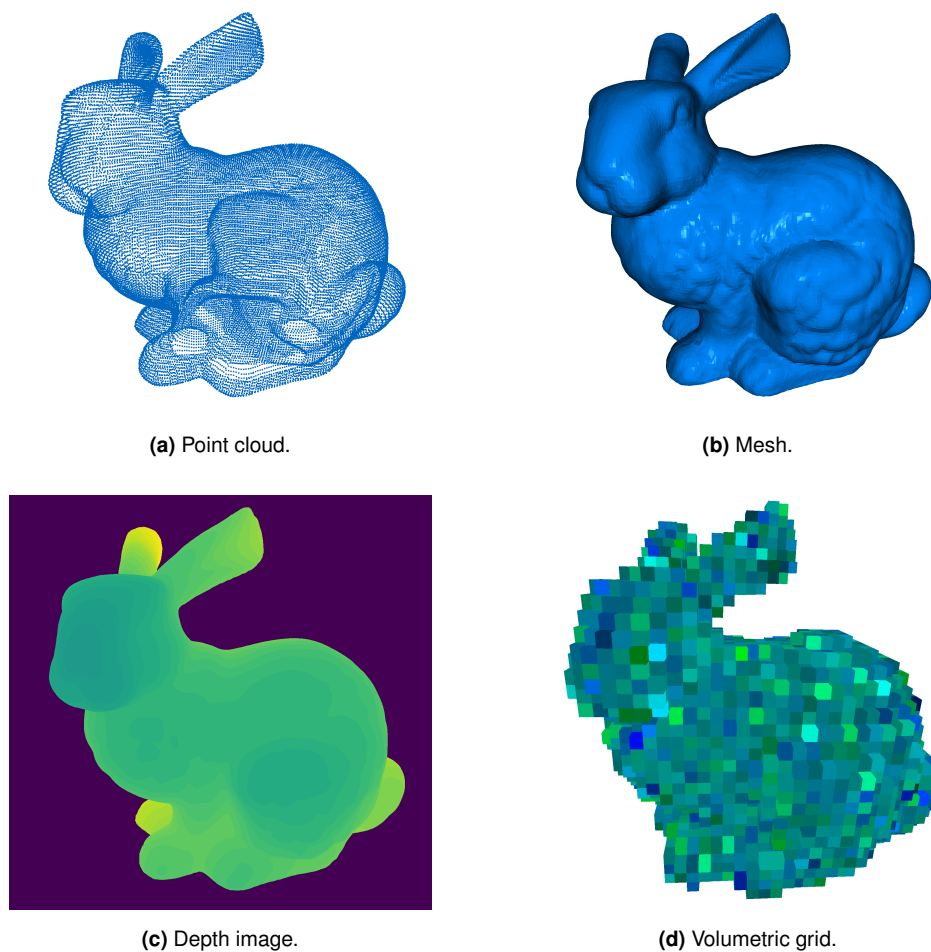
LiDAR data is temporarily divided into frames; vividly, the sensor outputs one frame after it spins one round [134], similar to a camera that also outputs a snapshot in one frame. A LiDAR frame consists of spatial points that reflected the emitted laser beams back to the sensor and that we usually view together as a point cloud. These points or the point cloud build a set of vectors  $P = \{\mathbf{p}_1, \dots, \mathbf{p}_N\}$ , where each vector  $\mathbf{p}_i = [C_i, A_i] \in P$  has 3D coordinates  $C_i = (x_i, y_i, z_i) \in \mathbb{R}^{3 \times 3}$  and optional, varying attributes  $A_i$  dependent on the used sensor model [112][51]. Attributes  $A_i$  can be the exact timestamp (points of a frame are not all recorded in the exact same moment), range, intensity, reflectivity, ambient, laser channel, or RGB values [134][19][100]. Regarding the 3D coordinates  $C_i$ , one has to specify that the raw sensor output is in spherical coordinates (range  $r$ , horizontal angle  $\phi$ , and vertical angle  $\theta$ ) and that we first need to transform them into Cartesian coordinates  $(x, y, z)$  [83][134][123][126] as shown in Equation 2.1 (MATLAB convention).

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



Aside from point clouds (Figure 2.3a), there are more formats how we can represent 3D data: meshes (Figure 2.3b); depth images (Figure 2.3c); and volumetric grids (Figure 2.3d), which are also called voxels and are the 3D equivalent of 2D pixels. The preferred format are point clouds though because they (1) preserve the geometric information in 3D space, which voxels do not due to discretization, a transformation where geometric information is lost [22], and (2) they consume less storage compared to meshes [34][71][112]. Point clouds are also preferred in autonomous driving [34] even though they are irregular spaced and sparse [123], especially in outdoor scenes [36], while vision systems produce dense illustrations [137]. Nevertheless, using an organized shape in the form of volumetric grids enables us to harness algorithms which can otherwise only be used for structured data such as images [22][53].

In comparison with images, it is important to note that both apply distinct coordinate systems. While the coordinate system in point clouds is oriented such that  $x$ =forward,  $y$ =left,  $z$ =up; for images it is  $x$ =right,  $y$ =down,  $z$ =forward [71].



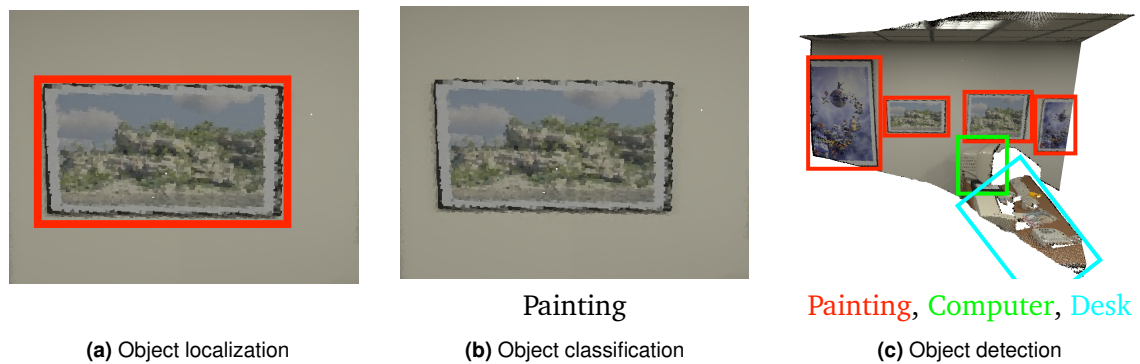
**Figure 2.3:** Four formats to represent 3D data [140].

3D data in the form of point clouds require a significant amount of space, despite the lower memory requirements compared to meshes. For instance, LiDAR with 64 laser channels produces over a million data points per second [62]. If we assume these one million points, where one point consists of at least three coordinates  $G_i$  and each coordinate is represented by a floating point number of 4 bytes, that results in approximately 12 MB per second, without taking into account the additional attributes  $A_i$  of each point. To illustrate this further, we consider PCD files, a data format to save point clouds on a computer. One frame or PCD

file in human-readable ASCII encoding contains up to 131,072 points when outputted from a Ouster OS1-64 LiDAR sensor with 64 channels [19]. This sensor model assigns each point six attributes  $A_i$  in addition to three coordinates  $C_i$ , resulting in 5.2 MB disk space per frame on a standard Linux computer. At 10 Hz, there are ten frames per second, meaning in practical figures, we generate 52 MB of data per second (!).

## 2.3 Computer Vision Tasks

Sensor data is crucial because it enables a technical system to perceive its environment. To make it truly valuable though, the system needs to process the acquired 3D data. The algorithms used are divided according to the computer vision task they perform.



**Figure 2.4:** Distinction between three computer vision task in a point cloud using 2D bounding boxes [140].

### Object Localization

Rhodes et al. [76] define object localization as the task of finding object instances within data frames. We often specify the object's pose by fitting a 3D bounding box around all points belonging to the object instance. How a 2D bounding box is fitted to a point cloud projected on a screen, is illustrated in Figure 2.4a.

### Object Classification

Assigning a predefined category to an object instance is a task called object classification [112] (see Figure 2.4b). Categories in autonomous driving are for instance cars, trucks, buses, pedestrians, and cyclists.

### Object Detection

Object detection combines the two previously mentioned tasks since we want to locate all objects in the data, estimate a bounding box around them, which represents the object's six degrees of freedom (three for translation and three for orientation), and assign it to a class [95][71][112]. Figure 2.4c is an illustration of the task, but with 2D bounding boxes. Researchers consider detection the pivotal task for 3D scene understanding because many downstream applications rely on it, applications such as autonomous driving, housekeeping robots, and augmented reality [71][112].

### Semantic Segmentation

Semantic segmentation is even more complex than object detection because it aims at assigning a class label to every point such that the whole point cloud is split into subsets each unifying points with the same semantic meaning [112][34][38].

### Object Tracking

In this task, we add time since the goal is to estimate the state of objects detected in a first frame in subsequent frames, a challenge where we need to recognize the objects [34][137]. Tracking is often attached to detection to obtain the speed and trajectory of each object, providing a comprehensive notion of the surrounding environment [135][137].

### Scene Flow

A related task with time component is scene flow which describes how each point  $p_i$  moves from its position in a point cloud  $X$  to its corresponding position  $p_i'$  in a second point cloud  $Y$  such that  $p_i' = p_i + d_i$ , where  $d_i$  is the translation that scene flow tries to determine [34].

### Domain Adaptation

The goal of domain adaptation is to generalize an algorithm developed for some computer vision task in a source domain such that it masters a novel target domain as well [115]. The issue is related to robustness of an algorithm as Qiangeng et al. [115] reveals when he demands a robust detector to cope with variable weather and different locations.

## 2.4 Paradigms of Machine Learning

Scientists address computer vision tasks with distinct algorithmic techniques which differ mainly in the extent of additional manual data preparation. Modern techniques are summarized under the term machine learning.

### Supervised Learning

Developments in the field of computer vision often apply supervised learning, a paradigm requiring extensive human labor in data preparation because the collected data frames need to be labeled manually. Labeling refers to the activity of annotating - in the case of object localization - where objects are found in data and in this way creating a dataset. The labeled dataset is essential because supervised learning deploys deep neural networks which need a training phase before they can perform a task [112]. After training, the artificial neural network extrapolates the gained knowledge to infer e.g. where objects instances are located on unseen and unlabeled data. Computer scientists have been developing a large number of network architectures and models in recent years underlining the progress of the technique [112]. However, the drawback of supervised learning persists, which is the substantial effort of manually labeling thousands of example frames to ensure training of a powerful deep neural network. Labeling is even more time-consuming in the case of 3D data because of the third dimension [112][114].

### Unsupervised Learning

There is no additional manual data preparation involved for unsupervised learning techniques as they do not require labeled data [112]. This is the most succinct difference from the other paradigms even though unsupervised methods require more elaboration. Further characteristics are detailed in Chapter 3.

### **Semi-Supervised Learning**

In semi-supervised learning, researchers train a deep neural network merely on a small quantity of labeled data [112]. After this initial training, researchers use the deep neural network to generate pseudo-labels for a large quantity of data with which they let the network continue training. Proceeding this way reduces the effort of human annotations compared to supervised learning.

### **Self-Supervised Learning**

This paradigm is a subset of unsupervised learning as self-supervised learning models train on labeled data generated by the model itself, so humans do not need to annotate anything [112].

### **Weakly-Supervised Learning**

Researchers focusing on object detection refer to weakly-supervised learning when they use data where the only label given is a tag or keyword describing an entire frame without detailing individual object instances in the frame [93].

Unsupervised, weakly-, and semi-supervised learning techniques limit the need for human labeling [95]. Gathering large-scale labeled point cloud data efficiently is one considerable drawback of alternative supervised approaches using deep learning models and cause of this research to employ unsupervised learning to accomplish 3D object detection.

## **2.5 Intelligent Infrastructure**

We can install sensors for environmental perception in three possible locations: on the road vehicle, in an air vehicle, or stationary on infrastructure [86][47]; the additional sensing equipment upvalues the latter to an intelligent infrastructure system, also called intelligent transportation system.

Sensors on the road vehicle are also referred to as being mobile or on-board, because that is, where we will finally need their generated data. These sensors perceive the environment from a vehicle perspective, yet not covering the entire surroundings due to a limited field of view and occlusions [47]. Occlusion refers to the situation when one road user obscures another [137], causing invisibility to the sensor, as sensors do not see through objects. The obscuring effect occurs particularly with trucks and buses because of their height and large surface [137]. Despite the visual disadvantages, scientists and car manufacturers usually attach sensors directly to the vehicle [99].

A second potential location are air vehicles that monitor traffic from a bird's eye view, vehicles that do not suffer from limited visibility and occlusion [47]. While air vehicles solve these two problems, they have a limited recording time and have to land often to recharge or to refuel, which diminishes the applicability of this approach [47]. Additionally, air vehicles cannot operate in places like tunnels.

The third option is to install sensors above the road on infrastructure such as traffic lights, lampposts, or gantry bridges and broadcast their information via vehicle-to-infrastructure (V2I) communication [25], a procedure that helps avoid the drawbacks of the previous two alternatives [47][3]. Occlusion in dense traffic is minimal [3][99] and can be further reduced when recording the same scene from different elevated perspectives [137]. However, installing sensors on stationary infrastructure requires a considerable initial outlay [47][133] since technicians have to safely fix the technology in moving traffic using expensive lifting equipment. In addition to the sensors themselves, the technicians usually lay cables to a local

computing unit that transmits the sensor data over the internet. While fixing the sensors at high position that are difficult to reach means effort, it prevents tampering [3]. After setup, sensors record traffic events perpetually [47], as long as no malfunctions occur or any sensor fails, both of which happen and causing high maintenance costs, according to Zhang et al. [133]. Wang et al. [99], in contrast, argue stationary sensors benefit from less vibration in surface mines concerning maintenance and operational life. The authors further state that they can protect infrastructure sensors more easily from harsh environmental conditions such as temperatures of  $-40^{\circ}\text{C}$  that can be found in some Chinese mines.

Autonomous vehicles can never rely solely on infrastructure sensors because vehicles must also function in the event of signal interference and disconnections from the external sensors. Infrastructure sensors, also called roadside sensors or roadside units (RSU) [99][106][55], complement on-board sensors by mitigating the impact of occlusions [50][81] and increasing the autonomous vehicle's perception range [99][81][25]. Increasing the perception range is especially useful when an autopilot needs to hand over responsibility to the driver because the system does so with more lead time if it can anticipate earlier that a situation is too complex [81]. This argument is supported by accidents related to autonomous vehicles from Google and Tesla, which failed to detect road users, tragedies leading us to the consensus that on-board sensor technology is not sufficient and supporting intelligent infrastructure is indispensable [55]. Both information sources together provide the maximum dependability, integrality, and credibility of environmental perception [55].

When working with both sensing procedures, we need to consider their differences. First, on-board sensors move, meaning they perceive changing surroundings while roadside sensors are installed at fixed locations and always observe the same scenery [121][86][92][126]. This trivial insight has considerable consequences. Second, roadside sensors have a greater distance to obstacles in the close proximity of a vehicle compared to its own on-board sensors [121][126]. This means in the case of LiDAR, we receive sparser data due to larger spacing between neighboring laser beams at a farther distance. Last, given the greater distance, a roadside sensor scans a larger area [126], so there are more perceivable obstacles.

Operators of intelligent infrastructure systems typically equip intersections with the technology or install the sensors along roads such that they can record high-resolution micro traffic data (HRMTD) at a frequency of more than 10 Hz [106]. According to Song et al. [86], HRMTD enables algorithms to determine trajectories of road users for near-crash analyses and dynamic traffic signal timing. Other authors emphasize that LiDAR was key to advance traditional macroscopic traffic data acquisition to a microscopic level, and that LiDAR can perform traffic monitoring completely independent of cameras [133][106].

Since roadside LiDAR sensors are installed at a fixed location, the majority of points from one sensor appear in all of its recorded frames, points that we consider background. Only a small subset of points from one frame are reflections from dynamic objects that are normally of interest and that we call foreground. We can exploit this characteristic to reduce the computational load for computer vision algorithms compared to mobile LiDAR when we identify and filter background points in point clouds [86]. The same characteristic causes problems when we deploy supervised learning for our intended computer vision task because the broad homogeneity of data makes deep neural networks generalize poorly in the training phase [99]. Deep neural networks require diverse, large-scale training data to reliably master new, unseen data and data is not diverse if the vast majority of a 3D point cloud - the background - never changes. When a model is too used to its training data, computer scientists like to call the defective training "overfitting", which is an issue also for domain adaptation because such an inflexible model will suffer from performance degradation on data from other sensor sites [99]. Wang et al. [99] conclude that overfitting hinders the use of powerful deep learning models with data from stationary sensors, which is the reason this

investigation targets methods where the model cannot accustom itself to data due to the lack of learning.

Regarding the sensor choice for intelligent infrastructure systems, there are two practical solutions: LiDAR and video cameras. According to Zhao et al. [137], however, camera setups require more computing resources and power, and operators need more sensors to equip an entire intersection such that all angles and distances are covered and there are no blind spots. Other researchers agree when stating LiDAR shows great potential for future intelligent infrastructure [126][99][133]. Zhang et al. [126] are convinced that these systems, along with next-generation communication networks, will process and transmit 3D LiDAR data in real-time achieving dependable cooperative autonomous driving. This development may even allow car manufacturers to equip vehicles with less sensor modalities as indicated by Fleck et al. [25].

## 2.6 Intelligent Infrastructure Projects

There were already projects 15 years ago where research groups equipped road infrastructure with sensors and communication channels to broadcast information about the traffic [25]. One early project called PRE-DRIVE C2X started 2008 and wanted to establish a vehicle-to-everything (V2X) communication architecture in Europe, but those familiar with the topic did not observe a significant expansion of intelligent infrastructure projects until 2019 [25].

At about this time, the Test Area Autonomous Driving Baden-Württemberg emerged, intelligent infrastructure at an intersection in Karlsruhe, Germany, equipped with two cameras and connected to the traffic lights that detects and tracks road users and publishes the result online via V2X communication [25].

Scientists in the United States in 2019 installed 16-channel LiDAR sensors at three intersections in Reno, Nevada with communication devices [137]. The authors around Junxuan Zhao decided to mount more than one sensor at different corners of an intersection to address occlusion.

In Ulm, Germany, a team of Robert Bosch GmbH together with the Karlsruhe Institute of Technology launched 2019 MEC-View, an intelligent infrastructure project, where they use cameras and LiDAR to investigate how fusing on-board and stationary sensor data can foster perception and tracking of traffic participants [28].

In 2020, the RWTH Aachen University deployed up to eight 64-channel LiDAR sensors at a German intersection to examine the benefit of fused point clouds over a single point cloud for multi-object detection and tracking [47]. Meanwhile, the project has expanded the sensor infrastructure to over 100 LiDAR sensors [87].

Providentia is a continuous intelligent infrastructure project close to Munich, Germany, where the project stakeholders started to equip gantry bridges on the A9 highway with cameras and radars [50]. Their research endeavor is to construct a digital twin of the current traffic, which is an accurate digital model composed of multimodal sensor detection, and share it live over the internet, particularly to extent the perception range of autonomous vehicles. The digital twin can also be employed for traffic flow management or warnings about wrong-way drivers though. In addition to traffic perception on the highway, the team expanded their system to rural roads and an inner-city intersection in the follow-up project Providentia++, where they add LiDAR to their sensor selection. Figure 2.5 illustrates the equipped gantry bridge S110 of the inner-city intersection.

Since this work is embedded in the Providentia++ project and will mainly use their LiDAR data, important hardware details are listed explicitly in Table 2.2.



**Figure 2.5:** Equipped gantry bridge near Munich, Germany as part of the Providentia++ project.

**Table 2.2:** LiDAR sensor specification in the Providentia++ project [19].

Characteristic	Ouster OS1-64 Gen 2	Valeo SCALA 2 Gen 2
Number of channels	64	16
Sensing range (80% reflectivity)	100 m (>90% detection probability)	150 m (100% detection probability)
Precision	7 mm to 50 mm	<100 mm
Vertical field of view	22.5° (below horizon)	10°
Horizontal field of view	360°	133°
Vertical angular resolution	0.26° to 0.52°	0.6°
Horizontal angular resolution	0.18° to 0.7°	0.125° to 0.25°
Scan rate	10 Hz	25 Hz
Points per scan	131,072	17,024

## 2.7 Datasets

Up to this day, there are few real-world datasets from stationary LiDAR as this sensor technology became suitable for widespread use in transportation in the last couple of years, in contrast to cameras. Also, LiDAR is mostly installed directly on the vehicle (on-board) [99][3][47], whereas this study requires LiDAR data from non-moving sensors. On-board LiDAR and stationary LiDAR generate distinct data for a number of reasons: (1) stationary LiDAR typically records from a higher position above the ground, which changes the perspective on road users; (2) stationary LiDAR is farther away from road users, meaning the point density per object is lower; (3) vehicles in data from on-board LiDAR are usually oriented identically and point in the same direction of travel as the ego vehicle, a data bias that is less common with stationary LiDAR [126]; (4) only a non-moving sensor produces data where we can benefit from the large number of uniform background points between recorded frames. For these reasons, prevalent datasets from mobile LiDAR cannot be used, datasets such as KITTI [30], Waymo Open [91], or nuScenes [11].

Strigel et al. [89] in 2014 published the first dataset using stationary LiDAR sensors called Ko-PER. After seven years of hiatus, the scientific community released further roadside

datasets in 2021 and 2022: IPS300+ [100], BAAI-VANJEE [21], Rope3D [117], DAIR-V2X-I [118], LUMPI [10], and the A9-Dataset [19] recorded from the sensors belonging to the hardware setup of the Providentia++ project [50] close to Munich in Germany. The A9-Dataset stores each point cloud scan in a PCD file with ASCII encoding. Each point cloud scan consists of maximum 131,072 points, each of which providing the attributes x, y, z, intensity, timestamp, reflectivity, ring, ambient, and range. The dataset stores the object labels or ground truth in corresponding JSON files, meaning each JSON file is the counterpart of a PCD file with identical file name but different file suffix. A JSON file contains the timestamp of the recorded point cloud frame, the weather type, and the object labels, which provide a unique object ID, the object class, bounding box information (3D object center location, 3D box dimensions, and orientation expressed in quaternions), as well as additional attributes. Since this study uses the A9-Dataset to develop an object detection method, the dataset is described in greater detail. The most important aspects of all listed dataset, however, are specified in Table 2.3.

**Table 2.3:** Real-world datasets with point clouds from stationary 3D LiDAR sensors.

Name	Year	Location	Point Cloud Frames	Classes	Diff. Weather/ Time	Obj./ Frame
Ko-PER [89]	2014	Germany	4850	4	No/No	-
IPS300+ [100]	2021	China	14k	8	No/No	320
BAAI-VANJEE [21]	2021	China	2500	12	Yes/Yes	30
Rope3D [117]	2022	China	50k	13	Yes/Yes	30
DAIR-V2X-I [118]	2022	China	10k	10	Yes/Yes	49
LUMPI [10]	2022	Germany	90k	6	Yes/Yes	-
A9-Dataset [19]	2022	Germany	7559	10	Yes/Yes	34

## 2.8 Point Cloud Libraries

One commonly used library to process 3D data is the point cloud library (PCL) [79], which was launched in 2011 and is based on the programming language C++. Computer scientists built a community around the open-source framework and contributed a large quantity of algorithms in its first years of existence [140]. While PCL was developed actively further during that time, contributions and maintenance declined significantly in the meantime turning PCL into a bloated and outdated software tool [140].

In 2018, Zhou et al. [140] published a new open-source library to process 3D data: Open3D. Just like PCL, Open3D is based on C++, but provides an interface for the programming language Python as well. The authors state that they only add selected data structures and algorithms and limit the number of dependencies. This consideration ensures performance and clean code.

## 2.9 Evaluation Metrics

Scientists that research object localization or object detection assess their developed software solutions quantitatively with an evaluation metric called average precision (AP) and mean average precision (mAP), respectively. These metrics are figures stating how well the software

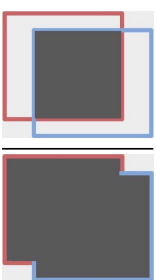


localized or detected objects in the recorded data. More specifically, the metrics indicate how many of the actual objects were found, between 0% and 100%. The actual objects are stored as labels in a dataset that is used for the evaluation and that is always necessary, even if an unsupervised approach was targeted during development. The information where which objects actually are is defined (labeled) once manually and then regarded as true, which is why scientists also refer to this information as ground truth. Scientists compare the dataset containing this ground truth to the predictions of their developed software when they evaluate.

Determining the average precision for evaluation involves some intermediate computations. Frame by frame, scientists first compute how well each predicted object bounding box overlaps with the ground truth bounding boxes, a procedure for which they use the intersection over union (IoU) in 3D.

### Intersection over Union 3D

The intersection over union in 3D states how well two 3D bounding boxes overlap. An  $IoU = 0$  means they do not overlap while  $IoU = 1$  means they match exactly. Scientists express the relation in a formula in which the volume of intersection is divided by the volume of union. By way of illustration, the 2D equivalent is shown in Equation 2.2 whereby it is worth noting that the corresponding calculation in 3D is more complex and only the reference to [32] from the PyTorch3D [74] library is given here.

$$IoU = \frac{\text{area of intersection}}{\text{area of union}} = \frac{\text{area of intersection}}{\text{area of union}} \quad (\text{Based on [1]}) \quad (2.2)$$


The computed IoU for two 3D bounding boxes indicates a correct localization if the value exceeds a predefined threshold, for example 0.5. For a correct detection, the predicted class label of the bounding box in question has to match the ground truth class label additionally. Refer to Zeng [119] for concrete threshold values that are used with popular datasets.

### True Positive, False Positive, False Negative

Taking into account the IoU, the predefined overlap threshold, and for detection the class label, scientists determine three intermediate metrics: true positive (TP), false positive (FP), false negative (FN). For completeness, there is also true negative (TN), but this metric is not required for the subsequent computations.

- True positive (TP): the software predicted an object bounding box at some position (positive) and this prediction is correct (true) considering the ground truth [1].
- False positive (FP): the software predicted an object bounding box at some position (positive) but this prediction was wrong (false) [1].
- False negative (FN): the software did not predict an object bounding box at some position (negative) but that was wrong (false) as a ground truth bounding box exists at this position [1].

- True negative (TN): the software did not predict an object bounding box at some position (negative) and this prediction is correct (true) [1]. This case applies to the major part of a point cloud and is irrelevant for further computations.

How scientists handle bipartite matching [42] between classes with a confusion matrix is explained excellently by Anwar [1]. The author also illustrates how difficult cases are addressed to obtain TP, FP, and FN, cases such as (1) the IoU exceeds the threshold but the class label is wrong or (2) there are multiple predictions for one ground truth bounding box.

### Precision and Recall

Assuming TP, FP, and FN are obtained, scientists compute the precision and the recall for each class. The precision in Equation 2.3 states how many detected, for instance cars, are truly cars [1] according to the ground truth. In turn, the recall in Equation 2.4 states how many of the ground truth cars was the software able to detect [1].

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.3)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.4)$$

A perfect prediction achieves precision = 1 and recall = 1, meaning all made predictions are correct and the software can detect all the occurrences of a class [1].

### Precision-Recall Curve

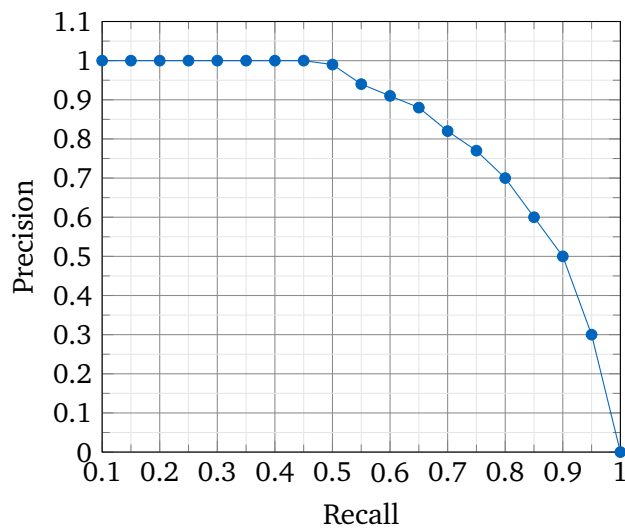
The precision-recall curve  $p(r)$  as in Figure 2.6 displays the precision against the recall for varying confidence thresholds in one class [1]. Varying confidence thresholds and their effect on precision and recall are depicted in Table 2.4. Confidences appear in a detection task with multiple classes when the software, for example based on an artificial neural network, outputs the probability that a particular bounding box belongs to class “pedestrian”, for instance [1]. While one class is usually more likely than others, the probability distribution is not Pedestrian = 1 and Other Classes = 0, but rather Pedestrian = 0.8 and Other Classes = [0.0; 0.2] to remain in the example [1]. Hence, the developers of a software select a confidence threshold that the software must exceed to assign a class to the bounding box. A small confidence threshold increases the number of predicted classes for a bounding box, resulting in more ground truth labels matched and a higher recall [1]. However, a high confidence threshold tends to lead to a single predicted class for a bounding box, reflecting the prediction confidence of the software and resulting in a higher precision [1]. Achieving high recall and high precision is a conflict of goals for an imperfect detection software, so scientists have to weigh between the two goals using the confidence threshold [1]. To find a suitable threshold graphically, scientists employ the precision-recall curve.

### Average Precision

Despite the precision-recall curve, selecting the confidence threshold appropriately is challenging and subjective [1]. To mitigate the dependence of the threshold selection on quantitative results, the average precision serves as objective performance indicator [1]. The average precision value between zero and one (or 0% and 100%) for a single class corresponds to the area under the respective  $p(r)$  curve, expressed in Equation 2.5 [1][42]. The average precision is close to one when when both, precision and recall, are high [1]. One of these metrics being low results in an average precision closer to zero [1].

**Table 2.4:** Precision and recall for varying confidence thresholds based on [1].

Conf. Th.	Recall	Prec.	Effect
0.95	0.1	1	More FN
0.9	0.15	1	
0.85	0.2	1	
0.8	0.25	1	
0.75	0.3	1	
0.7	0.35	1	
0.65	0.4	1	
0.6	0.45	1	
0.55	0.5	0.99	
0.5	0.55	0.94	
0.45	0.6	0.91	
0.4	0.65	0.88	
0.35	0.7	0.82	
0.3	0.75	0.77	
0.25	0.8	0.7	
0.2	0.85	0.6	
0.15	0.9	0.5	
0.1	0.95	0.3	
0.05	1	0	More FP

**Figure 2.6:** Precision-recall curve  $p(r)$  based on [1].

$$\text{Average Precision (AP)} = \int_{r=0}^1 p(r) dr \quad (2.5)$$

Scientists compute the average precision practically by approximating the integral in Equation 2.5, for instance, through interpolation and eleven average values [1]. The approach called 11-point average suggests calculating eleven precision values for the corresponding eleven recall values from 0.0 to 1.0 with a 0.1 increment between them [1]. Computing the mean of these eleven precision values according to Equation 2.6 yields the average precision [1].

$$\text{Average Precision (AP)} = \frac{1}{11} \sum_{\substack{r=0.0 \\ \text{step } 0.1}}^{1.0} p(r) \quad (2.6)$$

### Mean Average Precision

Scientists determine the average precision for each class in object detection, so they need to compute the mean across all  $k$  classes to obtain the mean average precision (mAP) reflecting the overall performance of their developed software. To obtain the mean average precision, they apply Equation 2.7.

$$\text{mean Average Precision (mAP)} = \frac{1}{k} \sum_i^k AP_i \quad (2.7)$$

# Chapter 3

## Related Work

### 3.1 3D Object Detection

An object detector is the model that performs object detection, meaning it inputs the point cloud generated by the LiDAR sensor and outputs 3D bounding boxes around all detected and classified objects in the point cloud [34][135][71]. While computer scientists refined 3D object detection substantially, the computer vision task still has not gained comparable importance to object detection in 2D images in the scientific community [71][114]. This observation can be derived, according to Xie et al. [114], from rudimentary procedures applied in 3D detection compared to 2D detection. The authors argue that deep neural networks for 3D detection are often trained from scratch instead of utilizing more advanced techniques like transfer learning.

One challenge in 3D object detection using point clouds is that small objects, such as pedestrians or cyclists, reflect few laser beams back to the sensor, resulting in few points that represent these small objects in the data [41]. This could be the cause for Hu et al. [41] to find that most developers of object detectors disregard these vulnerable road users and primarily seek to detect vehicles. This imbalance must be reduced, they appeal, by boosting multi-class object detection.

A second imbalance in object detection criticized by the scientific community is the limitation to a single domain when detectors are developed and evaluated [115]. Developers work in a single domain if their data is recorded by a single sensor, at the same location, the same time of the day, or under similar illumination and weather conditions. 3D detectors generalizing well across variable environmental conditions, still have not been discovered [115].

How well object detectors generalize is an issue related to robustness. A robust object detector performs well not only under variable environmental conditions, but also copes with poor visibility due to occlusion or glare, or defective data resulting from transmission errors or sensor noise [141][142]. Dealing with such adverse circumstances, is a key task when developing an object detector [141][142].

This study distinguishes between detectors applying supervised machine learning and those applying unsupervised machine learning. After briefly describing the first type of detectors, its deficiencies are pointed out, leading us to approaches that deploy the more promising unsupervised learning paradigm.

#### 3.1.1 Supervised 3D Object Detection

Supervised object detection using 3D data has made substantial progress since 2018, observable in numerous deep learning models created since that time [114][71][98][36]. As LiDAR

data is irregular spaced and sparse, successful deep learning techniques, originally developed for processing structured image data, cannot be applied directly [36][99]. These techniques are particularly artificial neural network layers performing convolutions, a mathematical operation to find spatial correlations in images [71]. Neural networks using such layers are called convolutional neural networks (CNN), networks that researchers can deploy to process irregular LiDAR data if they discretize it beforehand into structured voxels [99][36]. For a detailed explanation of convolutions and CNNs, the author refers to excellent available online references [68].

Computer scientists divide supervised 3D detectors into categories. First, they categorize these detectors into region proposal-based methods and single shot methods [34][95]. The second division with regards to the input data format is less universal and this overview limits itself, deliberately neglecting some minor ones, to the largest categories: bird's-eye-view (BEV)-based methods, voxel-based methods, and point-based methods.

### **Region Proposal-Based Methods**

Object detectors applying these methods process data in two steps. First, they propose regions that presumably contain an object. In the second step, the object detectors assign each of these proposals to a class. Region proposal-based methods are more common than single shot methods and in 2020 performed significantly better on the popular KITTI vehicle detection benchmark [34].

### **Single Shot Methods**

Detectors based on single shot methods directly fit bounding boxes around object instances and classify them in one step [95][34]. These detectors using a neural network architecture called single-stage [95] work faster than detectors using region proposal-based methods [34].

### **BEV-based Methods**

These methods input data from a bird's eye view meaning the 3D data was projected on a 2D plane such that we can utilize effective detector models developed for 2D images [34][99].

### **Voxel-Based Methods**

Voxel-based methods preprocess the irregular point cloud data into a volumetric grid representation to empower themselves to perform convolutions in 3D [71][41][12][34]. Through the use of this mathematical operation, object detectors process data more efficiently [71]. However, the preprocessing step curtails the fine-grained spatial accuracy of the data due to discretization [71][41]. Voxel-based methods tend to perform better than point-based methods as long as the discretization is not chosen too coarse [41].

### **Point-Based Methods**

As the name suggests, point-based methods input the raw point cloud for the object detection task without preprocessing [41][34].

For the time being, computer scientists are still trying to discover a powerful neural network architecture for 3D object detection that may become as ubiquitous as some architectures in 2D object detection with images [112]. The neural network architectures that exist for 3D object detection are trained from scratch, meaning only with data from the domain for which the neural network is developed [114][112]. Superior approaches are uncommon, approaches that transfer knowledge from other neural networks, which were already trained in a similar domain with a potentially larger dataset, and only finetune the network with

the target data [114][112]. Not using these superior approaches mainly stems from one reason: there is no 3D dataset containing millions of labeled example frames as in the case of images, because acquiring 3D data and labeling it is more costly [114][142][34]. Xie et al. [114] address the issue by suggesting unsupervised pretraining with unlabeled data and subsequent supervised fine-tuning for 3D scene understanding of which object detection is an integral part. Xiao et al. [112] also identify the potential for unsupervised learning in computer vision with 3D data and encourage more research than has been done in this field until now.

Aside from the serious lack of large-scale labeled data, 3D detectors based on supervised learning embody more deficiencies. For instance, they struggle to detect objects which are far away according to Guo et al. [34], although we can assume this originates from the data and affects all object detectors, no matter of what type.

Another deficiency of these detectors is their vulnerability to adversarial attacks [71]. Providing access to sabotage, poses a severe security risk [71] and could result in fallacious detections, which may ultimately be used for maneuver planning. This scenario developed further could cause traffic accidents, exactly what we are trying to prevent.

Fallacious detections can also emerge because of another property that lies in the nature of neural networks: their lack of explainability. This property refers to the notion artificial neural networks are a black box that not even their creators can penetrate [9]. They know which mathematical operations the network executes to reach the result it outputs when provided with input data, but the creators cannot formally prove the network's performance. This fact impedes that authorities approve artificial neural networks at critical points such as the perception of self-driving cars.

The mathematical operations a neural network executes are numerous. Convolutions as well as billionfold additions and multiplications per second are needed to train the neural network in a decent amount of time. Large models require weeks of nonstop training to converge to a satisfactory performance level, even on high-performance computers specially designed for this task. This amounts to substantial computational resources necessitated when employing deep learning and high energy consumption [80].

Finally, supervised learning not only demands a large quantity of labeled data, but also diverse data to learn effectively. Data is not diverse though when recorded with stationary sensors because the background never changes and background points constitute the majority in a point cloud. Consequently, a deep neural network overfits the uniform data from stationary sensors in the training phase and will fail to master new, unseen data, which ultimately hampers the use of supervised deep learning approaches [99][126][67].

Diversity is also an issue regarding the frequency with which an object class appears in the data. Autonomous driving datasets usually reveal that the object class "car" is over-represented compared to buses or bicycles. Classes less represented are learned worse by artificial neural networks with adverse effects on the detection performance turning imbalances between classes into a severe challenge in deep learning [67].

To summarize, a universal sophisticated neural architecture for supervised 3D object detection does not exist. Neural networks necessitate large datasets to master a computer vision task. They are prone to adversarial attacks and lack explainability. Training a neural network consumes a considerable amount of energy. These deficiencies combined with the special problem of uniform data in the case of stationary sensors prevents this study from pursuing this research direction. The question arises if there is a more promising direction.

### 3.1.2 Unsupervised 3D Object Detection

As supervised object detection based on deep learning entails multiple drawbacks, especially when using static roadside data, studies on intelligent infrastructure have been mainly deploying other methods. These methods used for detecting objects in point clouds involve one step where redundant points are discarded, points from the ground, from trees, walls, lampposts, and any other static background fragments that do not belong to dynamic objects of interest [126] [101][15][55][132][99]. Researchers in the field consider the remaining points of the first processing step the “foreground”, which are passed to the second step. The second step is usually to cluster the foreground points into distinct clusters. Fitting bounding boxes around these clusters is often the final step in the unsupervised detection pipeline as these methods focus on localizing vehicles. Extending the focus towards smaller road users and classifying the road users is not considered in many respective studies. While many studies neglect the classification step, they often complement their detection method by attaching an algorithm to track vehicles over time [57][105][88].

Since these unsupervised methods do not learn but rather represent traditional algorithms, we can call them traditional methods as well. Traditional algorithms are characterized by a rigorous possibility to explain how they function internally, as opposed to deep learning models, and some of them have existed for a long time already. The traditional methods used in 3D object detection are only outlined here, because we will discuss them in detail in the following sections. The remainder of this passage lists a brief chronology of complete object detection pipelines that apply traditional algorithms.

Beforehand, it should be noted that results of related research in this and in subsequent sections are quoted sparsely. Results given should be regarded with caution. This is due to the absence of a common benchmark that all researchers in the field of unsupervised object detection use to publish results with which others can compete and compare their own developments. In contrast, research on 3D object detection applying supervised learning usually evaluates on the KITTI Vision Benchmark Suite [31], so progress becomes immediately quantifiable and provable. A related issue is that, with one exception, there were no publicly available datasets on roadside LiDAR object detection until 2021, for what reason even recent publications hardly use the new datasets. Instead, the studies listed used their own non-public datasets, which impedes comparability between the presented results.

A recent survey paper [3] about object detection using infrastructure sensors encountered first detectors dating back to 2005. However, most studies in the field were published in the last couple of years.

In 2015, Börzs et al. [6] presented an object detection method consisting of seven steps: (1) discretizing the point cloud into volumetric grid cells, (2) discarding grid cells that contain less points than a specified threshold, (3) identifying grid cells that belong to the ground if the respective difference between the maximum and minimum height of points remain under another specified threshold, (4) finding objects in the remaining foreground by computing features like the point density in each grid cell and its neighboring grid cells and merging them if they match, (5) refining the result with a finer grid (6) fitting bounding boxes, and (7) using further hand-crafted features for detecting vehicles in a binary (vehicle or non-vehicle) classification step.

In 2018, when the scientific community began to publish studies more frequently, Kampker et al. [44] proposed a framework for object detection and tracking in urban areas. The authors first segment the ground with a method from previous research [38] (see Section 3.2). Then they cluster non-ground points projected on a 2D plane with Connected Component Clustering [77]. In their final object detection step they fit bounding boxes in 2D with the minimum area rectangle (MAR) algorithm [26], while retaining the height to later



complete the 3D bounding box, and an L-shape fitting algorithm to correct the result of MAR.

In 2019, Chen et al. [13] implemented an innovative procedure to detect crossing deers alongside conventional road users as these animals threaten road safety in rural areas. The authors extract the foreground using a background filtering algorithm of their previous research called 3D-DSF [106] (see Section 3.3). After clustering the extracted foreground into point groups with DBSCAN, Chen et al. investigate different classifiers to distinguish between vehicles, pedestrians, and deers (see Section 3.8). Their algorithm detects deers on average up to 31 m away and requires roughly 200 ms in one frame for this task.

In the same year, Zhao et al. [137] combined a set of algorithms for object detection and tracking. Regarding object detection, they (1) extracted the foreground by a statistics-based background filtering algorithm, (2) clustered the obtained foreground with DBSCAN, and (3) differentiate between vehicles and pedestrians employing a supervised neural network.

Also in 2019, Cui et al. [20] targeted real-time high-resolution micro traffic data for cooperative driving automation. Before tracking the road users, the authors detect them by means of background filtering with 3D-DSF, clustering the resulting foreground point cloud with DBSCAN, and finally classifying object clusters with a random forest algorithm, which is based on decision trees. Additionally, the processing chain from Cui et al. identifies lanes, so that the detected vehicles are assigned a lane. The outcome of the processing chain is visualized in a smartphone application.

Finally, Zhang et al. [133] investigated vehicle detection and tracking in complex traffic situations without fitting bounding boxes. In addition to this novel approach, the authors suggested to fuse consecutive point cloud frames to cope with occlusion.

In 2021, Gong et al. [33] addressed pedestrian detection in real-time by combining traditional and deep learning methods. The authors store the point cloud in an octree to rapidly trace spatial relations between points when extracting a predefined region of interest. Octree is a data structure in the form of a tree, where each tree node has eight successor nodes. Scientists use the data structure to recursively subdivide 3D space into octants [73]. Inside the extracted region of interest, Gong et al. identified related points through Euclidean clustering with an adaptive search radius [3].

Also in 2021, Zheng et al. [139] filtered the background of their roadside 3D LiDAR sensor with mean background modeling (MBM) and a background difference method (BDM). To refine and to cluster the extracted foreground point cloud they suggest hierarchical maximum density clustering of application with noise (HMDCAN). The extracted foregrounds are 97% accurate on average, according to the authors.

In 2022, Nagy and Rövid employed a high-definition (HD) map to extract a region of interest. Points inside the extracted region of interest are projected onto the road surface and grouped with Euclidean clustering. Around each cluster, the authors fit a 3D bounding box by determining the convex hull and computing a minimum area rectangle. Their algorithm detects objects up to 150 m away and needs 50 ms to process one LiDAR frame.

After this overview about state-of-the-art 3D object detection methods, this study wants to detail traditional algorithms identified in the literature research in a logical order. Logical means from the first step in the processing chain of a point cloud to the last.

## 3.2 Ground Segmentation

Unsupervised object detection in roadside LiDAR data begins the processing chain with algorithms that discard a great number of points not belonging to any sought road user. These discarded points depict static objects such as the ground, trees, walls, or lampposts and ac-

count for the majority of points in a point cloud. Only few points portray the road users that we seek to detect and that we consider dynamic objects [133][99]. Among those redundant static objects, some researchers first discard the ground in a separate processing step of the point cloud, a processing step they call ground segmentation (also: ground filtering, ground removal, ground extraction, or ground separation). Other researchers remove ground points as part of a background filtering step, where also points from static objects like trees, walls, or lampposts are eliminated. Lastly, some studies are conducted at sites where background filtering is not necessary because there are no other static objects nearby and thus a processing step to remove the ground is sufficient. Whether a separate ground segmentation step is advantageous or can be replaced entirely by background filtering, is controversial in the scientific community. The group of researchers that does not segment the ground, but rather starts filtering the background directly [106][134], is represented by the statement of Zhang et al. [133], who states ground segmentation might be unnecessary. This view is countered by many scientists who regard segmenting the ground a fundamental processing step [59][98][83][62][51][92].

Ground Segmentation separates all points of a point cloud into ground and non-ground points [44]. Retaining non-ground points only, makes downstream processing steps like clustering more efficient [62][51][101][83]. Discarding ground points can also contribute to significant data compression provided that the sensed environment allows modeling and replacing of the ground and walls through plane models. Li et al. [53] state up to 90% data compression rate in such environments because the ground, potentially accounting for 90% of the points in a point cloud, can be replaced by a plane model requiring merely four scalar parameters.

Employing ground Segmentation imposes challenges. First, the scanned environment may be uneven terrain, where a single plane model would approximate the whole road segment poorly [83]. Second, points in a point cloud are not uniformly distributed and the ground points farther away from the LiDAR sensor are sparser than ground points nearby, which complicates the detection of ground just like the detection of road users [83]. And third, many LiDAR sensors capture more than 100,000 points in one frame resulting in considerable computational costs when inspecting all of them at a scan rate of 10 Hz [83][44]. This strict time requirement imposes a trade-off between efficiency and accuracy [98] and may lead to segmentation errors. Errors in ground segmentation are distinguished into (1) commission errors (ground points are missed) and (2) omission errors (object points are removed) [123][98].

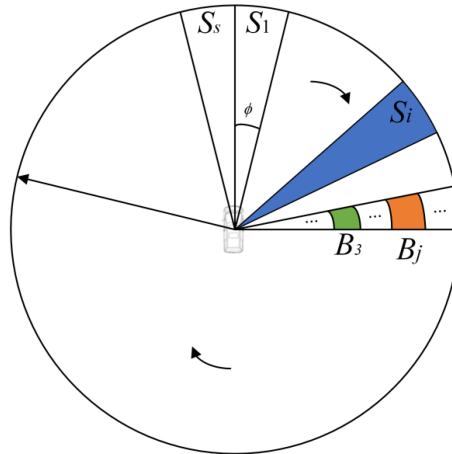
To describe ground segmentation algorithms this study adopts largely the division from Shen et al. [83], although there exist different ones [90][53][101][98].

### **Algorithms Based on Elevation**

Researchers refer to this category of algorithms when an elevation map is applied [83][62]. An elevation map is a mathematical transformation to project the 3D point cloud onto a 2D grid in the xy-plane (consider Figure 3.1 as an example), where each grid cell is described by computed features reflecting information about the elevation based on the points inside the grid cell [62]. These features are manually selected and can be any subset of the mean height, median height, height variance, maximum slope, and maximum relative height [62][128]. These features are compared to predefined thresholds to identify grid cells being part of the ground [62][101]. Grid cells with feature values below these thresholds are tagged as ground since the ground is rather flat and major variations are to be excluded. Major elevation variations, leading to feature values above the thresholds, occur when non-ground objects are in the grid cell, objects such as road users or static infrastructure [123]. These distinctive thresholds are defined by exploiting prior knowledge about the terrain in the

scanned environment [123]. Since the height as third dimension is not completely eliminated but rather kept as a feature in the described 2D grid representation, some researchers use the term 2.5D grid.

The elevation map is one of the most common techniques to segment the ground and was already outlined in a study from 2003 [123]. The technique was also successfully deployed in 2005 by the winning team of the DARPA Grand Challenge, an autonomous driving competition [83].



**Figure 3.1:** Polar 2D grid from a bird's eye view [43].

In 2010, Himmelsbach et al. [38] projected the 3D point cloud onto the  $xy$ -plane and partitioned the point cloud in polar 2D grid cells, as in Figure 3.1. From each non-empty grid cell the point with the minimum  $z$ -coordinate is retained while the other points are removed. The authors then apply a line extraction algorithm originally developed for 2D depth images to distinguish ground and non-ground grid cells, respectively. Remaining ground points in non-ground grid cells, as encountered beneath objects, are eliminated in a subsequent automatic revision.

In 2015, Börzs et al. [6] used an elevation map to segment the ground as first step of their object detection pipeline. After projecting the point clouds onto a 2D grid, the authors first discarded grid cells that contain less points than a specified threshold. Among those with enough points, Börzs et al. identified grid cells belonging to the ground if the respective difference between the maximum and minimum height of points remain under another predefined threshold.

In 2018, Kampker et al. [44] complemented the ground segmentation algorithm from Himmelsbach et al. [38]. After obtaining their segmentation result, Kampker et al. examined the determined height between adjacent grid cells, smoothed out prominent height variations, and filled empty grid cells by deploying a median filter.

The 2020 algorithm by Wang et al. [101] replicates the projection and partitioning of the point cloud from Himmelsbach et al. [38]. Instead of line extraction, this algorithm then exploits the slope inside the grid cell and of adjacent grid cells as a feature to separate cells with ground points only and cells with non-ground points.

Also in 2020, Leng et al. [51] suggested to divide a point cloud into vertical lines which are LiDAR slices consisting of all LiDAR channels at a single horizontal angle. The authors use them to analyze the elevation profile over the distance from the sensor for each horizontal angle. In particular, the height difference and the angle of slope between adjacent laser beams are considered to find segments peculiar to ground. Shen et al. [83] in 2021 created a two step coarse-to-fine ground segmentation algorithm. In the first step, the authors project

and partition the point cloud as showcased by Himmelsbach et al. [38]. In each grid cell they search the point with the minimum z-coordinate and set the z-coordinate as reference value. Points in the same grid cell higher but within a predefined tolerance range are tagged as ground points, while points outside of this tolerance range are tagged as non-ground points. In addition to the height of points, Shen et al. base their coarse segmentation on a computed slope in each grid cell. In the second step, the segmentation result is refined by computing confidence values in the transition area between ground and non-ground points and performing convolutions.

In 2022, Jin et al. [43] also adopted the partitioning of the point cloud from Himmelsbach et al. [38]. In each polar grid cell, their ground segmentation algorithm selects candidate ground points based on the height differences and the slope of adjacent points to obtain a ground reference height. Since not all grid cells are attributed a ground reference height and these references can be misjudged, a curve fitting algorithm based on Gaussian process regression refines the preliminary estimation. By comparing with RANSAC (see the rest of the section) and the original algorithm from Himmelsbach et al. [38], Jin et al. measure a higher processing time of 218 ms than RANSAC with 36 ms but lower than the algorithm from Himmelsbach et al. with 348 ms. On the other hand, their development shows fewer ground segmentation errors than RANSAC.

Also based on examining the elevation between neighboring grid cells, are algorithms functioning with morphology [98]. Morphology refers to a grid discretization which is gradually increased along with adapting the chosen feature thresholds to identify ground points on a larger scope [98]. An example is the simple morphological filter deployed by Srinivasan et al. [88] that upon termination outputs a binary mask to distinguish between ground and non-ground points.

### Algorithms Based on the Relationship between Adjacent Points

A 3D LiDAR sensors has multiple laser channels stacked above each other, where each channel scans the surrounding area with a different vertical angle [40]. Many 3D LiDAR sensors have a 360° field of view as a result of rotating around their vertical axis. By rotating each channel creates a circular scan that is shaped like a ring with the sensor in the center [62]. Each of these concentric scan rings has a different radius due to the different vertical angle of its corresponding laser channel [62]. This is a useful LiDAR feature for ground segmentation provided each point stores the laser channel it belongs to in its additional attributes [62].

In 2018, Narksri et al. [62] considered these concentric scan rings when examining the actual distances between neighboring rings and comparing it with the expected distances. They computed an expected inter-ring distance via the vertical angles of the two laser channels in question and the height of the LiDAR sensor above the ground applying simple trigonometry. The actual inter-ring distance decreases as the slope of the ground increases. If the actual inter-ring distance is much smaller than expected, the authors anticipate that it is due to the existence of a non-ground object. That is because in the case of a wall or a pedestrian standing approximately perpendicular to the ground, the difference between two radii approaches zero as the two adjacent laser channels have virtually the same distance to the sensor. This integral part of the algorithm by Narksri et al. [62] is also robust to slopes if the threshold for the expected distance is not set too tightly. After the first segmentation step, the remaining point cloud is divided into four quadrants, into each of which a ground is modeled to identify remaining ground points (described in the following paragraph).

Apart from the ring feature and the information by which laser channel a point was scanned, there are other features algorithms of this category employ to segment the ground. Other features involve the spatial relation of the points to each other, for instance their distance [90]. Combining points that are close to each other or otherwise related allows us to

construct isolated regions inside a point cloud, a procedure suggesting the alternative name region-based algorithms [53]. These isolated regions, in turn, allow us to compute surface normals, another feature examinable to identify groups of points as ground [90].

### Algorithms Based on Ground Modeling

These algorithms assume the ground is planar (“the [locally] flat world assumption” [101, p. 1]) allowing to approximate it with a plane model [62][90][98]. The plane model is computed via Random Sample Consensus (RANSAC) [24], an algorithm that can estimate cross-domain model parameters [62]. To fit an optimal plane model, the algorithm randomly selects three points of the point cloud (three points are the minimal subset required to uniquely define a plane in 3D space [53]) and determines a first estimate based on these three points and using straightforward algebra. The algorithm uses the first plane estimate coupled with a distance measure to find the number of points in the point cloud that lie in the plane or in the immediate proximity. The number of points nearby is the quality measure of the plane estimate and is saved together with the computed plane parameters. These figures define the current best plane model. Now, the algorithm starts again to randomly select three points to determine a second plane estimate as well as the number of points close to it. If the number of points near the second plane estimate exceeds the number of points near the first plane, the number and the corresponding plane parameters of the current best plane model are updated. RANSAC iteratively repeats this process until it has completed a predefined number of iterations. Once this termination criterion is reached, RANSAC outputs the current best plane model, which likely is the optimal one depending on the number of iterations RANSAC is granted. A reasonable number of iterations  $k$  needed can be computed by the Formula 3.1, where  $p$  is the desired probability to obtain the optimal plane model (e.g.  $p = 0.99$  for 99% probability),  $w$  is the number of points in the point cloud, and  $n = 3$  in accordance with the size of the randomly selected subset.

$$k = \frac{\log(1-p)}{\log(1-w^n)} \quad (3.1)$$

The plane model RANSAC outputs is likely to be optimal means it approximates the ground in the point cloud reasonably well. We can assume that RANSAC converges to this solution, because in this way the highest number of points can be approximated by a two-dimensional plane model, which is the objective of the algorithm. This reasoning is valid under the condition that the ground is planar, which is precisely the assumption from the beginning. After finding a plane model for the ground, we can identify ground points by employing a distance measure: all points of the point cloud close to the plane model are ground points and the rest non-ground points.

Given that RANSAC was proposed in 1981, many researchers have advanced the classic algorithm and assigned their development a new name. Li et al. [53] in 2017 proposed a RANSAC variant based on normal distribution transformation cells, which are a discrete grid representation. Their algorithm considers entire grid cells to fit a plane and to assess the plane estimate as opposed to classic RANSAC, which assesses the plane fit with regards to each individual point.

In 2018, Sun et al. [92] deployed a RANSAC variant named least median of squares regression (LMedS) [85], which does not require any parameter specification. In contrast, classic RANSAC requires us to specify the size  $n$  of the subset for constructing the plane estimates, the number of iterations until termination  $k$ , and the distance measure for assessing the plane estimates. Sun et al. complemented their ground segmentation using LMedS with an algorithm to remove residual ground points. Their complementary algorithm finds residual ground points by wrapping a 0.2 m thick cuboid around the plane.

In 2019, Sun and Mordohai [90] presented a new RANSAC variant to find the plane containing the most points: oriented point sampling (OPS). Their algorithm samples only one point in an iteration while RANSAC samples three to fit a plane. The algorithm from Sun and Mordohai requires only one sample point since they beforehand estimate normals for each point in the point cloud and a perpendicular normal is sufficient to uniquely define a plane.

The improved RANSAC from Wang et al. [98] from 2022 expedites two computationally expensive steps in classic RANSAC: (1) sampling a subset of the point cloud and (2) determining the number of points close to the plane estimate. Regarding sampling, they endorse the assessment from a previous study that yielded GroupSAC [64] according to which exploiting prior knowledge about the data is superior to naive random sampling as in classic RANSAC. Regarding the second step, Wang et al. adopt the idea from Preemptive RANSAC [66], another algorithmic advancement, which first determines several plane estimates before assessing them using merely a subset of the data.

There are more RANSAC variants from which some are listed, yet not analyzed in this study: MLESAC [96] from 2000, NAPSAC [61] from 2002, Lo-RANSAC [18] from 2003, PROSAC [17] from 2005, and RRANSAC [65] from 2013. Researchers compare several RANSAC variants regarding accuracy, runtime, and robustness in an extensive assessment [16] and conclude that no algorithm wins all metrics. All these variants emerged as classic RANSAC has flaws concerning efficiency and accuracy [98][53]. Moreover, RANSAC is not guaranteed to find optimal model parameters [92]. Apart from some flaws, the initial assumption about the ground being planar is naive and not met in all environments [98][133][40]. In fact, roads are intentionally sloped and not built flat so that rain does not collect in the middle of the road, but drains off to the side of the road to prevent aquaplaning.

More algorithms exist that segment the ground to discard it from the rest of the point cloud, also algorithms based on deep learning [97]. However, this approach suffers from the deficiency elaborated in the previous section and is therefore disregarded. Other challenges are also being faced by the presented unsupervised approaches. Ubiquitous challenges arise, for example, from uneven terrain, which contradicts the assumption of a planar ground. Further challenges include the few points in far distance to the sensor for identifying the ground, as well as the significant computational cost of this first step in the unsupervised processing chain to detect objects caused by the size of the raw point cloud.

### 3.3 Background Filtering

The serious computational cost when processing a raw point cloud affects background filtering if no ground segmentation is employed. Background points include not only the ground, but also static objects like walls or lampposts, and dynamic background such as swinging trees [127]. Background filtering, also called background subtraction, tries to eliminate these objects in the point cloud while retaining any road user present in the scene. Road users are the foreground that we seek to extract. Upon success of extracting the foreground, this processing step facilitates object detection regarding detection accuracy and computational efficiency [137][126][106][56][132]. Success refers to how well the algorithm upholds foreground points while at the same time excluding the maximum number of background points [99][137][56][106].

Considering walls as part of the background, one might think that plane fitting similar to ground modeling would be an option for background filtering as well. While plane fitting might work for house facades, it would struggle eliminating non-planar background such as

poles, plants, or benches [94]. Other challenges for background filtering algorithms are sensor noise, sensor shaking because of wind or vibration, temporary background like a parking car, and dynamic background like swinging trees [99][126][86].

Background filtering, involving these challenges, was addressed by previous investigations that we can divide into rule-based algorithms, volumetric-based algorithms, and point-based algorithms according to Song et al. [86].

### Rule-Based Algorithms

Rule-based algorithms become the prevalent technique to filter the background of a point cloud [86]. In this category of algorithms, researchers search or construct an empty LiDAR frame, meaning without any road user, and define its points as background points [86]. Assuming we possess such an empty LiDAR frame, we can use it to subtract the background from target LiDAR frames containing road users, such that we obtain these road users only. Subtracted are those points of a target LiDAR frame that lie in close distance to the points of the empty LiDAR frame, a distance being predefined [86]. This approach premises to have an empty LiDAR frame available, which can be difficult to obtain in congested traffic [86].

If it is not possible to record an empty frame, there is an alternative solution: constructing one through a set of non-empty frames recorded at different times [134]. After aggregating the set of non-empty LiDAR frames, researchers inspect the farthest distance reached by any combination of vertical and horizontal angles of the LiDAR sensor because they anticipate that this distance represents the background [134][126]. The number of combinations of vertical and horizontal angles in theory is finite since the vertically stacked laser channels scan at fixed vertical angles and each LiDAR sensor has a finite horizontal angular resolution. Real LiDAR scans, however, reveal that a laser beam radiates at slightly fluctuating angles, which is why researchers need to group and discretize the angles to obtain a manageable number of combinations. By taking the farthest distance at each vertical and horizontal angle and concatenating these three figures, researchers find the spherical coordinates of an empty LiDAR frame, convertible into Cartesian coordinates with Equation 2.1. This procedure works if each angle combination perceived the actual distance to the background at least once. If a direction is blocked, for example by a waiting vehicle, across the entire frame set, the procedure does not perceive the background behind and mistakenly sets the waiting vehicle as the background. Another premise to allow to inspect all angle combinations with regard to the distance is that the points of the non-empty frame set are available in spherical coordinates or are convertible into them.

Computer scientists use rule-based algorithms for images as well, where they tend to work more accurately due to the structured nature of images [134][132][56]. In contrast to images, point clouds are irregular meaning the position of points slightly changes between LiDAR frames [134][132][56]. This issue worsens when the LiDAR sensor is shaking because of wind or vibration [132][56]. Since few points coincide exactly, scientists cannot simply overlap LiDAR frames as it would be possible with images [134][132][56]. Despite the disadvantage for LiDAR data, scientists applied rule-based algorithms in several studies.

In 2016, Tarko et al. [94] assumed that static background objects over time maintain the same distance and direction from the LiDAR sensor. Under this assumption, the author aggregated LiDAR frames with low traffic and computed the mean distance and standard deviation across all frames at each vertical and horizontal angle. Using this constructed background frame, Tarko et al. defined in target frames that only points whose range is less than the mean plus three standard deviations at the correspondent angle combination are to be retained as foreground points.

In 2019, Zhao et al. [138] understood the finite number of vertical and horizontal angles as 2D coordinates in a conceptual image, an image in which the number of laser channels

specifies the number of columns  $m$  of the image and the horizontal resolution the number of rows  $n = \frac{360^\circ}{\alpha}$  (sensor with  $360^\circ$  field of view and horizontal angular resolution  $\alpha$ ). The image size then equals  $m \times n$ . See Figure 4.5 for an image section with six pixels resulting from three adjacent laser channels at two different horizontally adjacent angles. The pixels of the image do not represent distance values at each angle combination, but height values. At each angle combination, they stored the height of the background object because the idea of their azimuth-height background filtering algorithm is that foreground objects cause a detectable change in height. Lower laser channels pointing to the road notice when a higher object suddenly intersects the laser beams that normally reach the road. If the change in height exceeds a predefined threshold, concerning points are not filtered. The algorithm filters 98 % of background points while retaining up to 100 % foreground points in the authors' self-generated roadside LiDAR data. The authors measure 300 ms to process one LiDAR frame.

In 2020, Zhang et al. [134] constructed a similar conceptual image than Zhao et al. [138] but its pixels quantify the distance at each angle combination. Regard Figure 4.6 that visualizes two adjacent horizontal angles leading to two distinct distance values as the tree is closer to the LiDAR sensor than the wall. At each angle combination, the authors stored the maximum distance as well as the mean distance following Tarko et al. [94]. Using the notion that the background is farther away than passing road users when observing a particular direction over time (as in Figure 4.7), Zhang et al. extracted points belonging to road users based on the accumulated knowledge in their conceptual image. Their image was stored in a hash map to access data quickly.

In 2021, Zheng et al. [139] combined mean background modeling (MBM) with a background difference method (BDM). First, they defined a 2.5D grid map in the  $xy$ -plane around the LiDAR sensor, where each grid cell has a side length of 1.6 m. Each grid cell stores 16 values about the local height, where the 16 values are only influenced by their respective laser channel from the deployed LiDAR sensor. Because of the large side length, each rotating laser channel produces multiple points inside a grid cell, which is why the 16 values are average height values. The authors reasoned the height information varies significantly between the constructed empty background model and a target frame when a road user passes through an observed grid cell and is scanned by at least one laser beam. If the height deviation or difference caused by the road user is higher than a predefined threshold, the associated points are retained as foreground. To quickly check all points of a target frame, Zheng et al. store the grid map in a hash table and use as key the grid position and the laser channel associated with every point.

In 2022, Zhang and Jin [126] adopted the data representation of Zhang et al. [134] to transform an unstructured point cloud into a structured image. While previous research stored in each pixel the maximum distance, the mean distance, or some manual set threshold as a reference value to distinguish between background and foreground, Zhang and Jin developed a non-parametric coarse-fine triangle algorithm to determine a more suitable reference distance. Another distinct component of their algorithm is to extract a region of interest using the  $x$ - and  $y$ -coordinates of all points in a point cloud and a binary mask determined through geofencing, a procedure to associate GPS coordinates with points displayed in a virtual space. In addition to background filtering based on the distance, they proposed an algorithm to distinguish between background and foreground based on the intensity attribute of each LiDAR point. Apart from the intensity thresholds users have to specify when deploying this algorithm [126], scientists argue the intensity attribute were not distinct and unreliable [106][13].

In 2022, Liu et al. [55] constructed a static background image similar to Zhang et al. [134] and stored in each pixel either the maximum distance reached at this angle combination or zero. The authors choose zero if more than 50 % of their considered LiDAR frames



for constructing the image denote a point distance of zero, occurring when the laser aims at objects beyond the sensor range. The algorithm of Liu et al. assigned points of target frames to the foreground if their corresponding pixel contains a figure greater than zero and more distant from these points than a predefined threshold. Points of target frames are only examined if they lie in a region of interest, extracted similar to Zhang and Jin [126]. To refine the static background, the authors also deployed algorithms for clustering and outlier removal.

### **Volumetric-Based Algorithms**

To filter the background, volumetric-based algorithms aggregate a set of LiDAR data over time just like rule-based algorithms. After merging the aggregated LiDAR data into a single point cloud, volumetric-based algorithms partition this point cloud into a volumetric grid, such that every point resides inside a single voxel [86]. When using roadside LiDAR, static background points maintain a fixed position throughout all recorded frames, while dynamic foreground points vary. This consideration justifies that computer scientists assign all voxels to either the background or the foreground based on the point density inside [86]. Voxels with a high point density represent background, while those with a low point density represent foreground. Once assessed, the merged point cloud exemplifies future target point clouds as the volumetric grid is equally valid in all point clouds from the same sensor. Vividly speaking, one point of a real object is always assigned to the same voxel (not considering rare boundary points), no matter in which frame, so researcher can confidently discard those points in a target point cloud that lie in background voxels. To assign voxels to the background and the foreground, respectively, researchers define a threshold. Another parameter they have to specify is the side length of a voxel. If the side length is specified small, the computational load is higher since the algorithm has to examine more voxels [106]. If the side length is specified too large, foreground points are extracted inaccurately [106].

In 2018, Wu et al. [106] (based on a previous publication [109]) suggested a voxel side length of 0.1 m for their three-dimensional density-spatial filtering (3D-DSF). 3D-DSF uses a dynamic threshold learned by the algorithm that takes the decreasing point density over distance into account. To filter the background, the authors only consider points with a range below 60 m. In this study, they also assessed the number of frames required to assign the voxels one-time, a number they put at 1500 to 3000.

Zhang et al. [125] proposed a 3D occupancy grid in their study from 2019. They divided the environment into occupied, free, and unknown voxels to narrow the regions to be analyzed. The authors recognize foreground points by monitoring inconsistencies in voxels over time.

Following related research, Zhang et al. [133] in 2019 began to filter the background by extracting a region of interest solely covering the street. Inside the region of interest of an empty LiDAR frame without road users, the authors established a volumetric grid, in which each voxel saves (1) the maximum height among all points, (2) the minimum height, and (3) a computed feature, which they call the maximum relative height. By comparing points of target frames with these three values, the authors extracted foreground points in the scene.

Also in 2019, Lv. et al. [58] presented raster-based background filtering. They adopted recommendations such as a voxel side length of 0.1 m from their reserarch colleagues Wu et al. [106]. Their algorithm first extracts a cylindrical region of interest whose height Lv. et al. determine through a defined detection range as well as the maximum vertical angle of the LiDAR sensor. Their algorithm sets voxels as background or foreground by monitoring the number of points within the voxel across a set of consecutive LiDAR frames since the authors assume this number alternates significantly due to road users moving into and out of a voxel-defining region. They identify voxels where the number does not alternate as background voxels. Using a set of 60 LiDAR frames generated at three sensor locations, Lv.

et al. determine a higher precision of the algorithm compared to their colleagues' 3D-DSF: 99.4% of background points are excluded compared to 98.1%; and 0% of foreground is removed compared to 3.6%. Both algorithms need 100 ms to filter one LiDAR frame.

In 2020, Wu et al. [107] enhanced their algorithm 3D-DSF [106] for greater robustness under adverse weather in an development called ground surface-enhanced density statistic filtering (GS-DSF). Aside from picking frames randomly to construct the background model, the authors examine whether points appear in all regarded LiDAR frames or not. They find the same point in other frames by searching within a predefined distance. If a point appears in all LiDAR frames, it is assigned to background. If a point is missing in some frames, it is further analyzed and eventually assigned to either background or foreground. After constructing the background model this way, GS-DSF can filter target LiDAR frames. Wu et al. verify that GS-DSF (0.6%) results in significantly less miss-assigned foreground points in congested traffic and windy weather compared to their previous 3D-DSF (50.6%).

Also in 2020, Zhang et al. [131] introduced subspace-frames, which are fixed cubical subspaces with a side length of 5 m defined over a set of LiDAR frames. Inside such a spatial-temporal subspace, the algorithm from Zhang et al. applies unsupervised clustering with a t-distribution mixture model to assign the subspace to one of four groups: (1) many road user points, (2) few road user points, (3) no road user points, (4) occluded background points. Afterwards, each subspace is divided into voxels with a side length of 0.1 m, where each voxel is identified for accumulating background or foreground points. Compared to 3D-DSF [106], point association [132], and raster-based background filtering [58], the authors count significantly less miss-assigned points with their algorithm, especially in congested traffic: 25 miss-assigned foreground points per frame versus 63 (3D-DSF), 42 (point association), and 35 (raster-based background filtering); and 173 miss-assigned background points per frame versus 274 (3D-DSF), 199 (point association), and 209 (raster-based background filtering).

In 2021, Wang et al. [99] aggregated LiDAR frames from different points in time and modeled for each voxel the number of points and the average height as Gaussian distributions, rather than computing absolute figures. By applying a Gaussian distribution, defined through a mean value and a standard deviation, the authors address LiDAR fluctuation. The modeled distribution serves to divide voxels into background and foreground voxels, similar to previous volumetric-based algorithms. The outcome of this division is stored in a table to distinguish points from target frames based on their position.

### Point-Based Algorithms

Point-based algorithms process the raw point cloud without partitioning it into voxels. Instead, an early study, for example, [113] assumed that background points have more neighboring points when regarding a set of LiDAR frames. Employing an algorithm to count the neighboring points of every single point, the authors identified points with few neighbors over time and set those as foreground.

In 2018, Zhang et al. [132] proposed point association, where they first aggregated a set of empty LiDAR frames without road users and merged them into a single point cloud. Merging this point cloud with a target point cloud and projecting it into the xy-plane reveals points from the target point cloud not coinciding with any point from the empty reference frames. To spot these points, the authors must record from which point cloud a point originates before they merge the target point cloud. Points not coinciding protrude visually when the two projected point clouds are colored differently. Based on their position, Zhang et al. associate points between both point clouds. Points exceeding a predefined distance threshold are identified as foreground points. Remaining background points are filtered by establishing a region of interest.

In 2020, Song et al. [86] presented an algorithm that analyzes frames channel-by-channel

because they observed a distinctive pattern in the distances of related points across the 360° horizontal angle when road users are present. Points are related if they were scanned by the same laser channel at the same horizontal angle but belong to different LiDAR frames. When comparing a target frame with an empty frame, road users cause a significant difference in distance across multiple adjacent related point pairs. Corresponding points in the target frame belong to the foreground if (1) there are at least a predefined number of adjacent related point pairs (2) where the point pairs between the two frames are each farther apart than a sensor-dependent threshold. Pedestrians provoke few of those adjacent points because they are slim, so Song et al.’s algorithm consults multiple neighboring laser channels to recognize point pairs apart with certainty.

In 2021, Liu et al. [56] aggregated 1500 empty LiDAR frames without road users to synthesize a k-d tree, which is a data structure to efficiently trace neighboring points. By querying the k-d tree combined with a distance measure, they identified points exceeding a predefined threshold of neighbors, points that are assigned to the background. Missed points due to dynamic background are filtered by a region of interest [56].

In 2022, Wu et al. [104] proposed a variable dimension filter to filter background points without aggregating a huge LiDAR frame set. Starting with a random point  $A$  from a random frame  $i \notin NF$ , the authors inspect whether point  $A$  exists in other frames  $j \in NF$ . Because of laser beam fluctuations, they define a spatial search ellipsoid by computing a search distance in horizontal and in vertical direction based on specifications of their LiDAR sensor. If their algorithm finds point  $A$  in other frames  $j \in NF$ , it increments the number of temporal neighbors  $NN \in [0; NF]$  by one. The acquired information is stored by replacing point  $A$  in frame  $i$  with a vector  $(min_x, min_y, min_z, max_x, max_y, max_z, NN) \in \mathbb{R}^7$ , where  $min_k$ ,  $k \in [x, y, z]$  and  $max_k$ ,  $k \in [x, y, z]$  represent the minimum and the maximum coordinates of point  $A$  and its found temporal neighbors. Repeating the procedure for the remaining points in frame  $i$  creates a background matrix little by little.  $NN = 0$  means point  $A$  did not reappear in other inspected frames, indicating a foreground point. In contrast,  $NN = NF$  means point  $A$  appeared in other inspected frames as well, indicating a static background point. Wu et al. attribute  $0 < NN < NF$  cases to occlusion or congested traffic and they then employ a sophisticated algorithm to decide based on the point distribution. Their development demonstrates superior results than 3D-DSF [106] and raster-based background filtering [58] when considering their data and the designed error rate.

A background filtering algorithm not fitting in any of the three categories is slice-based projection filtering from 2020 by Lin et al. [54] because the algorithm involves supervised learning with an artificial neural network. The authors first project the raw point cloud into the two-dimensional xy-plane and transform the points into polar coordinates to reduce the computational load. The points are then divided into four categories: (1) valuable object points, (2) worthless object points, (3) abnormal ground points, and (4) normal ground points. To extract the first and the third category as points of interest, Lin et al. employ vertical slices, which facilitate the analysis of points. The artificial neural network enables their algorithm to adapt to different road inclinations and laser beam angles when the sensor is moved to a new location. Lin et al. observe that slice-based projection filtering retains more points of interest while filtering more background points than 3D-DSF [106]. Additionally, they measured a lower runtime of 51 ms compared to 113 ms for 3D-DSF.

The background that the LiDAR sensor perceives can change, considering parking vehicles or people sitting on a bench. To adapt to a changing background, algorithms for background filtering should regularly update the reference data used to distinguish between background and foreground points. Aside from a changing background, these algorithms are challenged by variable weather as researchers discovered serious performance drops during rain and snow [15]. These weather conditions are usually disregarded when developing an algorithm

to filter the background [15].

### 3.4 Outlier Removal

After extracting the foreground of the point cloud, the algorithmic pipeline addresses outliers, which are points markedly distant from other points [4] and due to several reasons. First, some outliers are measurement noise occurring when a laser beam points at objects beyond the sensor's measurement range [4]. A second cause for outlier points is a shaking LiDAR sensor due to wind or vibrations [4]. Outliers or noisy data can lastly be due to measurement inaccuracies of the sensor.

Removing outliers is a fundamental problem when processing large-scale 3D data and if successful ensures a more accurate result obtained in less time [4]. To accomplish this task, there are algorithms divisible into (1) traditional approaches, (2) wavelet-based approaches, and (3) approaches based on artificial intelligence (AI) [4].

#### Traditional Algorithms

Traditional algorithms for removing outliers use clustering or analyze data properties such as the data distribution, the density, the depth, or the distance [4]. These algorithms are challenged by big data and real-time requirements [4].

A typical algorithm available in software frameworks to process 3D data is radius outlier removal (ROR) [15]. ROR counts the number of adjacent points within a specifiable search radius around a point [15]. If inside this imaginable sphere there are fewer points than the number the user has specified, the point is off the rest of the data and is identified as an outlier. While being intuitive, ROR does not consider the decreasing point density of LiDAR data and tends to remove too many measurements points in the far distance [15].

Another prevalent algorithm in software frameworks which is supposed to be more accurate in the distance is statistical outlier removal (SOR) [15]. SOR computes the mean distances of all points to its  $k$ -nearest neighbors and then models these mean distances as a global Gaussian distribution in the point cloud [92]. If the determined mean distance of one point to its neighbors is outside the predefined standard deviation of this global distribution, the point is removed [92]. The presumed higher accuracy of SOR is offset by substantial computational costs for large-scale 3D data [15][4].

In 2018, Balta et al. [4] tried to reduce the computational costs of SOR while upholding the high accuracy of outlier removal. The authors call their presented algorithm fast cluster statistical outlier removal (FCSOR), which reduces the computational complexity even in big data through clustering and dimensional reduction of the 3D space. Balta et al. observe superiority over the classical SOR in experiments.

Zhang and Jin [126] deploy local outlier factor (LOF) to address outliers in their object detection pipeline. LOF considers a point as an outlier if the summed distance to its  $k$ -nearest neighbors is smaller than a predefined threshold.

#### Wavelet-based Algorithms

To identify outliers, wavelet-based algorithms transform the data into another space [4]. In the transformed space, outliers lie in regions with low density, which are then completely cleared of the few outlier points [4].

### AI-based Algorithms

In 2019, Hermosilla et al. [37] presented total denoising, which cleans point clouds from outliers by employing unsupervised learning. Their algorithm aims to model a noise distribution and demonstrates performance similar to supervised learning approaches in outlier removal while not requiring clean ground truth samples.

Even though many studies utilize algorithms to remove outliers in their data, some studies do not mention it explicitly. Wu et al. [106] even state that any noisy points remaining after they filtered the background are eliminated automatically by advanced clustering algorithms such as density-based spatial clustering of applications with noise (DBSCAN).

## 3.5 Point Generation

After removing points in the previous step, point generation paradoxically intends to add new points to a point cloud. Removing outliers and adding new points in locations of road users, both enhance the accuracy of object detection [133][84]. What is summarized in this section under the term point generation, are two related computer vision tasks: (1) point cloud upsampling and (2) point cloud completion. Given a sparse point cloud, point cloud upsampling aims to make the visible objects contours denser [112]. Parts of objects not scanned by the LiDAR sensor are not reconstructed. Reconstructing these missing parts is the goal of point cloud completion, which predicts how complete objects appear based on the observed contours [112]. Both computer vision tasks necessitate unsupervised learning using a deep learning model to comprehend the underlying geometry of 3D shapes [112]. Point cloud upsampling and point cloud completion also belong to unsupervised representation learning, a research field more active in natural language processing and computer vision with images [112].

### Semantic Point Generation

In 2021, Xu et al. [115] observed a performance drop in supervised deep learning models when they are confronted with point cloud data from another domain. To address the domain shift, they proposed semantic point generation (SPG), a point cloud completion technique capable of improving the detection accuracy of supervised LiDAR detectors. Since SPG is a general approach, it can improve the detection accuracy not only in a new target domain, but also in the source domain, meaning independently of any domain change. While SPG was designed for supervised LiDAR detectors, the method itself is based on unsupervised learning. SPG's architecture is an encoder-decoder network about which the interested reader can inform him- or herself in excellent online resources [5]. The encoder-decoder network learns the semantic information of a point cloud, that is, it identifies voxels belonging to the foreground and generates points only inside these foreground voxels to complete the point cloud. When applying traditional algorithms, as targeted in this study, SPG would struggle less identifying foreground voxels since the previous steps already excluded non-foreground points. The generated semantic points combined with the original point cloud yield an augmented point cloud that is then input by a supervised deep learning model.

The authors state that on average 6% additional points are added to an unprocessed point cloud and that their network requires 0.39 million parameters. Regarding processing speed, Xu et al. measured 17 ms additional runtime to process one LiDAR frame.

## 3.6 Clustering

The extracted foreground, perhaps refined by outlier removal or point generation, represents an elusive data complex, in which individual road users can hardly be identified. For this reason, unsupervised object detection pipelines divide the remaining point cloud into individual groups of points [101][45] hoping that each group embodies a road user [132]. The grouping process is called clustering and associates points according to their similarities [137]. One cluster unites points that are distinct to the rest of the points and that share common properties [137].

Scientists cluster the remaining point cloud in 3D or in 2D depending on whether they include the height information [44]. The height as third dimension is negligible as road users never move over each other and thus can be distinguished unambiguously by their 2D coordinates [10]. Projecting all 3D points on the xy-plane before clustering reduces the computational load and expedites object detection.

A challenge for clustering are road users close to each other because the algorithm might combine them into one cluster [94]. Road users like vehicles and pedestrians are especially close at intersections when they are waiting for a green light [132][133]. To separate the objects correctly from one another, even in this critical situation, the parameters of the clustering algorithm must be set small [133]. Difficulties in clustering arise from outliers located between the road users because the algorithm misunderstands them as a connection point between two clusters [94].

While scientists favor small parameters to ensure an accurate division, they have to be careful not to select them too small because this can lead to a subdivision of a road user into several clusters [132][133]. Road users prone to be divided in this process are vehicles with large windows since they barely reflect laser beams at this point which causes the actually dense object to be displayed discontinuously in the sensor data [132][133]. To minimize both processing errors, the parameters must be meticulously adjusted [132].

Similar to outlier removal, it is unreasonable to set constant parameters to cluster the entire point cloud without considering the decreasing point density in farther distance [101][137]. Another analogy is the substantial computational cost for large-scale data, so Wang et al. [101] suggested to split the foreground in fragments and to cluster objects inside each fragment. While this approach reduces the computational complexity, they need to reunite objects that were split apart during the fragmentation.

### **k-means**

k-means is a clustering algorithm where the user specifies the number of clusters  $k$ , each of which initialized with a random mean value [86]. The algorithm now performs two steps in alternation: (1) k-means assigns each data point to the nearest cluster mean and then (2) updates the mean values by averaging over all assigned points of one cluster. Repeating both steps in alternation leads eventually to convergence and k-means terminates when the update of the mean is smaller than some threshold.

The huge disadvantage of this algorithm for object detection is the user needs prior knowledge about the number of road users in the data as misstating this number leads to poor results [86]. How many road users are present is unknown [137][133][132], which is why other algorithms are preferred in unsupervised object detection.

### **DBSCAN**

Density-based spatial clustering of applications with noise (DBSCAN) [23] is a commonly used clustering algorithm for object detection [137] since researchers do not have to specify the number of clusters or road users beforehand [134]. DBSCAN exploits the density of data

to identify clusters of arbitrary shape [133]. To succeed the algorithm uses two parameters: (1) the minimal number of points required to form a cluster and (2) a parameter reflecting the maximum search radius [132][137][86][55]. The second one defines how far a data sample can lie apart from the rest while still being grouped into the same cluster [137]. By traversing all data samples, the clusters grow and incorporate points one by one [132][134]. Points outside of the maximum search radius that pooled do not reach the minimum number are identified as noise [86][134].

Many studies cluster the entire point cloud with DBSCAN using the same parameters despite the heterogeneous point density, which is anticipated to be challenging given that the algorithm is based on density [99][86]. For the same reason, DBSCAN benefits from point generation since increasing the density of road user points leads to a more accurate clustering result [133].

Apart from point generation, Zhao et al. [137] and Chen et al. [13] propose a modified DBSCAN clustering algorithm to address the heterogeneous point density. Instead of a constant minimal number of points and search radius, their DBSCAN variants adapt these parameters. Similarly, Zheng et al. [139] ascertains advantages of their hierarchical maximum density clustering of application with noise (HMDCAN) over DBSCAN, because HMDCAN adapts dynamically to the local density distribution. Zhang et al. [124] suggest Grid-DBSCAN to avoid calculating the Euclidean distance from one point to all other points, which means a high computational effort avertible with only a small sacrifice in recall. Wu et al. [110] proposed 3D-SDBSCAN to address the performance degradation of DBSCAN in snowy weather. Also, Wu et al. [109] developed multi rectified DBSCAN (MCDBSCAN) to identify lanes more accurately as DBSCAN is confused by remaining, non-filtered background points.

### **Euclidean Clustering**

Another clustering algorithm used in object detection is Euclidean clustering, which requires the user to specify three parameters to efficiently cluster a point cloud: (1) a distance tolerance to separate groups of points, (2) the minimum number of points, as well as (3) the maximum number of points in a valid cluster [101][122]. Researchers also developed variants of Euclidean clustering with adaptive parameters to cope with varying point densities [56].

### **Mean-shift Clustering**

Following k-means, this algorithm is based on continuously updating the mean of each point cluster [86]. Different from k-means, however, scientists do not have to specify the number of clusters as mean-shift clustering discovers the number itself [86]. The algorithm starts by randomly selecting a point and introducing a circular sliding window around this point with a predefined radius called the kernel [86]. The kernel is updated such that it finds regions with a higher point density [86]. By continuously updating, the algorithm converges to the center points of each cluster, a characteristic to be regarded as centroid-based algorithms [86]. The drawback of mean-shift clustering is the low computation speed due to the sliding window [86].

### **Gaussian Mixture Models**

A Gaussian mixture model (GMM) assumes that the point cloud clusters follow a Gaussian distribution, meaning we can describe them with their mean and standard deviation [86]. After randomly initializing these two quantities for each cluster, the expectation-maximization (EM) algorithm optimizes them [86]. To optimize, the EM algorithm considers the probabilities that a point belongs to some cluster [86]. The closer a point reaches the mean of a cluster, the more likely it is part of the cluster, so the EM algorithm tries to maximize this

probability [86]. This optimization problem is computational demanding in addition to the drawback of having to specify the number of clusters [86]. A remedy to this problem would be to generate different GMMs simultaneously and to choose the best model but this increases the computational load even further [86].

### Deep Learning

Xiao et al. [112] proposed to employ clustering algorithms to create a training dataset, where each identified cluster is labeled with a cluster ID. Using this dataset, a deep learning model can be trained to understand the context similarity and data distribution in point clouds. This self-supervised approach though depends on the performance of a clustering algorithm and the quality of the generated dataset.

k-means, DBSCAN, mean-shift clustering, and GMMs are currently the prevalent algorithms to cluster point clouds [86]. However, some studies in the field deploy other algorithms: Zhang et al. [120] developed an adaptive two-stage clustering algorithm, which combines Euclidean clustering with DBSCAN; Zhang et al. [121] and Luo et al. [57] use a radially bounded nearest neighbor graph (RBNN) [46]; Kim et al. [45] cluster the point cloud in a spherical coordinate system based on the angular distance; Hanten et al. [35] discretize the point cloud into voxels and cluster with a k-d tree; Wu et al. [107] also deploy voxelization-based clustering to cope with adverse weather; Zhang et al. propose an unsupervised clustering method specifically designed for roadside LiDAR data [130]; Cen et al. [12] apply depth clustering in proposal regions in their metric learning and unsupervised clustering (MLUC) network; Sun et al. [92] use an algorithm named basic clustering techniques [78].

## 3.7 Bounding Box Fitting

Clustering divides the foreground point cloud into point groups, each representing a road user. The location of a cluster is approximated by a reference, for example the geometric center of all cluster points, to allow object tracking and to obtain the trajectory of an object [137]. Objects like vehicles are not well approximated by a single reference point since a point has no orientation and insinuates the object could move in arbitrary directions but vehicles have non-holonomic constraints and for instance cannot drive sideways. Also, some vehicle parts pointing away from the laser sensor are not visible in the point cloud but merely the object contours facing the sensor, making the cluster contour a poor approximation of a vehicle body [137][82][72][45][116][135]. The issue worsens considering occlusion and the changing perspective on moving vehicles [45][135]. To grasp an object's dimension and orientation, in addition to its position, scientists fit a shape model to each cluster. These shape models display the entire object pose more expressively. The shape model for vehicles is a bounding box or an L-shape [72][6][132][116][135]. A 3D bounding box is often described by seven parameters: three coordinates for the box center, the box extent in three dimensions (width, length, height), and a heading (or yaw) angle [71][12]. In this description scientists neglect the two angles for roll and pitch movements. The L-shape refers to two characteristic perpendicular lines that portray vehicles in point clouds that are recorded from a diagonal perspective, for example from the front left. Aside from adopting vehicle shape models, scientists approximate pedestrians or cyclists by circles and ellipses, respectively [49]. Since most studies focus on vehicle detection though, this section outlines vehicle shape models.

One option to find two orthogonal lines that depict the front (or the rear) and one side of the vehicle is principal component analysis (PCA) [116][6]. PCA finds the dominant axis



inside a cluster, whose length and orientation in many cases coincides with one side edge of the vehicle. In this case, the orientation also matches the heading angle of the vehicle. By computing a orthogonal line and sliding it to the end of the dominant axis, PCA finds the front or the rear edge of the vehicle. Lastly, the algorithm adjusts the length of the axis ensuring the minimum length but still enclosing all points of the cluster.

To find a vehicle shape model, Börçs et al. [6] in 2015 projected cluster points onto the xy-plane and computed in this 2D representation the convex hull around all points of a cluster. The convex hull is a polygon from whose boundary an imaginary person can see the entire boundary in a straight line (without visual obstruction), implying the boundary must not be invaginated. Using the 2D convex hull, the authors determine a minimum rectangle enclosing the convex hull and all cluster points. By incorporating the maximum height information inside a cluster and fitting another hull for the side view, Börçs et al. fit a more accurate vehicle shape model than a 3D bounding box and with useful features for the subsequent classification step.

In 2016, Tarko et al. [94] applied a procedure similar to Börçs et al. to find a rectangle enclosing all points of a cluster. After computing the 2D convex hull, Tarko et al. executed a modified version of the minimum area bounding rectangle algorithm called minimum area error rectangle. The modified version assumes the rectangle must have a common edge with the convex hull, a conception to estimate the yaw angle more accurately.

In 2017, Zhang et al. [132] proposed L-shape fitting based on a search algorithm. The search algorithm iterates over all possible yaw angles each combined with a rectangle that encloses all cluster points. To find the best fit among all rectangles, the algorithm relies on the distances of all points to the rectangles' four edges to compute an error. This error is incorporated in three different criteria which are all suitable to select the best fit: (1) rectangle area minimization, (2) points-to-edges closeness maximization, and (3) points-to-edges squared error minimization. The performance of all three are evaluated in experiments.

In 2018, Naujoks et al. [63] presented a 3D bounding box fitting approach based on the 2D convex hull but complemented by a step to correct the orientation. Taking the 2D convex hull, the authors compute a minimum area bounding rectangle using Rotating Calipers [2]. Since the point cloud often does not contain the entire contour of a vehicle, the estimated orientation of the rectangle can be inaccurate. To correct the orientation, Naujoks et al. suggest a three point based heuristic approach or a Kalman filter. After correcting the orientation, their algorithm adjusts the dimensions of the rectangle, so that all points are enclosed again. Finally, they add the height information to the bounding box by considering the minimum and maximum height among all cluster points, respectively.

In 2018, Kim et al. [45] projected the points of a cluster on the xy-plane and exploited the resulting characteristic L-shape that vehicle clusters depict from a bird's eye view. In each L-shaped cluster, they determined the vertex using their iterative end-point fit method. Starting from this vertex, Kim et al. extracted the two lines of the L, which are the outer lines of the vehicle. The longer line defines the heading angle of the vehicle. Based on both extracted lines, the author enclose the cluster points in a 2D bounding box with minimum area.

In 2018, Yuan Sun et al. [92] addressed incomplete vehicle contours with a reconstruction method restoring the missing points. Based on the completed point cloud, the authors fit a minimum-volume bounding box by applying PCA. PCA outputs the principle eigenvectors, which they use to construct a new coordinate system. The vehicle cluster is translated into this new coordinated system, where the 3D bounding box is determined, before the cluster with corresponding 3D bounding box is translated back to the original coordinate system. The closer front corner and back corner point pair of the bounding box indicate the location of a vehicle.

In 2018, Qu et al. [72] observed that we can describe an L-shape with three key points: (1) the outer two vertices and (2) the corner vertex. They decomposed the L-shape fitting procedure into two steps. The first step determines the two outer vertices of an L-shaped vehicle cluster. The second step localizes the corner vertex. After finding these three key points, their algorithm determines the four edges of a fitted 2D bounding box.

In 2018, Kampker et al. [44] fitted 2D bounding boxes with the Minimum Area Rectangle (MAR) algorithm [26]. Afterwards, they corrected the result of MAR by deploying a feature-based L-shape fitting algorithm. To complete the 3D bounding box, they restore the height of each object cluster.

In 2018, Oniga and Nedevschi [69] assumed fitted 3D bounding boxes whose three angles are equal to zero, meaning aligned to the coordinate axes. For these axis-aligned bounding boxes, they try to find an optimal yaw angle from a bird's eye perspective using RANSAC. RANSAC finds the dominant line of an object corresponding to its yaw angle. The algorithm from Oniga and Nedevschi then determines the perpendicular second line and validates the result given the point cluster shape. Their implementation requires less than 0.2 ms to find the orientation of an object and the average deviation is at most  $0.4^\circ$  during the two conducted experiments.

In 2018, Yang et al. [116] adopted the search-based rectangle fitting algorithm from Zhang et al. [132] from the year before. To find the most suitable yaw angle among all possible ones, they suggest a contour aggregation (CTAG) criterion.

In 2019, Fu et al. [27] also adopted the search-based rectangle fitting algorithm from Zhang et al. [132]. To obtain more accurate rectangles around point clusters, the authors first refine the clusters with a T-linkage-based algorithm, which is a RANSAC variant to remove outliers. After the clusters are cleared of outliers, a cost function identifies the rectangle with the most suitable yaw angle. In this cost function, the authors involve multiple criteria in contrast to Zhang et al. [132] and Yang et al. [116], which use a single criterion.

In 2019, Zhang et al. [125] suggested rectangle fitting that considers multiple consecutive LiDAR frames to average the parameters of the rectangle. To recognize objects in consecutive frames, their algorithm needs to track the clusters.

In 2019, Zhao et al. [137] projected cluster points originating from vehicles onto the xy-plane and computed 2D bounding boxes by inspecting the minimum and maximum x and y coordinates of a cluster, respectively. Depending on the location of a vehicle cluster with respect to the LiDAR sensor, the authors select one or two of the closest bounding box corners to reference the vehicle.

In 2020, Kloeker et al. [47] fitted bounding boxes using fixed dimensions for every object class. The fixed dimensions are determined from detected objects in the scenario center, where the point density and thus the dimensions are most accurate. The heading angle of objects is smoothed by a triangular kernel window.

In 2021, Zhao et al. [135] proposed an L-shape fitting algorithm that first determines the closest edge of a vehicle cluster to the LiDAR sensor. The remaining edges are then fitted by deploying RANSAC. As deriving the heading angle from the longer line of the L-shape can be inaccurate for some perspectives on vehicles (e.g. when viewed from behind), the authors correct the heading angle while tracking the detected objects.

### 3.8 Classification

Classifying extracted point clusters is not necessarily a processing step after fitting a bounding box. In fact, researchers can bypass bounding boxes for object detection [133] or deploy bounding boxes only for vehicles [49], for which they have to identify a cluster as vehicle be-

forehand. However, knowing the dimensions of all object clusters can be used to differentiate between object classes. For instance, a pedestrian occupies much less space in the xy-plane and is smaller in height than a truck, observations exploitable to classify an object. Because of this consideration, classification is placed after bounding box fitting.

When scientists classify objects into cars, trucks, buses, pedestrians, and cyclists [136], they are faced with challenges, some of which similar to challenges in clustering. A group of people close to each other, for example, might be classified as a vehicle when not correctly separated [100]. Identifying small road users such as pedestrians is generally difficult since they exhibit a small surface area to intersect laser beams, resulting in few points in a LiDAR scan. Since state-of-the-art detectors struggle to detect small road users, scientists tend to attribute little attention to enhance the detection performance of them and focus primarily on detecting vehicles. Another challenge in object classification is to distinguish between road users having almost identical dimensions and shapes such as cyclists and motorcyclists. These challenges reveal how sophisticated classification methods must be; they involve expert knowledge or thorough feature engineering, depending on the approach.

### Heuristics

By applying expert knowledge, researchers can distinguish between multiple object classes in a decision tree [125]. Each branch of the decision tree uses one or multiple features with corresponding thresholds, features like the object's occupying area or its height. A heuristic approach is also referred to as rule-based [122].

### Machine Learning

Feature-based machine learning is a prevalent approach to classify objects [137]. The features scientists define can be bounding box dimensions, the total number of points in a cluster, or the positions of clusters. The quality of handcrafted features and the consequent classification accuracy depends on the point density of object clusters [137]. The more detailed and complete objects appear in the point cloud, the more accurately objects can be classified.

In 2009, Zhao et al. [136] established the object size and speed as features for all of their object classes. In addition to those general features, they defined another feature for each individual class. The additional feature for the car class are two lines, which Zhao et al. call axes. The two axes they observe correspond to the two perpendicular lines of the L-shaped car contour in LiDAR scans. If a point cluster has these two axes, it is likely a car. To account for cars recorded from behind or from the front that only show one axis, they determine a second axis based on the car's velocity vector when tracking localized objects. Using similar considerations, the authors define cyclists as one-axis objects and pedestrians as zero-axis objects. By examining the likelihood functions of their handcrafted features for each object class, Zhao et al. determine likelihood measures, which they employ to train a supervised classifier.

In 2015, Börcs et al. [6] used features like the length and width of the 2D bounding box as well as less intuitive 3D features from a principal curvature estimation and a side-view shape approximation with convex and concave hulls to train a support vector machine on a created labeled dataset. To understand support vector machines, the author again wants to refer to descriptive online resources [29]. The trained support vector machine functions as binary classifier and distinguishes between vehicles and static background objects.

In 2016, Tarko et al. [94] selected an object's dimension, speed, and position inside their defined region of interest as decision foundation. Based on these features and a created labeled dataset, they designed a multinomial logic model with a discrete outcome to distinguish between classes. If a tracked object is temporarily located on the sidewalk, the authors assume it is a pedestrian or a cyclist. To further distinguish these two classes, a speed thresh-

old is employed since cyclists normally move faster than pedestrians. If an object is never located on the sidewalk, the multinomial logic model computes probabilities how likely an object belongs to the remaining object classes based on the three defined metrics and the inferred information from the labeled dataset. The class with the highest probability is then assigned to the object in question.

In 2019, Zhao et al. [137] extracted three features from point clusters to distinguish between vehicles and pedestrians: (1) the total number of points in a cluster because vehicles should exhibit more points than pedestrians; (2) the 2D distance in the xy-plane between the reference point of a cluster and the LiDAR sensor; and (3) the direction of the point distribution within a cluster because pedestrian clusters are predominantly distributed vertically (they are higher than wide) while vehicle clusters are mainly distributed horizontally (they are wider than high). Employing these three features, the authors trained an artificial neural network with a created labeled dataset. By means of the supervised training, the network can classify a cluster as vehicle or pedestrian.

Also in 2019, Chen et al. [13] used five features to distinguish between vehicles, pedestrians, and deers: (1) the cluster height, (2) the cluster length, (3) the 3D distance between the LiDAR sensor and the nearest point of a cluster, (4) the number of points in a cluster, and (5) an estimation of the point distribution. Based on these features, the authors train and assess multiple supervised classifiers: the naive Bayes classifier [60], random forest classifier [8], and K-nearest neighbor classification [52]. Ultimately, they choose random forest because it demonstrates a moderately higher accuracy of 99.8 % on their test data.

Likewise in 2019, Wu et al. [111] chose six features for object classification in roadside LiDAR data after analyzing which features related studies had selected. In contrast to other classification approaches mentioned in this section, the classifier from Wu et al. relies on features requiring upstream tracking. Using these features, the authors investigate four supervised classifiers: (1) the naive Bayes classifier [60], (2) random forest classifier [8], (3) K-nearest neighbor classification [52], and (4) a support vector machine [7]. In experiments, random forest achieves the highest correct classification rate, a metric representing accuracy, of 75.8%; in comparison, naive Bayes achieves 56.7%, K-nearest neighbor classification 61.2%, and the support vector machine 58.6%.

In 2020, Zhang et al. [122] decided on four feature groups (28 individual features) to binary classify between vehicles and non-vehicles: (1) a vertical point distribution histogram with 20 bins for each object cluster, (2) the standard deviation in the three coordinate directions, (3) spatial dimensions, and (4) the area of a fitted 2D minimum area rectangle. Using these features a random forest classifier [8] and a support vector machine [7] are trained and compared to rule-based classification. Among the 28 individual features, the authors identify as the most important ones the area occupied by an object, its width, the standard deviation in x-direction, an object's length, and the standard deviation in y-direction (in this order). Trying multiple feature combinations and employing precision 2.3, recall 2.4, and the F1-score 5.3 as metrics, Zhang et al. summarize that all three approaches work similarly well with an F1-score above 92 % with the support vector machine leading when using all 28 features (rule-based classification only uses the area). Since rule-based classification achieves the highest recall of 99 % and thus retains more detected objects than the supervised classifiers, the authors choose this approach.

Also in 2020, Wu et al. [108] exploited three features to distinguish between vehicles and pedestrians: (1) the total number of points in a cluster, (2) the 2D distance of a cluster to the LiDAR sensor, and (3) the direction of the point distribution for the same reasons as Zhao et al. [137]. They use these features to train an artificial neural network that reaches 93.6 % detection rate on their test dataset.

### Open-Set Classification

In supervised as well as in unsupervised object detection scientists usually assume a closed set of classes, meaning any localized object belongs to one of the predefined object classes  $1, \dots, C$ . If an unknown object like a Segway driver appears in the point cloud, closed-set detectors would erroneously classify the unknown object as one the predefined object classes. Classifying a Segway driver as a pedestrian, for instance, could lead to an underestimation of velocity that the object gains. Open-set detectors assume more object classes than anticipated by scientists ( $1, \dots, C, C + 1, C + n$ ) to account for Segway drivers, buggies, kids on Kettcars or scooters, or a towed vehicle.

In 2021, Cen et al. [12] investigated open-set 3D object detection and designed an artificial neural network capable of assigning an “unknown” class in addition to the classes predefined by the authors. They named their network metric learning and unsupervised clustering (MLUC). Metric learning refers to the authors’ procedure to identify detected objects with a low confidence score. Objects with confidence score lower than the specified threshold, also receive a bounding box, but are classified as “unknown” as opposed to detected cars or pedestrians.



# Chapter 4

## Methodology

To accomplish 3D object detection in traffic, LiDAR is identified as the most promising sensor system because it outputs accurate 3D information about the scanned environment. Distances to objects as well as the extent of objects are measured directly and do not need to be estimated. In addition to 3D information, LiDAR provides sensor data by day and by night as it is lighting invariant.

The deployed LiDAR sensor is attached statically to road infrastructure instead of a vehicle because this way sensors perceive the environment from a more favorable elevated perspective, which reduces occlusion. After processing the sensor data, the intelligent infrastructure system can transmit the detected objects via vehicle-to-infrastructure (V2I) communication to autonomous and conventional vehicles. Knowing where other road users are located helps autonomous vehicles and drivers to navigate safely through traffic. If autonomous vehicles and drivers access this information in real-time, they can anticipate hazardous traffic situations seconds or even minutes before approaching them, making roadside sensors a valuable complement to on-board sensor systems.

Roadside and on-board sensor systems can be based on multiple sensors whose data developers fuse together. Multi-sensor solutions ensure redundancy but increase maintenance and costs [116]. As researchers regard a LiDAR sensor capable of working independently [133][106][3], this work investigates single-sensor-based object detection.

Most related studies about roadside LiDAR for unsupervised 3D object detection did not use publicly available datasets for experiments with their detectors since the majority of roadside LiDAR datasets were published during the last one and a half years. In fact, most studies employ their own LiDAR sensor, which they temporarily install on the road, and acquire data only accessible by the researchers involved in the study. In contrast, this study exclusively uses publicly available datasets that provide ground truth labels and allow an objective evaluation of the developed detector. This emphasis on transparency enables future studies to compare their detectors with the detector of this study such that progress becomes ascertainable, which was hardly possible in the past when the research practice in the field was to generate project-specific LiDAR data.

Another prevalent research practice this study avoids is not to grant access to the developed detector or algorithms. While most studies do not publish their code in a public repository, the detector described in this chapter can be accessed through a link<sup>1</sup>. Publishing code alongside a research paper enables other scientists to replicate stated results without reimplementing an algorithm that is only presented theoretically. Additionally, code open to the public can serve as code base and facilitates future developments.

Since source code tends not to be publicly accessible, software libraries like Open3D [140] do not incorporate the latest algorithms for, for instance, clustering. As a result, the latest

---

<sup>1</sup>[www.github.com/marcelbrucker/Unsupervised-3D-Object-Detection-Using-Roadside-LiDAR-Sensors](http://www.github.com/marcelbrucker/Unsupervised-3D-Object-Detection-Using-Roadside-LiDAR-Sensors)

algorithms cannot be deployed efficiently and time optimized, even if one reimplements an approach. In order to uphold the emphasized real-time demand, existing algorithms of software libraries are preferred in this study, even though the final result may not represent the state of the art and the highest possible precision.

Also worth mentioning on the technical side is that this study exploits two algorithms to extract the foreground containing road users from the point cloud: (1) ground segmentation and (2) background filtering. In contrast, most studies only employ one of the algorithms. By using both, foreground extraction is achieved more accurately, as seen in Table 5.3.

Related works often focus on localizing objects or on detecting only vehicles. Disregarding smaller road users such as pedestrians and cyclists results in incomplete scene understanding, especially in an urban area, where these smaller road users represent a substantial part among all road users. To consider and identify all road users, the proposed detector classifies objects in addition to localizing them. Classifying objects is achieved through heuristics, instead of deploying deep learning as normally targeted by studies in the field that do object classification.

Experimenting with public datasets, making accessible a complete roadside object detector, and combining algorithms to extract the foreground of the point cloud constitute the major contributions of this study. To develop an object detector, computer scientists use supervised learning or traditional unsupervised methods [55][99]. Using supervised learning for LiDAR-based 3D object detection is impeded by the absence of sufficiently large datasets. Artificial neural networks, the basis for supervised learning, are also prone to adversarial attacks and lack explainability, which diminishes approval chances in mass-production traffic applications. Ultimately, supervised learning is particularly unsuitable for roadside sensor data as roadside sensors do not move and generate largely uniform point clouds. The lack of diverse training data makes neural networks learn defectively as they overfit, even if the number of frames were vast. Because of these deficiencies, traditional unsupervised methods are preferred for roadside object detection. The study adopts an algorithmic chain often suggested in the field: algorithms to extract the foreground, clustering, and object classification [137][3][63][135][55][99][22][133][126][101]. The implemented processing chain of algorithms is visualized in Figure 4.1.

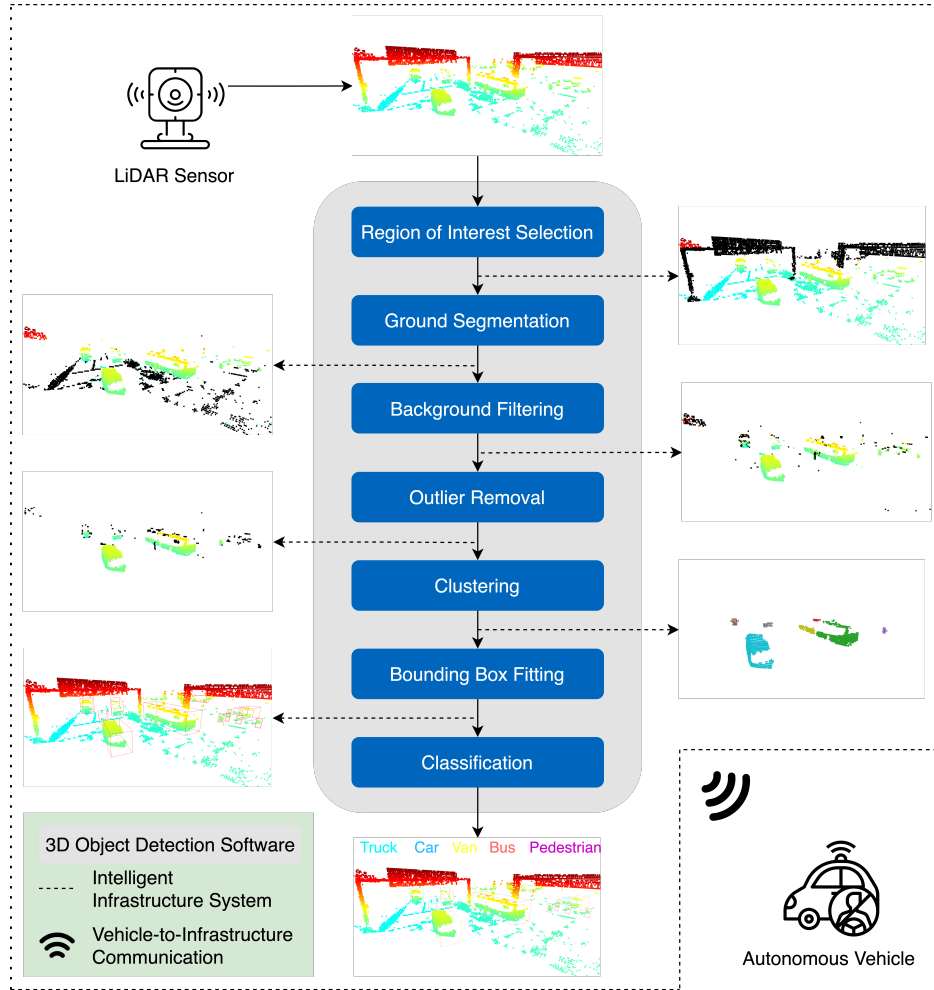
## 4.1 Region of Interest Selection

The first step a raw point cloud passes is the region of interest selection, in which irrelevant point regions are removed, so that merely the road and the road users remain. Irrelevant points are buildings and trees, which do not belong to the dynamic road users object detection in traffic is interested in. Since static objects like buildings and trees do not move between frames in roadside LiDAR scans, computer scientists locate them once in the data and can associate their corresponding point region as irrelevant for all future LiDAR scans. Remaining point regions constitute the region of interest.

Region of interest selection is a preprocessing step that reduces the number of points; for a quantification refer to Table 5.1. Reducing the number of irrelevant points decreases the data size and expedites downstream processing steps without affecting the detection performance [92]. For this reason, this study exploits the efficacy gained when selecting a region of interest.

The region of interest is different for each sensor location and computer scientists have to select the region anew when using data from another sensor. This study primarily develops a 3D object detector for R1 LiDAR frames of the A9-Dataset [19] generated at an urban intersection (R1 LiDAR frames from the highway are not included). The two Ouster LiDAR





**Figure 4.1:** Illustration of the proposed processing chain for unsupervised 3D object detection. Part of the intelligent infrastructure system is a roadside LiDAR sensor that generates raw point clouds, which are processed by a local computing unit in seven algorithmic steps. The black points depict the points removed in a step. Information about the detected objects is transmitted wirelessly to affected autonomous vehicles so that they can maneuver safely in traffic.

sensors installed on one of the four gantry bridges perceive the intersection from roughly 8 m above the ground. The northern Ouster LiDAR sensor is called Ouster north while the southern sensor is called Ouster south. Both sensors have their own regions of interest.

The study identifies four point regions in the LiDAR frames R1 that can be eliminated: (1) the back sensor area, (2) outside the sensor range, (3) the gantry bridges, and (4) two building blocks in the background. The back sensor area is the semicircle laying in the 180° behind the Ouster LiDAR sensors, away from the intersection. This area is largely obstructed by the road signs of the gantry bridge, visible in Figure 2.5, so the sensors mainly scan the road sign instead of the environment behind. As the sign obstructs road sections, the points in these 180° area are largely useless and consequently removed. The concerning points are found using the condition in Equation 4.1 ( $p_{i,x}$  is the x-coordinate of point  $p_i$  of the point cloud  $P$ ) given the coordinate system in point clouds (x points forward).

$$p_{i,x} < 0 \mid p_i \in P \quad (4.1)$$

Points outside the 120 m sensor range are very sparse and typically noise, so the detector should not consider them. Also, points below 8 m from the LiDAR sensor are noise as road

users do not move subterraneously. Neither are they found high above the ground, thus points fulfilling any of the conditions in Equation 4.2 are removed by the detector. A point  $\mathbf{p}_i$  with coordinate  $p_{i,z} = -10$  is located  $-10$  m below the LiDAR sensor (and here  $-2$  m subterraneous), because the sensor lies in the origin of its generated point cloud.

$$\begin{aligned}
 p_{i,x} < -120 \text{ m} \mid \mathbf{p}_i \in P \\
 p_{i,x} > 120 \text{ m} \mid \mathbf{p}_i \in P \\
 p_{i,y} < -120 \text{ m} \mid \mathbf{p}_i \in P \\
 p_{i,y} > 120 \text{ m} \mid \mathbf{p}_i \in P \\
 p_{i,z} < -10 \text{ m} \mid \mathbf{p}_i \in P \\
 p_{i,z} > 0 \mid \mathbf{p}_i \in P
 \end{aligned} \tag{4.2}$$

The pillars of the three other gantry bridges at the intersection makes the laser beams, emitted from the fourth gantry bridge, scatter. Scattering is a challenge for background filtering as the deployed algorithm does not identify points scattered at road infrastructure consistently as background points. To prevent incorrect background filtering at pillars, requiring vigorous outlier removal in the downstream step, the pillars are removed, similarly to the previous two point regions, with predefined thresholds. However, the thresholds for each coordinate are expressed using an axis aligned bounding box, a bounding box with all three rotating angles fixed to zero. The position and the extent of one axis aligned bounding box are chosen such that one box encloses one third of a gantry bridge; two boxes enclose the two pillars of a bridge and one box encloses the road signs in between above the street as illustrated in Figure 4.2. Since each of the three gantry bridge requires three axis aligned bounding boxes, the detector stores the position and the extent of nine boxes. Defining a bounding box in the point cloud library Open3D [140], enables users to find points laying inside the box easily. All points inside one of the nine bounding boxes are identified as belonging to a gantry bridge and removed.

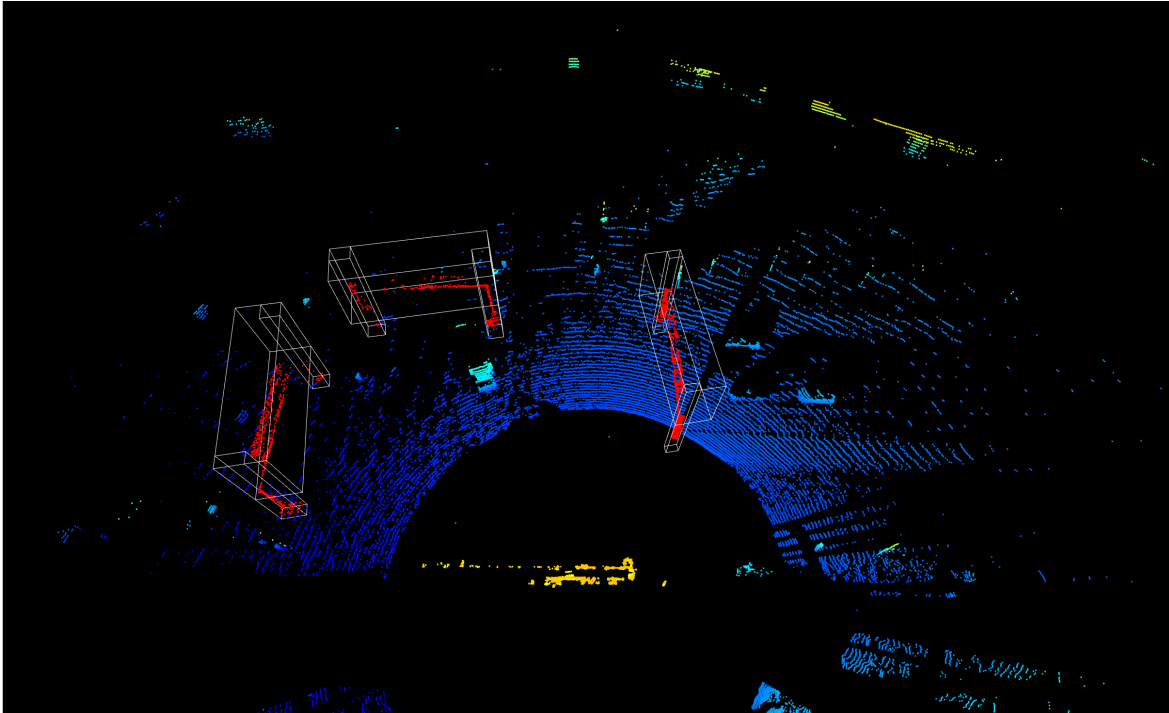
Aside from those three gantry bridges, the bridge where the sensors are attached to is also partly visible in the point cloud due to reflections of laser beams. The reflections of the sensor station are removed using a single axis aligned bounding box (see Figure 4.3), amounting to ten boxes in total for finding irrelevant points.

Lastly, to enclose the building blocks in the two street corners on the opposite side of the sensor station, two oriented bounding boxes, only rotating around the z-axis, are defined. The two enclosed building blocks along with both oriented bounding boxes are highlighted in Figure 4.4. The points of those building blocks are also removed by deploying the same functionality in Open3D that was used for the gantry bridges.

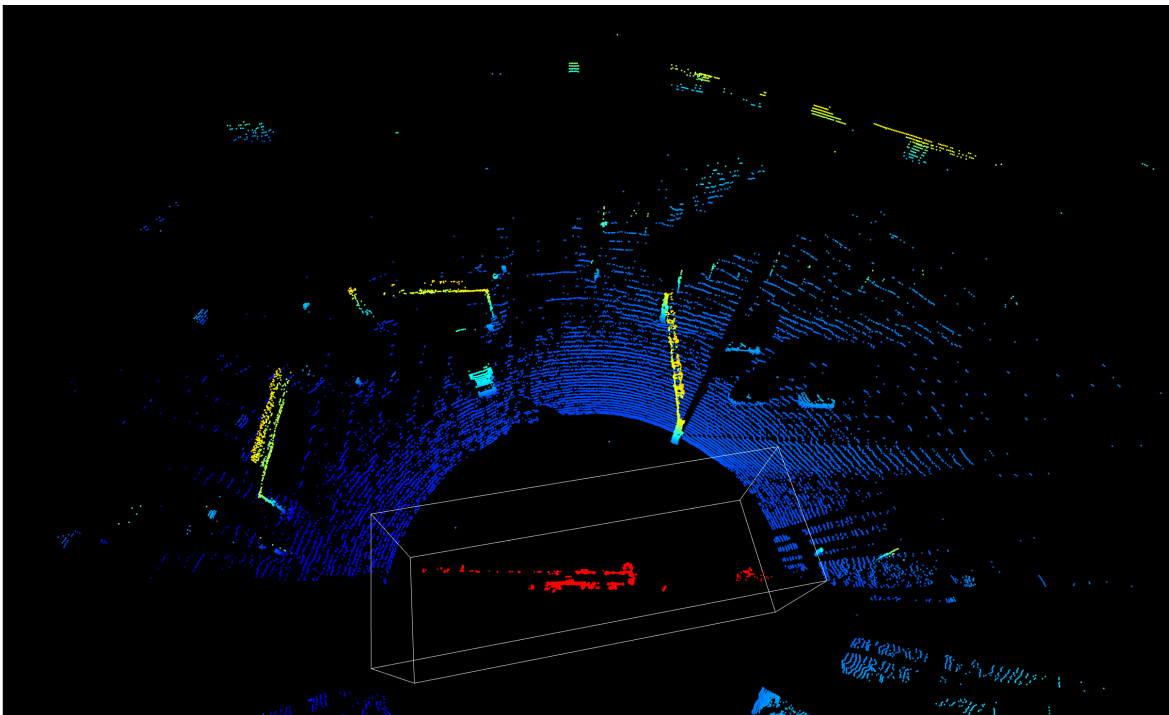
By eliminating the irrelevant point regions, the detector extracts a region of interest significantly reducing the size of a point cloud. From 131,072 points in the raw point cloud, on average 17,240 remain after the first processing step (averaged over the raw R1\_S5 batch with 300 LiDAR frames from the Ouster north). This significant reduction is due to 80,442 points  $\mathbf{p}_i = (0,0,0) \in R^3$  (averaged the same way) in the origin resulting from laser beams that are reflected by surfaces within too short distance to the sensor (the road signs behind the sensor) or from laser beams that point at targets outside of the sensor range.

## 4.2 Ground Segmentation

The second step processes a point cloud already containing significantly less points. The remaining point cloud primarily portrays the road and present road users as a large portion

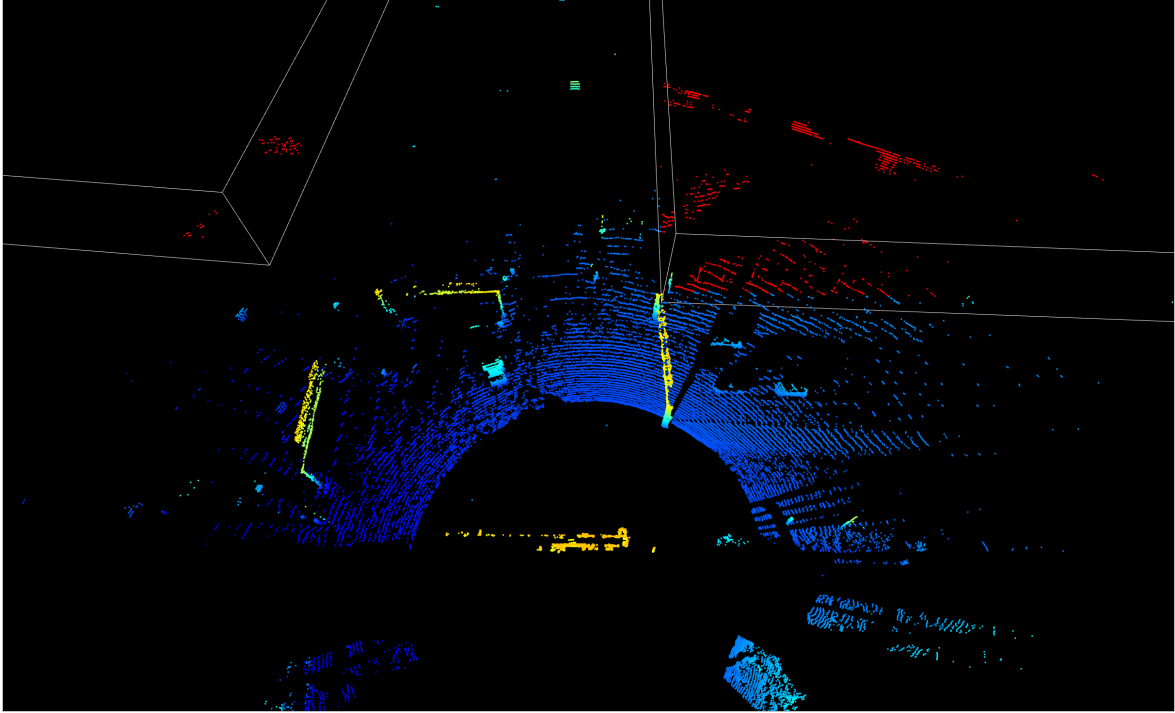


**Figure 4.2:** Removed gantry bridges from LiDAR point cloud (excluded points, axis aligned bounding boxes).



**Figure 4.3:** Removed sensor station from LiDAR point cloud (excluded points, axis aligned bounding boxes).

of the background is eliminated when not belonging to the predefined region of interest. Removing the ground or road depicted in the point cloud is the goal of ground segmentation. To segment the ground, the detector exploits the low relief at the intersection of the sensor station providing the R1 LiDAR data in the A9-Dataset. A flat ground enables computer scientists to model a plane close to which are the road points. The study obtains the plane



**Figure 4.4:** Removed building blocks from LiDAR point cloud (excluded points, oriented bounding boxes).

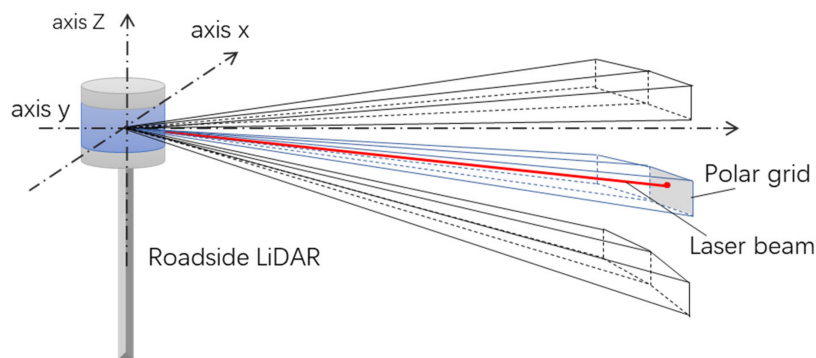
model once for each sensor by merging point clouds of 300 LiDAR frames and estimating the plane parameters with classic RANSAC (see Section 3.2). To identify road points, the detector computes the Euclidean distance from the modeled plane to all points within a target point cloud. Points within a 0.2 m distance threshold to the plane are assigned to ground points while the rest are non-ground points. Ground points, which depict the road in the point cloud, are discarded in this step efficiently, but due to time constraints not using a state-of-the-art algorithm (see runtime comparisons in Chapter 5) and under the assumption that the road at the intersection is flat. The detector passes the extracted non-ground points to the next step.

### 4.3 Background Filtering

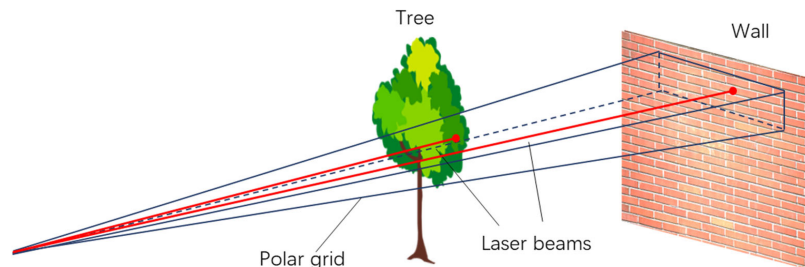
To filter further background points from the extracted point cloud, which are missed by the region of interest selection, this study adopts the background filtering algorithm based on distance thresholds from Zhang and Jin [126] from 2022. To apply their algorithm, the 3D points in Cartesian coordinates are reverted to spherical coordinates, whose vertical and horizontal angles can be considered finite and, when discretized, suitable as 2D coordinates in a conceptual image. Discretizing the computed angles is necessary as real LiDAR scans show fluctuations. The number of columns  $m$  of the conceptual image is specified by the number of laser channels. The used Ouster LiDAR sensors aggregate 64 laser channels, hence the resulting image has  $m = 64$  columns. Figure 4.5 illustrates three vertically adjacent laser channels at two different horizontal positions, which together display six pixels of the conceptual image. The total number of rows  $n$  of the conceptual image is specified by the horizontal angular resolution  $\alpha$  of the sensor. Instead of estimating  $n$  with the equation  $n = \frac{360^\circ}{\alpha}$  ( $360^\circ$  horizontal field of view), we can also exploit the consistent output of 131,072 points per scan of the Ouster sensors to compute  $n = \frac{131,072}{64} = 2048$  number of rows. The

image size then equals  $64 \times 2048$  and its pixels quantify the distance at the respective angle combination. Figure 4.6 shows two horizontally adjacent laser beams leading to horizontally adjacent pixels with distinct distance values due to the tree being closer to the sensor than the wall.

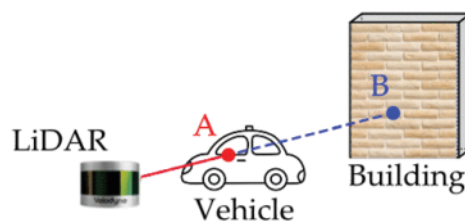
Not all pixels are filled with a distance value as many points were already removed before the transformation into spherical coordinates by the previous two processing steps. The distance value of filled pixels is compared with the corresponding distance threshold stored in a constructed background image of the same size. If the value is lower than the threshold, the respective 3D point is retained as foreground; if not, the respective point is removed as it is considered background. The distinction is valid since the background, for example a building wall, displayed in the background image is more distant to the LiDAR sensor than a road user passing this point (or angle combination) as illustrated in Figure 4.7.



**Figure 4.5:** Three adjacent laser channels each pointing in the same two horizontal directions [55].



**Figure 4.6:** Two adjacent laser beams from the same laser channel capturing different distances. The two laser beams are apart according to the horizontal resolution  $\alpha$  [55].



**Figure 4.7:** A building wall representing background (B) is more distant to the LiDAR sensor than a passing road user (A) [55].

The background image storing the distance thresholds to distinguish foreground from background is constructed through a set of raw point clouds recorded during the night with low traffic. Following empirical results from Wu et al. [106], this study aggregates 3000

raw point clouds. After aggregating this set of point clouds, the points are transformed into spherical coordinates and stored analogously in a set of  $64 \times 2048$  images. By regarding the same pixel from the whole set of 3000 images at a time, the distance thresholds of the final background image are determined. Instead of choosing the maximum or mean distance reached at every angle combination, Zhang and Jin suggested their coarse-fine triangle algorithm (see Algorithm 1) to determine a suitable reference distance. The coarse-fine triangle algorithm computes the reference distance without requiring any parameters. Part of it is the triangle algorithm that is based on the histogram computed at each angle combination and that is best described using Figure 4.8. The histogram illustrates the distribution of reached distances discretized in a data-dependent and automatically determined number of bins. Each bin stores how many images are within the distance range that the bin represents. Most angle combinations show a characteristic maximum peak at far distance measured by the majority of point cloud scans. From the left edge of the maximum peak, a diagonal line is drawn to the peak at distance 0 m (often zero counts). Starting from the diagonal line, orthogonal lines are determined. The orthogonal line with maximum line length reveals the threshold.

---

**Algorithm 1: Coarse-Fine Triangle Algorithm**


---

**Input** : An array  $d = [d_i], i = 1, 2, \dots, n$ , where  $d_i$  is a distance.  
**Output**: Reference distance threshold.

```

1 if countZeroElements( $d$ ) >  $0.8 \cdot n$  then
2   |  $threshold \leftarrow \max(d)$ ;
3 else
4   |  $d \leftarrow \text{removeZeroElements}(d)$ ;
5   |  $nBins \leftarrow \text{int}(\max(d))$ ;
6   |  $hist \leftarrow \text{histogram}(d, nBins)$ ;
7   |  $binEdgeMax \leftarrow hist.getBinEdge(hist.argmax())$ ;
8   | for  $i \leftarrow 0$  to  $n - 1$  do
9     |   | if  $d_i > binEdgeMax + 2\sigma$  then
10    |   |   |  $d.remove(i)$ ;
11    |   |   end if
12   |   end for
13   |  $maxDist \leftarrow \text{int}(\max(d))$ ;
14   |  $nBins \leftarrow maxDist / 100$ ;
15   |  $hist \leftarrow \text{histogram}(d, nBins)$ ;
16   |  $threshold \leftarrow \text{triangle}(hist)$ ;
17 end if
18 return  $threshold$ 

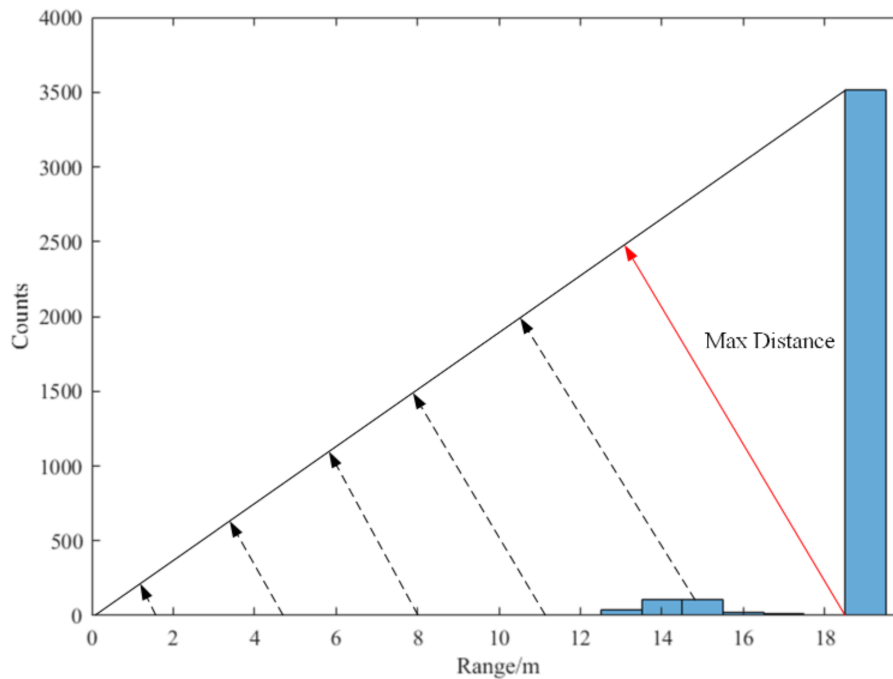
```

---

After dividing the points of the remaining target point cloud into foreground and background deploying the distance thresholds, the foreground points are transformed back to Cartesian coordinates. The determined foreground point cloud is then passed to the next processing step.

#### 4.4 Outlier Removal

The three algorithms to extract the foreground sometimes make mistakes when letting points pass although they are background. Those few background points not representing any road



**Figure 4.8:** Distance threshold determined by the triangle algorithm maximizing orthogonal lines on the drawn diagonal line [126].

user are outliers and diminish the detection result. To reduce the impact of outliers, this study equips its detector with an algorithm to remove outliers. The algorithm used is called radius outlier removal (ROR) and is provided by the open-source point cloud library Open3D [140]. ROR counts the number of adjacent points within a 0.8 m search radius around a point. If a point has inside its imaginable search sphere fewer than 15 adjacent points, the point is considered an outlier and removed.

ROR has low computational costs and is extremely fast due to the provision by Open3D (2.40 ms, averaged over the R1\_S5 batch with 300 LiDAR frames from the Ouster north). In qualitative visual assessments, the algorithm prevailed over statistical outlier removal (SOR). Through these assessments, the parameters for ROR were set as well, in addition to a quantitative analysis as in Table 5.4.

## 4.5 Clustering

To cluster the foreground point cloud into distinct groups each representing a road user, the detector employs density-based spatial clustering of applications with noise (DBSCAN), which does not require specifying the number of clusters or road users. DBSCAN inputs the foreground point cloud in 3D and outputs the cluster label for each point if the point is not identified as noise. Points being considered noise receive the label -1. DBSCAN finds noise differently than ROR of the previous step, for what reason more points may be discarded during clustering. Only a marginal difference in the R1\_S5 batch is observed: the algorithm removes on average 0.10 further points per point cloud. To find cluster labels for all points, DBSCAN analyzes the density of the point cloud taking into consideration (1) 3 minimal number of points are required to form a cluster and (2) the maximum search radius is set to 0.8.

DBSCAN is commonly used in roadside object detection [137][86], so this study adopts

the experience of other scientists. The algorithm is available in several libraries, just like in Open3D, where it clusters a foreground point cloud in 3.24 ms (averaged on the R1\_S5 batch). Just like for outlier removal, the clustering parameters were determined in qualitative visual assessments using complex traffic situations with road users moving side by side. The parameters must ensure that these road users are separated. In addition to visual analysis, parameter selection for DBSCAN is examined in the ablation study of Table 5.4.

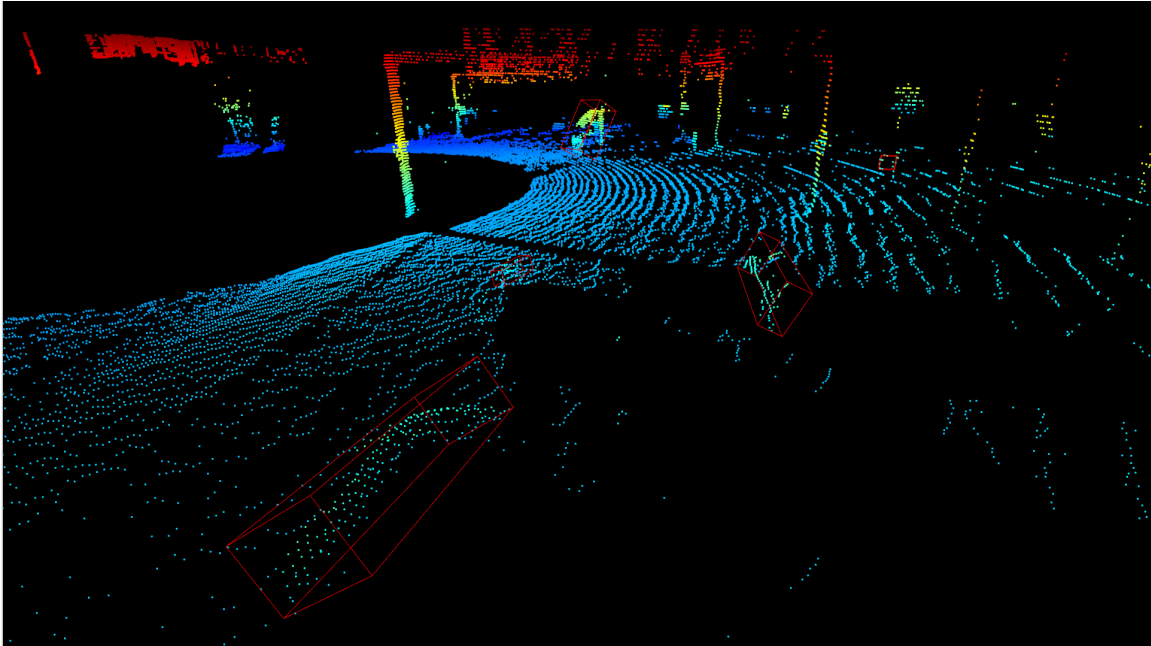
## 4.6 Bounding Box Fitting

A single reference point is a poor approximation of a vehicle body; the contour of a point cluster too when recorded incompletely, which is common with a single LiDAR sensor [137][82][72][45][116][135]. To grasp an object's position, dimension, and orientation, the detector uses 3D bounding boxes as shape model for each cluster. 3D bounding boxes are described by seven parameters: three coordinates for the box center, the box extent in three dimensions (width, length, height), and a heading (or yaw) angle. Determining the seven parameters for each detected point cluster is done through functionalities that are provided by the point cloud library Open3D [140] and that are also employed to select the region of interest in a previous processing step. Open3D uses the cluster labels found by DBSCAN to recognize which points of the foreground point cloud belong to the cluster it is creating the 3D bounding box for. By considering only those points, Open3D's algorithm computes the convex hull enclosing the entire cluster (for an explanation refer to Section 3.7). Based on the convex hull, the algorithm continues to determine the principal components via principal component analysis (PCA), a technique to find the dominant axis inside an object. This dominant axis often coincides with the length and orientation of the longer side edge of a vehicle. The subsequent dominant axes then correspond to the front or rear edge, and reflect the height of the object, which lastly determines all box parameters.

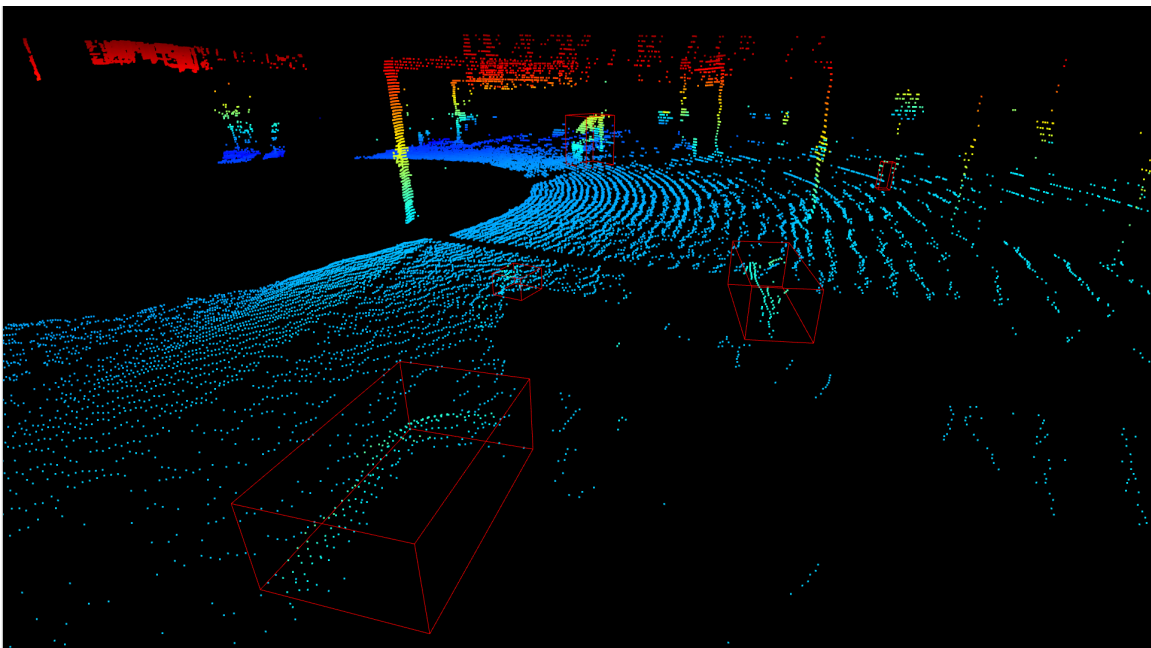
The result is an oriented bounding box, which often demonstrates skewed box parameters due to the incomplete vehicle contour in point clouds; for instance in Figure 4.9, oriented bounding boxes appear strangely rolled or tilted so that the bottom side does not coincide with the ground plane. For this reason, only the yaw angle describing the orientation around the z-axis is retained from this box. The pitch and roll angle around the x- and y-axes are set to zero because we assume a (locally) flat world. The box center and the box extent are defined by an axis aligned bounding box additionally fitted in this processing step. By combining the six parameters of the axis aligned bounding box with the yaw angle of the oriented bounding box, the detector determines a box only rotated around the z-axis. The result of this consideration is visualized in Figure 4.10.

A last essential adjustment to this box becomes evident if we consider road users not being scanned down to the tires or feet by the LiDAR sensor at the lower edge of its vertical field of view. If the lower object surfaces are missing in the point cloud scan, as in Figure 4.11, the box floats above the road since the lowest cluster point defines the bottom side of the box. To compensate this flaw, the detector fixes the bottom side of a box to the road by adopting the road plane parameters originally determined for ground segmentation. Figure 4.12 illustrates the effectiveness of this adjustment.





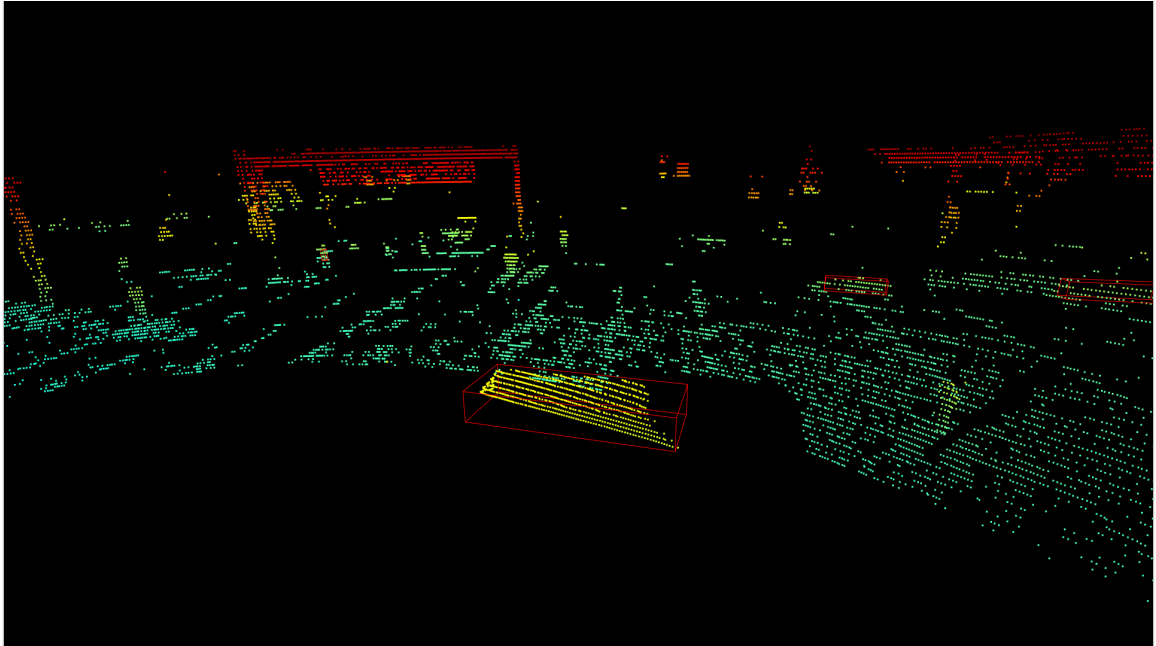
**Figure 4.9:** Unmodified oriented bounding boxes from Open3D are rolled and tilted around localized vehicles.



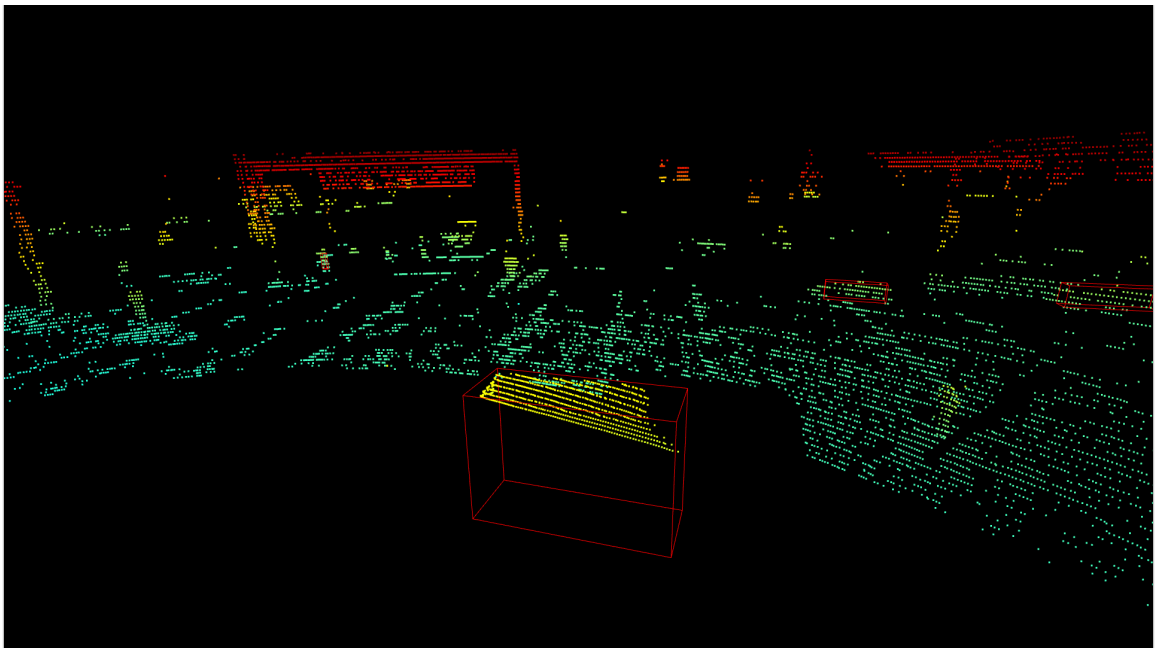
**Figure 4.10:** Combining information from an axis aligned bounding box with the yaw angle from an oriented bounding box leads to a bounding box that is only rotated around the z-axis.

## 4.7 Classification

After obtaining bounding boxes as shape model for localized objects, the goal is to assign them to a class. The A9-Dataset categorizes road users into ten classes: (1) car, (2) truck, (3) trailer, (4) van, (5) motorcycle, (6) bus, (7) pedestrian, (8) bicycle, (9) emergency vehicle, and (10) other that enables open-set classification following Cen et al. [12]. This is a detailed categorization and challenging for detectors, especially when employing classification



**Figure 4.11:** Object from which only the upper part is captured.



**Figure 4.12:** Corrected incomplete scan of an object by fixing the bottom side of the bounding box to the road.

approaches like in this study.

The detector of this study relies on heuristics for classification following Zhang et al. [122], who opted for a rule-based approach after comparing it with supervised classifiers. While Zhang et al. based their classification solely on the area occupied by an object, this detector considers an object's width, length, and height to assign it to a class. Only the following classes are assigned since heuristics better handle few classes: (1) car, (2) bus, (3) pedestrian, (4) bicycle, and (5) other, where it must be specified that some of them include multiple classes as they are labeled in the A9-Dataset.

- Car: combines (1) car, (4) van, and (9) emergency vehicle
- Bus: combines (2) truck, (3) trailer, and (6) bus
- Pedestrian: remains (7) pedestrian
- Bicycle: combines (5) motorcycle and (8) bicycle
- Other: remains (10) other

After the object is classified, the detector expands small object dimensions in horizontal direction to predefined default object dimensions to account for occlusions and incompletely scanned object contours. The default object dimensions have been determined using the average object dimensions in the A9-Dataset. However, the defaults do not correspond exactly to the computed averages, since, for instance, the average dimensions for bicycles and motorcyclists must be combined for a unifying bicycle class. In addition to addressing occlusions and incompletely scanned object contours, default object dimensions improve the evaluation results achieved by the detector, as a comparison between Table 5.7 and Table A.1 reveals. Note that the vertical dimension remains unchanged, since it is assumed that the object heights are captured completely and unobscured by LiDAR sensors.

All object classes are characterized through thresholds that an object's dimensions must satisfy to be assigned to the class in question.

- Pedestrian: the object's width and length must be both between 0.1 m and 0.8 m. In addition to the horizontal extent, the object must have a height between 0.5 m and 2 m. With respect to expanding object dimensions: if a detected pedestrian's width or length remains under 0.8 m, the concerning dimension is expanded to 0.8 m.
- Bicycle: the object's width or length (no clear distinction possible since a 90° rotation around the vertical axis swaps the two quantities) must be between 0.5 m and 1.7 m. The resulting counterpart must be between 0.2 m and 1 m. In addition to the horizontal extent, the object must have a height between 0.5 m and 2 m. If the longer side of a detected bicycle's length and width remains under 1.7 m, the concerning dimension is expanded to 1.7 m. The corresponding shorter dimension is expanded to 0.9 m if it is shorter than 0.9 m.
- Car: the object's width or length must be between 1.5 m and 5 m. The resulting counterpart must be between 0.1 m and 2 m. In addition to the horizontal extent, the object must have a height between 1 m and 2 m. If the longer side of a detected car's length and width remains under 4.3 m, the concerning dimension is expanded to 4.3 m. The corresponding shorter dimension is expanded to 1.9 m if it is shorter than 1.9 m.
- Bus: the object's width or length must be between 6 m and 20 m. The resulting counterpart must be between 2.2 m and 3 m. In addition to the horizontal extent, the object must have a height between 2.5 m and 5 m. If the longer side of a detected bus' length and width remains under 12 m, the concerning dimension is expanded to 12 m. The corresponding shorter dimension is expanded to 2.3 m if it is shorter than 2.3 m.
- Other: the object is classified as other if the previous four classes do not match.

The presented classification based on heuristics does not require any ground truth data and leaves the detector unsupervised. Heuristics are comprehensible, implemented quickly,

and easily adaptable. With an average of 0.32 ms (averaged over the R1\_S5 batch with 300 LiDAR frames from the Ouster north), heuristics area also fast and thus respect the emphasized efficiency in this study.

## Chapter 5

### Evaluation

The developed 3D object detector is evaluated on real static LiDAR data coming from public datasets like the A9-Dataset [19]. Real data is to be preferred over simulated data to avoid a simulation-to-real (sim2real) gap, emerging when a detector trained on simulated data performs worse on real data. Datasets for object detection contain raw sensor data, ground truth labels, and additional information such as the date and time of recording, weather conditions, the sensor location, or calibration values. Ground truth labels portray the road users actually present in the scene through 3D bounding boxes along with an assigned class. This study compares the ground truth with the predictions when processing the raw sensor data frame by frame. Comparing the precision quantitatively requires metrics and this study employs two frequently used metric procedures: (1) mean average precision (mAP) by means of intersection over union (IoU) in 3D, and (2) average precision (AP) and average recall (AR).

The first procedure regards individual objects, meaning their shape and pose expressed through 3D bounding boxes as well as their class. By utilizing this object description, individual predicted objects are compared with individual ground truth objects in an evaluation that is explained in detail in Section 2.9. Following this evaluation, the final result is the mAP.

In contrast to evaluating individual objects, studies on 3D object detection using roadside LiDAR sensors often assess the extracted foreground, for instance obtained through a proposed background filtering algorithm [58]. The extracted foreground is considered as a whole, without the subdivision into point clusters or objects. For assessing the extracted foreground, scientists employ the notion of true positive (TP) points, false positive points (FP), and false negative points (FN). TP points reside inside ground truth bounding boxes and belong to the extracted foreground. FP points are part of the extracted foreground although they do not reside in any ground truth bounding box and should have been filtered as background. FN points are filtered points not belonging to the extracted foreground although they reside inside a ground truth bounding box. After determining these three figures, scientists compute the precision ( $\text{Precision}_i$ ) and recall ( $\text{Recall}_i$ ) for a single LiDAR frame  $i$  with Equations 2.3 and 2.4. By averaging over a set of frames  $[0;h]$ , as in Equation 5.1 and 5.2, scientists can compute an average precision (AP) and an average recall (AR), which serve as evaluation metrics. To combine AP and AR into a single metric, scientists use the F1 score, which is calculated as in Equation 5.3.

$$\text{Average Precision (AP)} = \frac{1}{h} \sum_i^h \text{Precision}_i \quad (5.1)$$

$$\text{Average Recall (AR)} = \frac{1}{h} \sum_i^h \text{Recall}_i \quad (5.2)$$

$$\text{F1 Score} = 2 \cdot \frac{\text{AP} \cdot \text{AR}}{\text{AP} + \text{AR}} \quad (5.3)$$

Preferring such an evaluation is due to many studies in the field that investigate single algorithms for ground segmentation [98] or background filtering [58] instead of entire detection pipelines, which precludes an object-centric evaluation.

In addition to examining the precision, this study emphasizes processing speed, which is why the runtime of the detection pipeline as well as its algorithmic modules is measured. Studies that state the runtime  $T$  do so in milliseconds or in the reciprocal frequency  $f = \frac{1}{T}$  measured in hertz.

The processing speed is related to the number of redundant background points that the detector manages to filter in early processing steps. These redundant points do not contribute to a higher precision of detecting objects. On the contrary, they hamper both the precision and the processing speed.

Lastly, qualitative visual results provide insights as to what extent the detection performances suffices an application. Even more important is to examine visual results because they indicate how quantitative results can be improved. For this reason, the chapter involves illustrations of detected objects in point clouds.

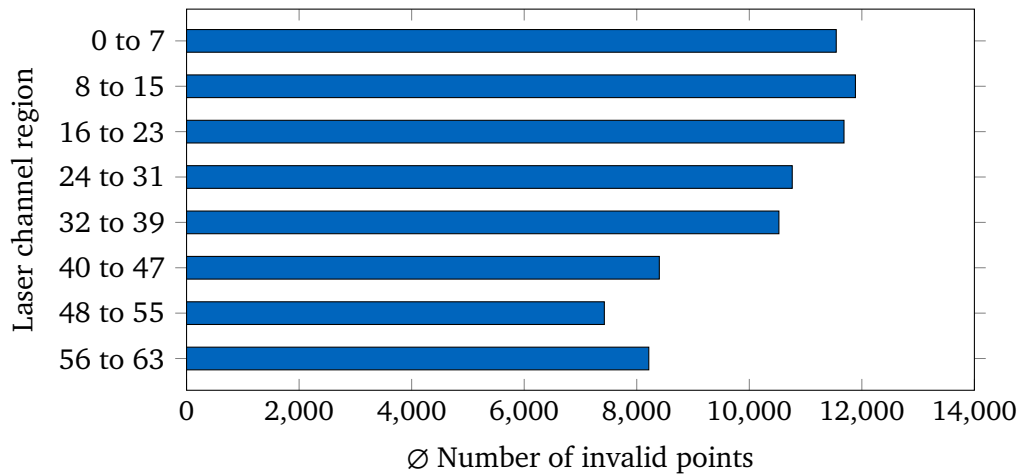
### A9-Dataset

The R1 LiDAR frames from the A9-Dataset used here were recorded at an urban intersection at different times, meaning during the day and at night, which implies low and high traffic volumes. A balanced subset of the R1 LiDAR frames constitute a test set that fosters an unbiased evaluation, what this study is interested in and hence its results refer to the test set, unless otherwise noted. This test set represents 10 % of the total A9-Dataset as well as 10 % of the individual sequences or batches, a balance resulting from the use of stratified sampling. Also important is that the test set does not contain points  $p_i = (0, 0, 0) \in R^3$  in the origin, unlike the R1\_S5 batch described in the previous Chapter 4. Point clouds without these points are processed significantly faster by the detector (see Table 5.5), since the number of points in the origin is considerable. This peculiarity is examined in the following.

## 5.1 Analysis of Missed LiDAR Reflections

The two Ouster LiDAR sensors used in the Providentia++ project are notable for a high number of missed reflections, reflections that occur when a laser beam hits an obstacle too close to the sensor or when a laser beam is not reflected by any obstacle within the sensor range. In either of these scenarios, missed reflections manifest themselves through invalid points  $p_i = (0, 0, 0) \in R^3$  in the origin. Section 4.1 indicates that on average 80,442 points in a raw point cloud with 131,072 points are such missed reflections. One assumption is that the LiDAR sensor is not optimally oriented but tilted too far upward, causing many laser

channels to point into the sky instead of at the intersection. To verify this hypothesis, the invalid points in the origin are counted across the raw R1\_S5 batch with 300 LiDAR frames from the Ouster north, averaged, and divided into eight laser channel regions. The top laser channels have the lowest tilt angle and could be the most likely to suffer from invalid points.



**Figure 5.1:** Average number of invalid points  $p_i = (0, 0, 0) \in R^3$  in the origin in a point cloud due to missed laser beam reflections. This number per laser channel is divided into eight laser channel regions, each representing eight of the total 64 channels.

Figure 5.1 provides an indication that the top 24 channels indeed might be tilted too far upward. There is a decreasing trend from top to bottom and the channels with the fewest missed reflections (48 to 55) have 38% less than the channels with the most missed reflections (8 to 15). However, the last channel region (56 to 63) has again more missed reflections. This phenomenon requires a closer examination of whether the sensors should be tilted further.

## 5.2 Point Reduction

Table 5.1 quantifies how many redundant background points in a point cloud the detector manages to filter in each module. The table also states how many points a module passes to the subsequent one and how large the relative reduction of the point cloud is. The input is an unprocessed point cloud, where unprocessed does not concern the elimination of the invalid points in the origin. These invalid points are not included in the test set.

The first module selects a region of interest with average 9,032 points, extracted from the unprocessed point cloud with average 29,995 points. Removing average 20,963 points or 69.89% is a significant reduction already in the first processing step and proves the effectiveness of selecting a region of interest. The total average data compression rate amounts to 97.38% for a point cloud without invalid points. A point cloud with these points, like in the R1\_S5 batch, has unprocessed 131,072 points yielding a total average data compression rate of 99.40%.

## 5.3 Average Precision and Average Recall

Table 5.2 compares ground segmentation algorithms that were examined in the course of this research work. A precomputed plane model is obtained by merging several point clouds and

**Table 5.1:** A9-Dataset test set evaluation regarding the reduction of the points of a point cloud through the different detector modules. Clustering based on DBSCAN finds outliers differently than the employed outlier removal algorithm and reduces the number of points additionally.

Module	∅ Removed Points	∅ Output Points	∅ Rel. Point Red.
Unprocessed	0	29,995	0 %
Region of Interest Selection	20,963	9,032	69.89 %
Ground Segmentation	7,215	1,817	24.05 %
Background Filtering	797	1,020	2.66 %
Outlier Removal	226.99	793.01	0.76 %
Clustering	0.37	792.64	0.001 %
Total	29,202.36	792.64	97.38 %

storing the computed plane parameters to process target point clouds. For details, refer to Section 4.2.

**Table 5.2:** A9-Dataset test set evaluation of ground segmentation algorithms. Bold and underline denote the best and second best results, respectively.

Algorithm	AP	AR	F1 Score	∅ Runtime in ms
RANSAC [24]	<b>0.61</b>	<u>0.48</u>	<u>0.54</u>	<u>3.16</u>
Slope-Robust Cascaded [62]	<b>0.61</b>	0.45	0.52	2,536.13
Jump-Convolution-Process [83]	<u>0.60</u>	0.47	0.53	1,4463.75
Precomputed Plane Model	<b>0.61</b>	<b>0.51</b>	<b>0.55</b>	<b>1.26</b>

Because of the significant reduction in runtime to 1.26 ms when using a precomputed plane model for ground segmentation, the detector deploys this approach.

To demonstrate the effectiveness, regarding precision, of multiple algorithms when extracting the foreground point cloud, Table 5.3 states results of an ablation study, where individual detector modules were alternately deactivated.

**Table 5.3:** A9-Dataset test set ablation study of detector modules, a study in which individual modules are alternately deactivated to extract the foreground in a point cloud. The abbreviated module names stand for RIS = region of interest selection, GS = ground segmentation, BF = background filtering, OR = outlier removal, and C = clustering. Bold and underline denote the best and second best results, respectively.

Set-Up	AP	AR	F1 Score
BF	0.12	<b>0.80</b>	0.21
RIS + BF	0.46	<u>0.77</u>	0.58
RIS + GS + BF	0.50	<u>0.74</u>	<u>0.60</u>
RIS + GS + BF + OR	<u>0.60</u>	0.51	0.55
RIS + GS + BF + C	0.53	0.72	<b>0.61</b>
RIS + GS + BF + OR + C	<b>0.61</b>	0.51	0.55

The results of Table 5.3 show that each module contributes to a higher average precision (AP), even if all modules combined do not yield the highest recall and the highest F1 score. Recall signifies how many ground truth objects are detected, meaning if a detector predicts objects everywhere in the scene, it has detected all ground truth objects and obtains a perfect recall. This finding is indicated in Table 5.3 by the high recall when many modules are



inactive for extracting the foreground point cloud.

Precision indicates how many detected objects truly exist according to the ground truth. Predicting more than the actual objects present leads to a higher number of FP detections, which reduces the precision. A predicted object that does not really exist triggers a self-driving car to take evasive action, which might endanger traffic. Frequent prediction of non-existent objects impedes traffic flow and additionally reduces the credibility of machine perception, which is why this study considers precision to be more important than recall.

The combination without outlier removal outputs a foreground point cloud that is noisy and detects many FP objects, resulting in a lower AP of 0.53. To prevent FP detections and to increase the AP to 0.61, outlier removal is necessary.

Table 5.4 is an ablation study conducted to determine the parameters that the detector uses and that are stated in Chapter 4. The ablation study does not cover region of interest selection and background filtering. Region of interest selection, the way it currently functions, depends on too many thresholds, which are better determined visually. Background filtering has no specifiable parameters. The algorithm determines necessary parameters by itself, which is a great advantage of it.

**Table 5.4:** A9-Dataset test set ablation study of foreground extraction in a point cloud, a study in which different parameters of the adjustable algorithms are tested.  $d_{max}$  is the maximal Euclidean distance up to which a point is considered part of the precomputed plane model.  $n_{adj}$  is the number of adjacent points within the search radius  $r_{search}$  required for a point to not be removed as outlier.  $n_{min}$  is the minimum required number to form a point cluster and  $r_{search}$  defines the maximum search radius. Bold and underline denote the best and second best results, respectively.

Set-Up			AP	AR	F1 Score
Ground Segmentation	Outlier Removal	Clustering			
$d_{max} = 0.2$	$n_{adj} = 30, r_{search} = 0.8$	$n_{min} = 3, r_{search} = 0.8$	0.56	0.38	0.45
$d_{max} = 0.2$	$n_{adj} = 5, r_{search} = 0.8$	$n_{min} = 3, r_{search} = 0.8$	0.58	<b>0.66</b>	<b>0.62</b>
$d_{max} = 0.2$	$n_{adj} = 15, r_{search} = 1.2$	$n_{min} = 3, r_{search} = 0.8$	<u>0.60</u>	<u>0.56</u>	0.58
$d_{max} = 0.2$	$n_{adj} = 15, r_{search} = 0.4$	$n_{min} = 3, r_{search} = 0.8$	0.56	0.32	0.41
$d_{max} = 0.2$	$n_{adj} = 15, r_{search} = 0.8$	$n_{min} = 10, r_{search} = 0.8$	0.59	0.50	0.55
$d_{max} = 0.2$	$n_{adj} = 15, r_{search} = 0.8$	$n_{min} = 3, r_{search} = 1.2$	<b>0.61</b>	0.51	0.55
$d_{max} = 0.2$	$n_{adj} = 15, r_{search} = 0.8$	$n_{min} = 3, r_{search} = 0.4$	<u>0.60</u>	0.51	0.55
$d_{max} = 0.1$	$n_{adj} = 15, r_{search} = 0.8$	$n_{min} = 3, r_{search} = 0.8$	<u>0.60</u>	0.52	<u>0.56</u>
$d_{max} = 0.2$	$n_{adj} = 15, r_{search} = 0.8$	$n_{min} = 3, r_{search} = 0.8$	<b>0.61</b>	0.51	0.55

The outcome of the ablation study in Table 5.4 confirms the chosen parameter values. Following the previous ablation study, precision is considered to be most important. The precision might be further increased through a comprehensive parameter search based on grid search. The author suggests supplementing detector modules and trying other algorithms as well.

## 5.4 Runtime Measurements

How fast the detector modules run, is documented in Table 5.5. The detector was implemented in Python 3.8 on a MacBook Pro (Retina 13-inch, Early 2015), equipped with a 2.9 GHz dual-core Intel Core i5 central processing unit (CPU), 8 GB of RAM, and an Ubuntu 20.04 operating system.

**Table 5.5:** A9-Dataset test set evaluation regarding the runtime of detector modules, an evaluation based on a dual-core Intel Core i5 CPU. The figures in parentheses refer to the batch R1\_S5 that contains invalid points in the origin.

Module	Ø Runtime in ms	Ø Frequency in Hz
Region of Interest Selection	4.05 (28.46)	
Ground Segmentation	0.83 (1.81)	
Background Filtering	1.05 (1.06)	
Outlier Removal	4.82 (2.80)	
Clustering	8.00 (3.59)	
Bounding Box Fitting Filtering	2.15 (2.34)	
Classification	0.18 (0.16)	
Total	21.08 (40.22)	47.44 (24.86)

Table 5.5 proves that the emphasized runtime efficiency is taken into account. The detector processes big data point clouds rapidly and achieves a processing speed of 47.44 Hz, which is nearly five times faster than the 10 Hz scan rate of the Ouster LiDAR sensors. This high speed allows time for more computing-intensive algorithms as well as embedding the detector in a system that publishes or streams detections over the internet for vehicle-to-infrastructure (V2I) communication.

The table additionally demonstrates how much performance is gained if the detector does not input the high number of invalid points in the origin. 80,442 additional points have to be filtered out in the first processing step through the region of interest selection, a processing step lasting seven times longer.

## 5.5 Mean Average Precision through Intersection over Union in 3D

The IoU in 3D states how well the 3D bounding box of an object, predicted by the detector, overlaps with the ground truth 3D bounding box. The overlap  $[0; 1]$  determines whether the prediction counts as TP, FP, or FN, in addition to the query whether the classes match. The overlap needs to exceed a threshold to be considered sufficient, a threshold that can be chosen differently for classes and that is often set to the value 0.3, 0.5, or 0.7. The results in Table 5.6 use as threshold 0.7 for vehicles and unclassified objects (other), and 0.5 for pedestrians and bicycles, which matches the thresholds in the widely used KITTI Vision Benchmark Suite [31].

As the detector of this study classifies objects into five classes ({1} car, {2} bus, {3} pedestrian, {4} bicycle, and {5} other) instead of the ten classes ({1} car, {2} truck, {3} trailer, {4} van, {5} motorcycle, {6} bus, {7} pedestrian, {8} bicycle, {9} emergency vehicle, and {10} other) labeled in the A9-Dataset, classes are remapped.

- Car: combines (1) car, (4) van, and (9) emergency vehicle
- Bus: combines (2) truck, (3) trailer, and (6) bus
- Pedestrian: remains (7) pedestrian
- Bicycle: combines (5) motorcycle and (8) bicycle
- Other: remains (10) other

This remap also affects the count of how many times an object class appears in total in the ground truth of the test set and in the predictions, respectively. Also, note that for the following evaluations, ground truth objects are included only if their associated 3D bounding box encloses at least 5 LiDAR points.

**Table 5.6:** A9-Dataset test set evaluation regarding mAP. AP@0.7 (IoU threshold is 0.7) applies to the classes: car, bus, and other; AP@0.5 applies to the classes: pedestrian and bicycle. Classes are remapped and ground truth objects with more than 5 LiDAR points are included. Occurrence refers to how often a class appears in total in the predictions and in the ground truth.

Class	AP	Occurrence	
		Prediction	Ground Truth
Car	0.04	457	3287
Bus	0.00	338	1034
Pedestrian	0.00	363	340
Bicycle	0.22	439	152
Other	0.00	454	10
mAP	0.05	2051	4823

As the detection precision in Table 5.6 is insufficient, an analysis is needed on how to improve these results. One indication are insufficient intersections between the predicted and the ground truth bounding boxes, since vehicle contours are only captured from one sensor in this study and are likely to be incomplete. Following this finding, Table 5.7 experiments with a threshold lowered to 0.1 for all classes.

**Table 5.7:** A9-Dataset test set evaluation regarding mAP. AP@0.1 (IoU threshold is 0.1) applies to all classes. Classes are remapped and ground truth objects with more than 5 LiDAR points are included. Occurrence refers to how often a class appears in total in the predictions and in the ground truth.

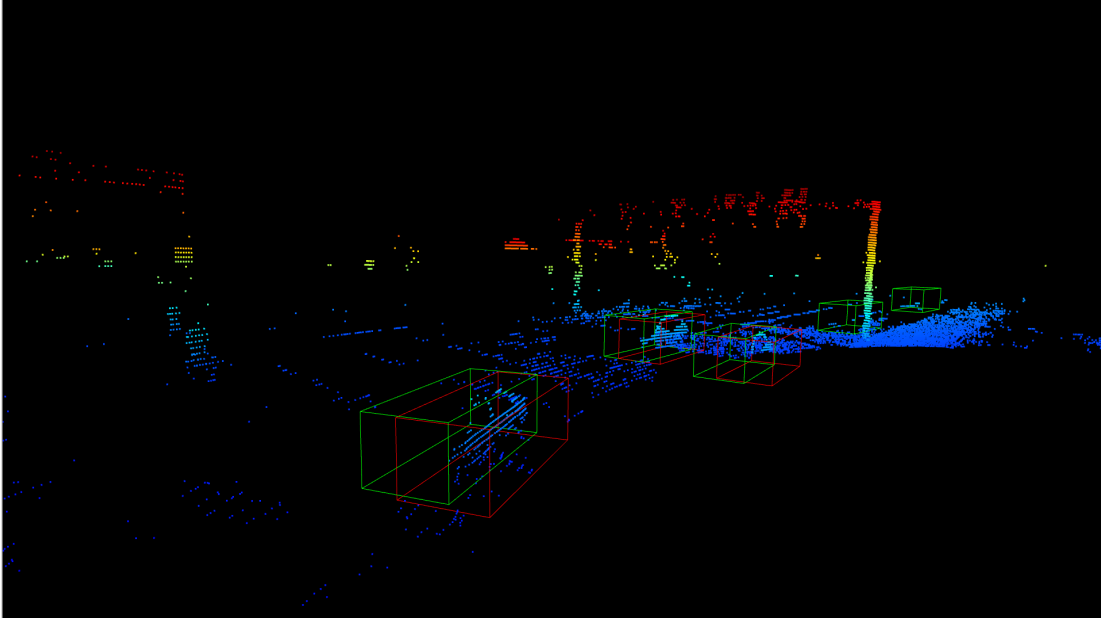
Class	AP	Occurrence	
		Prediction	Ground Truth
Car	0.97	457	3287
Bus	0.40	338	1034
Pedestrian	0.02	363	340
Bicycle	0.61	439	152
Other	0.00	454	10
mAP	0.40	2051	4823

The lower IoU threshold yields a significantly higher mAP of 0.40 and an outstanding AP of 0.97 for cars, meaning there are almost no FP car detections. This experiment signifies that a higher IoU must be ensured to enhance the result in Table 5.6, which is based on common thresholds. A higher IoU or overlap between the predictions and the ground truth can be attained if the vehicle contours are completely displayed in the point cloud, achievable by point generation or by using multiple LiDAR sensors. Suggestions for improvement like these are discussed in more detail in Chapter 6.

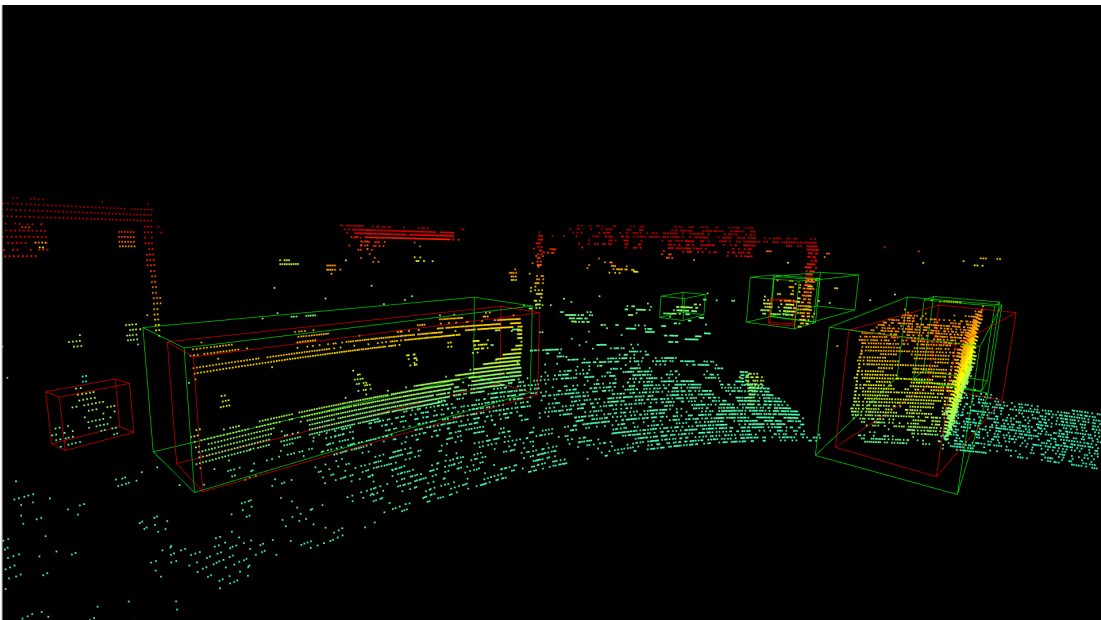
Public roadside LiDAR datasets have been published only recently, just like the A9-Dataset. For this reason, this work does not have comparative values obtained on the same data and with the same thresholds. This general deficiency in roadside LiDAR object detection will be overcome as soon as prevalent benchmarks like the KITTI Vision Benchmark Suite [31] become established.

## 5.6 Visual Results

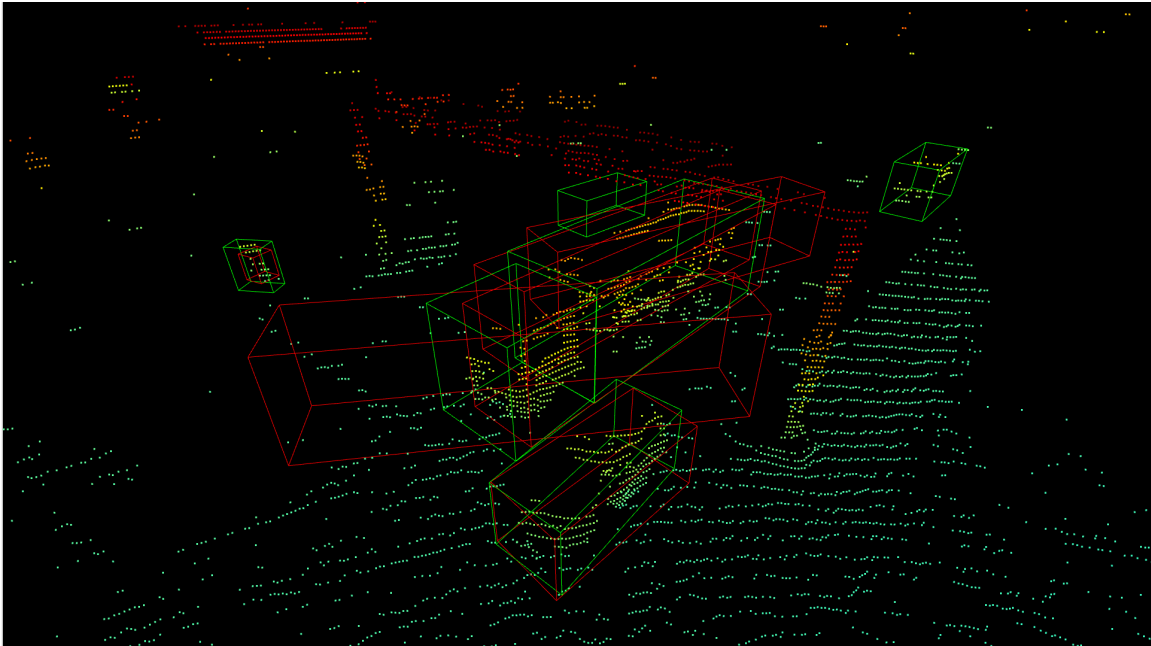
This section presents and analyzes qualitative, visual results directly in the image captions. Detected objects are highlighted via a red coloration of the 3D bounding box, while the ground truth 3D bounding boxes are colored green.



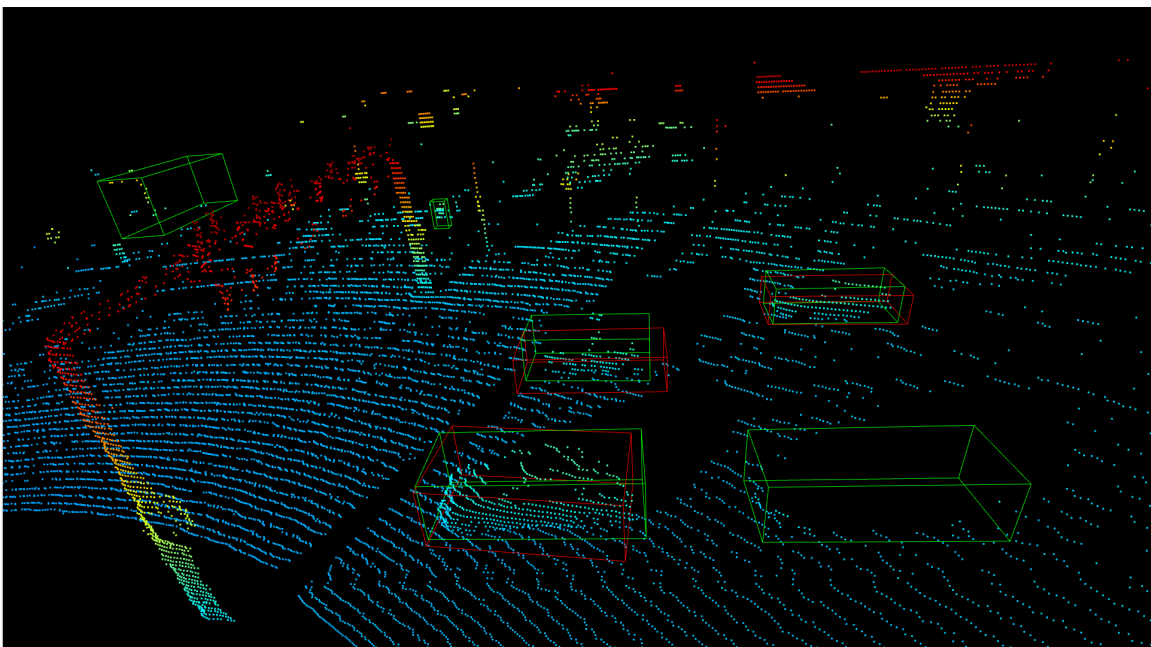
**Figure 5.2:** Detected cars in the foreground whose predicted bounding boxes are inaccurately fitted. This is due to the incomplete capture of the cars by a single LiDAR sensor, which only scans the vehicle contour facing it. The bounding box is centered at this scanned contour, making it appear displaced. Two missed FN objects in the background. (Ouster north at timestamp 1651673153.662048363)



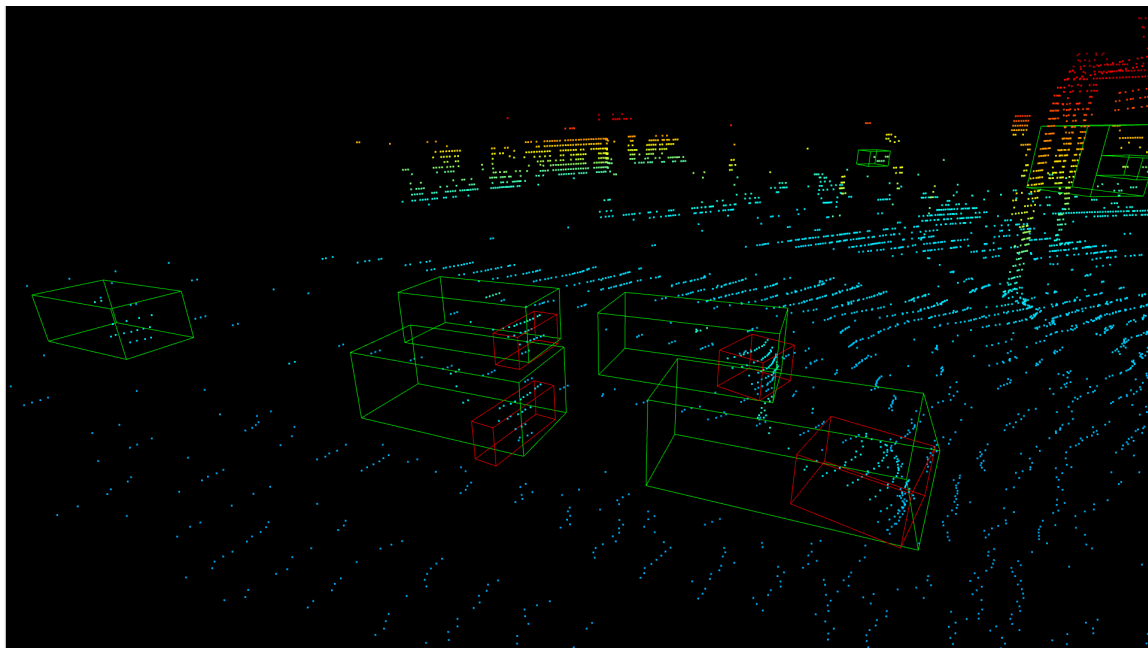
**Figure 5.3:** A detected bus and truck, where the truck is also classified as a bus due to remapping. Predicted FP objects on the left and a missed FN object in the background, next to an incompletely detected vehicle. (Ouster south at timestamp 1651673051.455442782)



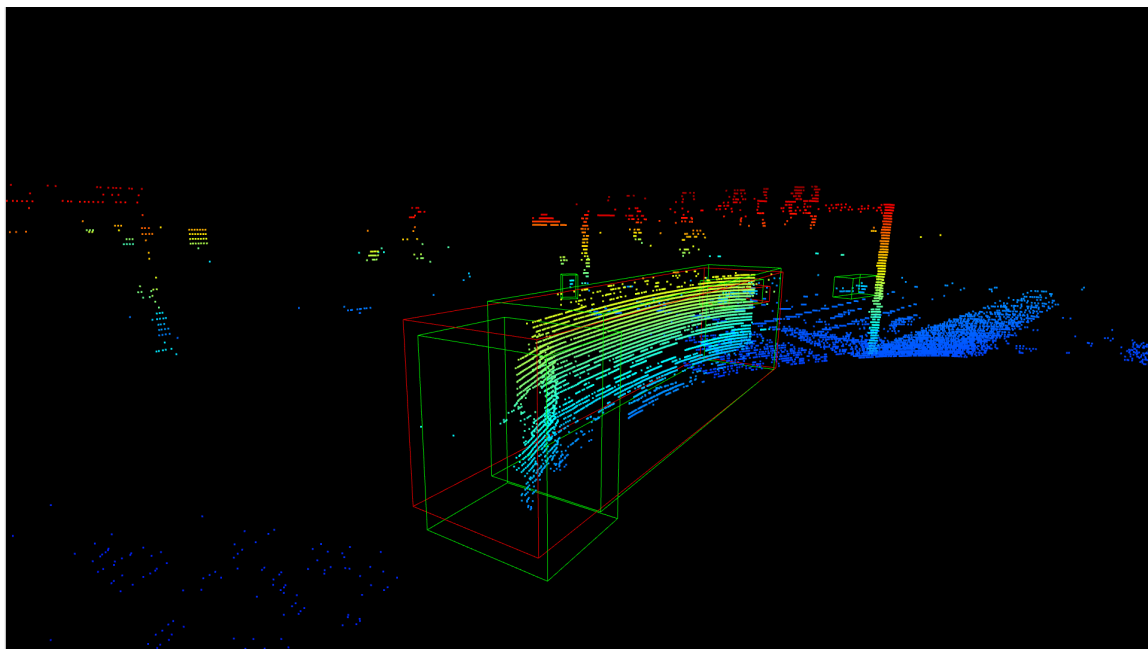
**Figure 5.4:** A truck divided into separate point clusters resulting from the clustering parameters that prioritize the separation of road users in congested traffic, while accepting that a single road user might be divided into multiple clusters. Refer to Chapter 6 for a proposal on how this problem can be fixed. Each of the separate point clusters are classified as bus due to the noticeable height of the object. Since the separate point clusters are smaller than the determined thresholds, the dimensions of the fitted bounding boxes are expanded to standard dimensions. (Ouster south at timestamp 1651673072.052599059)



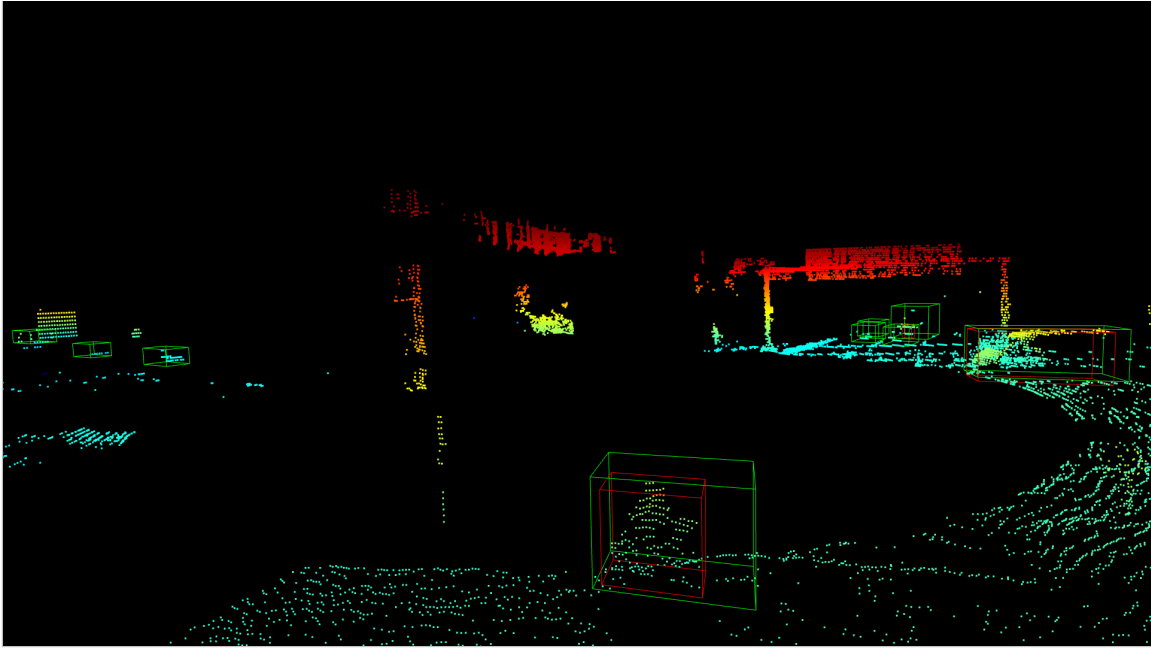
**Figure 5.5:** Vehicles closer to the LiDAR sensor are more likely to be captured in full length as opposed to the vehicles in Figure 5.6. Missed FN objects on the right and in the background. (Ouster north at timestamp 1646667315.546447382)



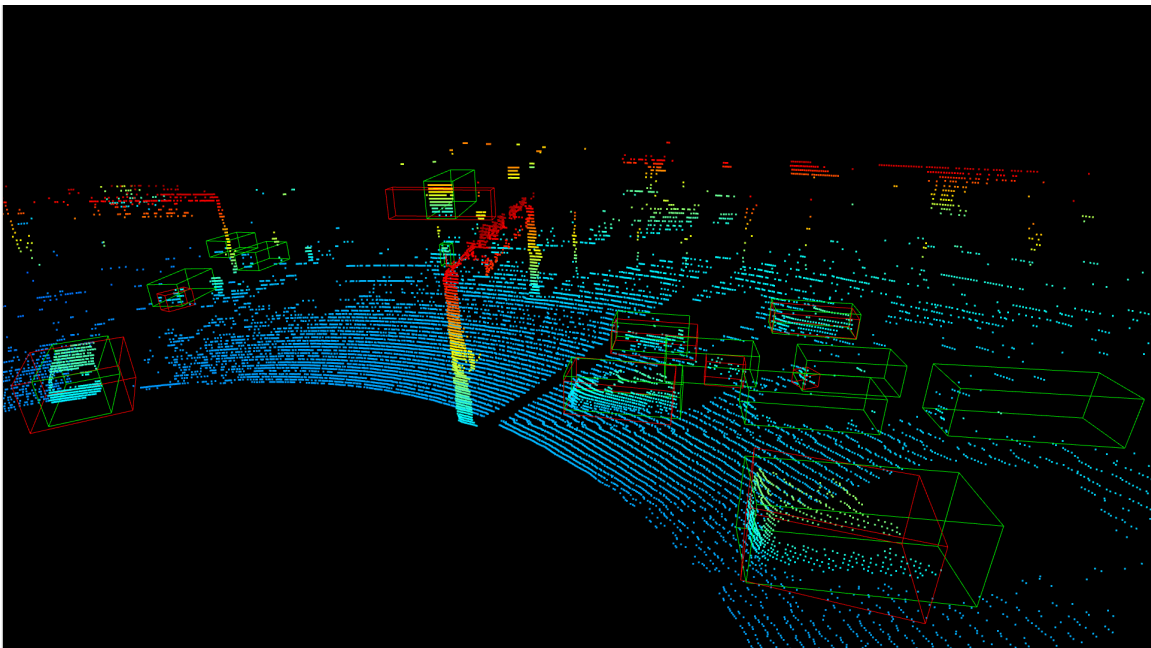
**Figure 5.6:** Vehicles in farther distance to the LiDAR sensor are less likely to be captured in full length as opposed to the vehicles in Figure 5.5. A missed FN object on the left. (Ouster south at timestamp 1646667312.952981510)



**Figure 5.7:** A truck-trailer combination is labeled separately as truck and trailer in the A9-Dataset (Note the two green bounding boxes around the vehicle in the front). Both are recorded at such close distance to each other that the detector regards them as one object. (Ouster north at timestamp 1651673166.359945107)



**Figure 5.8:** A detected bicycle in the foreground. A detected vehicle on the right. The vehicles on the left in the background are not detected because they are behind the S110 gantry bridge and this area is removed from the point cloud when selecting the region of interest with Equation 4.1. (Ouster south at timestamp 1651673088.655928807)



**Figure 5.9:** Several correctly detected road users, but also a few missed FN objects. The truck in the background is only scanned from the rear, resulting in a perceived vehicle width greater than the perceived vehicle length. Since fitting a bounding box using principal component analysis (PCA) determines the longest axis or principal component and sets its size as the vehicle length, the vehicle length (actually being the vehicle width) is shorter than the threshold and is then expanded to default length of a bus with 12 m (the detector does not have a separate truck class and classifies sufficiently high objects as a bus). (Ouster north at timestamp 1646667338.039713163)





# Chapter 6

## Future Work

Since this study is the first one within the Providentia++ project [50] that researches unsupervised 3D object detection, many directions are identified on how to enhance the developed detector in follow-up work. Promising directions of research are identified through the extensive literature research and the experience gained when experimenting with the developed detector. Since the detector is modular, individual software modules can be substituted and assessed quickly, making it a good foundation for future research.

### Object Detection Performance

Object detection is the pivotal task for 3D scene understanding [71][112] and in autonomous vehicles, object detection affects decision making and path planning. The importance of the computer vision task comes along with a high level of complexity, which is far from being mastered by the scientific community. In 2022, Bai et al. conclude in their survey [3] that there is additionally an evident performance gap between object detection with mobile, on-board LiDAR and object detection with static roadside LiDAR to the detriment of the latter. In line with this observation, the achieved detection results as part of the outcome of this study are improvable. To achieve more precise object detection, this chapter also states concrete guidance.

### Object Classification

Assigning sparse object clusters to detailed classes is challenging without additional information. Tracking objects can provide additional information as it estimates an object's velocity. Knowing how fast an object moves facilitates distinguishing between bicyclists and motorcyclists because a bicyclist typically drives slower than a motorcyclist on rural roads or when crossing an intersection. The same assumption could apply to trucks and cars, since cars often travel faster than trucks.

The precision with which the detector classifies objects can also be enhanced by adding another sensor modality. Cameras are suitable because of their dense scene recording and ability to capture colors, so the sensor fusion of camera and LiDAR is suggested.

Other helpful additional information originates by using a high-definition (HD) map because objects localized on such a map enables detectors to classify more accurately. The majority of object classes such as cars, vans, buses, trucks, and other vehicles are normally not located on sidewalks. In turn, pedestrians are usually located on sidewalks and crosswalks, so knowing on which road segment an object is narrows down the number of possible classes. This bias, on the other hand, excludes rare deviating and potentially dangerous situations; for example a person with a broken car on the highway or a child jumping into the road.

Apart from feeding additional information, localized objects have more exploitable features than their object dimensions or number of points that may allow a clearer distinction between object classes. For instance, Zhao et al. [136] examine object-specific features: vehicles are identified in point clouds if they portray the two perpendicular lines of the L-shaped vehicle contour; bicycles (one-axis objects) and pedestrians (zero-axis objects) are identified if they have only one axis and no axis, respectively. Ultimately, classifying objects could be accomplished more precisely by choosing another approach. Other researchers developed usable detectors based on random forests and trained on a set of labeled data samples [13][111].

### **Object Localization**

HD maps are useful for localizing objects as well. The detector can correct an object's position when it discovers, for example, that the current position of an identified car is on a center line according to the HD map. To discover that, the LiDAR sensor must be calibrated on the global positioning system (GPS) and thus be in line with the HD map.

Alternatively, an object's position can be corrected when detecting the lane markings in the point cloud. This idea is investigated in studies under the name lane identification [20][109] and can be achieved through the special clustering algorithm MCDSCAN [109]. There is even a benchmark for lane identification [14].

### **Object Orientation**

Other approaches could ensure a more precise orientation of the objects around the z-axis than the approach used here. Estimating a more precise orientation of sparse object clusters is achievable with an HD map as well because it stores the imposed direction of travel as well as the cardinal direction at any point on the map.

A more generic approach is to improve orientation by means of a different algorithm. Instead of considering the principal components of the convex hull, the author suggests an algorithm based on RANSAC as targeted by Oniga and Nedeveschi [69], Fu et al. [27], and Zhao et al. [135].

### **Region of Interest**

Apart from object classification and object localization, HD maps are suitable to extract a more accurate region of interest than with the presented method. With an HD map, the road can be cropped to centimeter accuracy, which means that more of the background can be removed already in the first processing step.

### **Detecting Far Objects**

The more distant roadside sensor entails a larger field of view and its elevated perspective solves occlusion, from which on-board sensors suffer. While the larger field of view of a more distant sensor facilitates perceiving an entire intersection, LiDAR sensors capture objects with fewer points. Sparse point cloud data portrays objects like cars inaccurately in the scene and complicates object detection. This issue is more severe for small road users such as pedestrians and worsens the farther away from the sensor objects are located. Detecting far objects is a common limitation of developed methods in the research area [34] and also recognized with the presented detector.

Extending the detection range is investigated by Liu et al. [55] from whose publication ideas can be adopted. Another solution is augmenting the number of points by which objects are depicted artificially with point generation, shape completion, or object reconstruction algorithms [115]. Instead of augmenting points artificially, merging synchronized point clouds from different LiDAR sensors portrays objects more accurately as well [105].

### Occlusion and Partial Sensing

Aside from extending the detection range, deploying multiple LiDAR sensors to enable merging point clouds also alleviates occlusion and sensing objects partially, which still occurs when relying on a single sensor although not as bad as with on-board sensing. Road users are occluded by others closer to the sensor or by infrastructure such as large road signs. Road users are also perceived incompletely when they are at the lower edge of the roadside sensor's vertical field of view. When they are directly below the sensor, they are not perceived at all. In addition to occlusion, objects are partially sensed because LiDAR only perceives the object contours facing the sensor.

To address occlusion and partial sensing, using multiple LiDAR sensors is one of the main suggestions of this chapter and they are best installed at different corners if it is an intersection. In the Providentia++ project, the gantry bridge opposite the S110 sensor station is a good choice. Scanning road users from all sides ensures recording the object dimensions correctly.

### Congested Traffic

Congested traffic poses two challenges for unsupervised object detection in roadside LiDAR data [86]. First, popular background filtering algorithms such as 3D-DSF forfeit precision due to foreground regions containing similar number of points than background regions in dense traffic. Under these circumstances, it is difficult to find a threshold to distinguish foreground and background voxels, which is the working principle of 3D-DSF and other volumetric-based algorithms. This difficulty thought further may lead to an algorithm filtering too many foreground points from a target point cloud.

This study does not employ 3D-DSF to filter the background, but rather a more recent rule-based algorithm. Nevertheless, even better algorithms for extracting the foreground might exist or evolve. The author suggests experiments with point-based background filtering. Another background filtering algorithm might also supersede the need to remove the pillars of the gantry bridges with predefined thresholds, which is not a generic solution.

Besides background filtering, unsupervised object detection in roadside LiDAR data is based on clustering algorithms [137][3][63][135][55][99][22][133][126][101]. While clustering, it is more challenging to accurately separate points of different road users and uniting points belonging together when road users are close to each other. Situations when road users are close to each other occur more frequently in congested traffic.

This study prioritizes separation of road users, accepting that a single road user might be divided into multiple clusters. Subdivided road users could be corrected once object detection is extended by object tracking since a tracking algorithm can recognize objects in consecutive frames and monitor whether near clusters move with the same speed in the same direction.

Testing further algorithms may solve clustering issues as well. A clustering algorithm taking the decreasing point density over distance into account would also help to detect far objects. Algorithms more suitable for this particular application, as presented in recent years, may find their way into software libraries or users could implement them efficiently themselves in a C++ programmed detector. Possible options are hierarchical maximum density clustering of application with noise (HMDCAN), presented by Zheng et al. [139], or the adaptive two-stage clustering algorithm from Zhang et al. [120].

### Adaption to Changing Weather

Adapted background filtering and clustering algorithms also help to cope with changing weather as scientists ascertain declining detection performance under adverse weather conditions. Research arisen from this problem proposes ground surface-enhanced density statistic filtering (GS-DSF) [107] or clustering algorithms such as 3D-SDBSCAN [110] and voxelization-

based clustering [107].

### **Updating the Background Model**

Even the background that roadside sensors perceive can change, considering a parking vehicle that drives on or people that sit down on a bench. Moreover, a sensor can decalibrate when employed over a long time [50]. To adapt to a changed background, an automatism is needed that regularly updates the background model.

### **Runtime**

The runtime of the detector can be further improved by deploying the tensors of Open3D [140], which will soon be fully supported. Tensors are data containers that enable to faster process data on the graphics processing unit (GPU).

# Chapter 7

## Conclusion

The proposed unsupervised detector is an effective software solution to tackle 3D object detection using roadside LiDAR. The detector is the first within the Providentia++ project [50] to adopt an approach that is not based on deep learning and that does not require large-scale sample data to be deployable in a new domain. Its modular architecture qualifies the detector to serve as code base for follow-up work.

The current solution demonstrates potential as it achieves an average precision (AP) of 0.61 regarding the extraction of the foreground from raw point clouds. Regarding the challenging object detection task, the detector obtains a mean average precision (mAP) of 0.40 if thresholds are reduced to a level where the sparsely and incompletely captured road users in the A9-Dataset [19] are taken into account. This adaption will be no longer necessitated once the urban intersection in Munich, Germany is recorded from opposing perspectives.

Aside from precision, the object detection runs fast even on a standard central processing unit (CPU), where point clouds are processed at a frequency of 47.44 Hz. This high speed is five times faster than the 10 Hz scan rate of the employed LiDAR sensors, which makes the detector real-time capable and allows time for more computing-intensive operations. Processing speed is key when the software is deployed in intelligent infrastructure as the embedding adds considerable computational load with respect to streaming and publishing detected obstacles via vehicle-to-infrastructure (V2I) communication.

How the software can be advanced is outlined based on the extensive literature review that this work conducts. In the literature, many unsupervised processing approaches of point clouds are presented. However, it is hardly possible to compare the results between these approaches since the available roadside datasets are not yet used on a large scale. It is often not possible to test other approaches either, since the researchers in the field rarely include source code in their publications. This study aims to set an example by using only publicly available data and by sharing source code.

3D object detection in traffic must meet strict approval criteria to prove its reliability. Traditional unsupervised algorithms are inherently explainable, which facilitates a certified deployment required when the technology is rolled out broadly for safer traffic.

In fact, roadside sensors are a suitable measure to increase traffic safety, as they complement the environment perceived by connected autonomous vehicles and drivers of conventional vehicles. Roadside sensors capture the environment from a beneficial elevated perspective and overcome the limited visibility at intersections in congested traffic; they even capture what is occurring around corners. In addition to a complete field of view at intersections, roadside sensors enlarge the perception range in transportation to an extent limited only by the distribution of intelligent infrastructure systems. Enlarging the perception range enables vehicles and drivers to be warned of hazards long before they can perceive them themselves, for instance the hazard emanating from a broken-down vehicle in the lane 30 seconds ahead.

In addition to safety, intelligent infrastructure allows to analyze and optimize traffic flow, which decreases travel times and reduces emissions. Most importantly, this infrastructure will advance autonomous transportation, as collaboration between roadside sensors and on-board sensors is more effective and economic than equipping every vehicle with the entire sensor selection of cameras, radar, and LiDAR [3]. When these sensors are attached to infrastructure and data is transmitted to all road users via the latest wireless communication channels, society may benefit from comfortable travel with self-driving vehicles sooner.

# Appendix A

## Appendix 1

**Table A.1:** A9-Dataset test set evaluation regarding mAP. AP@0.1 (IoU threshold is 0.1) applies to all classes. Classes are remapped and ground truth objects with more than 5 LiDAR points are included. Occurrence refers to how often a class appears in total in the predictions and in the ground truth. No default object sizes are applied after classification.

Class	AP	Occurrence	
		Prediction	Ground Truth
Car	0.88	457	3287
Bus	0.28	338	1034
Pedestrian	0.02	363	340
Bicycle	0.34	439	152
Other	0.00	454	10
mAP	0.31	2051	4823





# List of Figures

2.1	Stacked laser channels in a 3D LiDAR sensor scanning at different vertical angles [102]. . . . .	5
2.2	A laser beam being emitted, reflected, and received [102]. . . . .	6
2.3	Four formats to represent 3D data [140]. . . . .	9
2.4	Distinction between three computer vision task in a point cloud using 2D bounding boxes [140]. . . . .	10
2.5	Equipped gantry bridge near Munich, Germany as part of the Providentia++ project. . . . .	15
2.6	Precision-recall curve $p(r)$ based on [1]. . . . .	19
3.1	Polar 2D grid from a bird's eye view [43]. . . . .	27
4.1	Illustration of the proposed processing chain for unsupervised 3D object detection. Part of the intelligent infrastructure system is a roadside LiDAR sensor that generates raw point clouds, which are processed by a local computing unit in seven algorithmic steps. The black points depict the points removed in a step. Information about the detected objects is transmitted wirelessly to affected autonomous vehicles so that they can maneuver safely in traffic. . . .	49
4.2	Removed gantry bridges from LiDAR point cloud ( <b>excluded points</b> , <b>axis aligned bounding boxes</b> ). . . . .	51
4.3	Removed sensor station from LiDAR point cloud ( <b>excluded points</b> , <b>axis aligned bounding boxes</b> ). . . . .	51
4.4	Removed building blocks from LiDAR point cloud ( <b>excluded points</b> , <b>oriented bounding boxes</b> ). . . . .	52
4.5	Three adjacent laser channels each pointing in the same two horizontal directions [55]. . . . .	53
4.6	Two adjacent laser beams from the same laser channel capturing different distances. The two laser beams are apart according to the horizontal resolution $\alpha$ [55]. . . . .	53
4.7	A building wall representing background (B) is more distant to the LiDAR sensor than a passing road user (A) [55]. . . . .	53
4.8	Distance threshold determined by the triangle algorithm maximizing orthogonal lines on the drawn diagonal line [126]. . . . .	55
4.9	Unmodified oriented bounding boxes from Open3D are rolled and tilted around localized vehicles. . . . .	57
4.10	Combining information from an axis aligned bounding box with the yaw angle from an oriented bounding box leads to a bounding box that is only rotated around the z-axis. . . . .	57
4.11	Object from which only the upper part is captured. . . . .	58
4.12	Corrected incomplete scan of an object by fixing the bottom side of the bounding box to the road. . . . .	58

5.1	Average number of invalid points $p_i = (0, 0, 0) \in R^3$ in the origin in a point cloud due to missed laser beam reflections. This number per laser channel is divided into eight laser channel regions, each representing eight of the total 64 channels. . . . .	63
5.2	Detected cars in the foreground whose predicted bounding boxes are inaccurately fitted. This is due to the incomplete capture of the cars by a single LiDAR sensor, which only scans the vehicle contour facing it. The bounding box is centered at this scanned contour, making it appear displaced. Two missed FN objects in the background. (Ouster north at timestamp 1651673153.662048363)	68
5.3	A detected bus and truck, where the truck is also classified as a bus due to remapping. Predicted FP objects on the left and a missed FN object in the background, next to an incompletely detected vehicle. (Ouster south at timestamp 1651673051.455442782) . . . . .	68
5.4	A truck divided into separate point clusters resulting from the clustering parameters that prioritize the separation of road users in congested traffic, while accepting that a single road user might be divided into multiple clusters. Refer to Chapter 6 for a proposal on how this problem can be fixed. Each of the separate point clusters are classified as bus due to the noticeable height of the object. Since the separate point clusters are smaller than the determined thresholds, the dimensions of the fitted bounding boxes are expanded to standard dimensions. (Ouster south at timestamp 1651673072.052599059)	69
5.5	Vehicles closer to the LiDAR sensor are more likely to be captured in full length as opposed to the vehicles in Figure 5.6. Missed FN objects on the right and in the background. (Ouster north at timestamp 1646667315.546447382) . . . . .	69
5.6	Vehicles in farther distance to the LiDAR sensor are less likely to be captured in full length as opposed to the vehicles in Figure 5.5. A missed FN object on the left. (Ouster south at timestamp 1646667312.952981510) . . . . .	70
5.7	A truck-trailer combination is labeled separately as truck and trailer in the A9-Dataset (Note the two green bounding boxes around the vehicle in the front). Both are recorded at such close distance to each other that the detector regards them as one object. (Ouster north at timestamp 1651673166.359945107) . . .	70
5.8	A detected bicycle in the foreground. A detected vehicle on the right. The vehicles on the left in the background are not detected because they are behind the S110 gantry bridge and this area is removed from the point cloud when selecting the region of interest with Equation 4.1. (Ouster south at timestamp 1651673088.655928807) . . . . .	71
5.9	Several correctly detected road users, but also a few missed FN objects. The truck in the background is only scanned from the rear, resulting in a perceived vehicle width greater than the perceived vehicle length. Since fitting a bounding box using principal component analysis (PCA) determines the longest axis or principal component and sets its size as the vehicle length, the vehicle length (actually being the vehicle width) is shorter than the threshold and is then expanded to default length of a bus with 12m (the detector does not have a separate truck class and classifies sufficiently high objects as a bus). (Ouster north at timestamp 1646667338.039713163) . . . . .	71

## List of Tables

2.1	Performance matrix for different perception sensors [55]. . . . .	8
2.2	LiDAR sensor specification in the Providentia++ project [19]. . . . .	15
2.3	Real-world datasets with point clouds from stationary 3D LiDAR sensors. . . . .	16
2.4	Precision and recall for varying confidence thresholds based on [1]. . . . .	19
5.1	A9-Dataset test set evaluation regarding the reduction of the points of a point cloud through the different detector modules. Clustering based on DBSCAN finds outliers differently than the employed outlier removal algorithm and reduces the number of points additionally. . . . .	64
5.2	A9-Dataset test set evaluation of ground segmentation algorithms. Bold and underline denote the best and second best results, respectively. . . . .	64
5.3	A9-Dataset test set ablation study of detector modules, a study in which individual modules are alternately deactivated to extract the foreground in a point cloud. The abbreviated module names stand for RIS = region of interest selection, GS = ground segmentation, BF = background filtering, OR = outlier removal, and C = clustering. Bold and underline denote the best and second best results, respectively. . . . .	64
5.4	A9-Dataset test set ablation study of foreground extraction in a point cloud, a study in which different parameters of the adjustable algorithms are tested. $d_{max}$ is the maximal Euclidean distance up to which a point is considered part of the precomputed plane model. $n_{adj}$ is the number of adjacent points within the search radius $r_{search}$ required for a point to not be removed as outlier. $n_{min}$ is the minimum required number to form a point cluster and $r_{search}$ defines the maximum search radius. Bold and underline denote the best and second best results, respectively. . . . .	65
5.5	A9-Dataset test set evaluation regarding the runtime of detector modules, an evaluation based on a dual-core Intel Core i5 CPU. The figures in parentheses refer to the batch R1_S5 that contains invalid points in the origin. . . . .	66
5.6	A9-Dataset test set evaluation regarding mAP. AP@0.7 (IoU threshold is 0.7) applies to the classes: car, bus, and other; AP@0.5 applies to the classes: pedestrian and bicycle. Classes are remapped and ground truth objects with more than 5 LiDAR points are included. Occurrence refers to how often a class appears in total in the predictions and in the ground truth. . . . .	67
5.7	A9-Dataset test set evaluation regarding mAP. AP@0.1 (IoU threshold is 0.1) applies to all classes. Classes are remapped and ground truth objects with more than 5 LiDAR points are included. Occurrence refers to how often a class appears in total in the predictions and in the ground truth. . . . .	67

- A.1 A9-Dataset test set evaluation regarding mAP. AP@0.1 (IoU threshold is 0.1) applies to all classes. Classes are remapped and ground truth objects with more than 5 LiDAR points are included. Occurrence refers to how often a class appears in total in the predictions and in the ground truth. No default object sizes are applied after classification. . . . . 79

## Bibliography

- [1] Anwar, A. “What is Average Precision in Object Detection & Localization Algorithms and how to calculate it?” In: *Towards Data Science* (May 13, 2022). URL: <https://towardsdatascience.com/what-is-average-precision-in-object-detection-localization-algorithms-and-how-to-calculate-it-3f330efe697b> (visited on 08/31/2022).
- [2] Arnon, D. S. and Gieselmann, J. P. “A Linear Time Algorithm for the Minimum Area Rectangle Enclosing a Convex Polygon”. In: 1983.
- [3] Bai, Z., Wu, G., Qi, X., Liu, Y., Oguchi, K., and Barth, M. J. “Infrastructure-Based Object Detection and Tracking for Cooperative Driving Automation: A Survey”. In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. 2022, pp. 1366–1373. DOI: 10.1109/IV51971.2022.9827461.
- [4] Balta, H., Velagic, J., Bosschaerts, W., De Cubber, G., and Siciliano, B. “Fast Statistical Outlier Removal Based Method for Large 3D Point Clouds of Outdoor Environments”. In: *IFAC-PapersOnLine* 51.22 (2018). 12th IFAC Symposium on Robot Control SY-ROCO 2018, pp. 348–353. ISSN: 2405-8963. DOI: <https://doi.org/10.1016/j.ifacol.2018.11.566>. URL: <https://www.sciencedirect.com/science/article/pii/S2405896318332725>.
- [5] BM, N. “What is an encoder decoder model?” In: *Towards Data Science* (Oct. 7, 2020). URL: <https://towardsdatascience.com/what-is-an-encoder-decoder-model-86b3d57c5e1a> (visited on 09/10/2022).
- [6] Börcs, A., Nagy, B., Baticz, M., and Benedek, C. “A Model-Based Approach for Fast Vehicle Detection in Continuously Streamed Urban LIDAR Point Clouds”. In: *Computer Vision - ACCV 2014 Workshops*. Ed. by Jawahar, C. and Shan, S. Cham: Springer International Publishing, 2015, pp. 413–425. ISBN: 978-3-319-16628-5.
- [7] Boser, B. E., Guyon, I. M., and Vapnik, V. N. “A Training Algorithm for Optimal Margin Classifiers”. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 1992, pp. 144–152. ISBN: 089791497X. DOI: 10.1145/130385.130401. URL: <https://doi.org/10.1145/130385.130401>.
- [8] Breiman, L. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32.
- [9] Buhrmester, V., Münch, D., and Arens, M. “Analysis of Explainers of Black Box Deep Neural Networks for Computer Vision: A Survey”. In: *Machine Learning and Knowledge Extraction* 3.4 (2021), pp. 966–989. ISSN: 2504-4990. DOI: 10.3390/make304048. URL: <https://www.mdpi.com/2504-4990/3/4/48>.
- [10] Busch, S., Koetsier, C., Axmann, J., and Brenner, C. “LUMPI: The Leibniz University Multi-Perspective Intersection Dataset”. In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. 2022, pp. 1127–1134. DOI: 10.1109/IV51971.2022.9827157.

- [11] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. “nuScenes: A multimodal dataset for autonomous driving”. In: *CoRR abs/1903.11027* (2019). arXiv: 1903.11027. URL: <http://arxiv.org/abs/1903.11027>.
- [12] Cen, J., Yun, P., Cai, J., Wang, M. Y., and Liu, M. *Open-set 3D Object Detection*. 2021. DOI: 10.48550/ARXIV.2112.01135. URL: <https://arxiv.org/abs/2112.01135>.
- [13] Chen, J., Xu, H., Wu, J., Yue, R., Yuan, C., and Wang, L. “Deer Crossing Road Detection With Roadside LiDAR Sensor”. In: *IEEE Access* 7 (2019), pp. 65944–65954. DOI: 10.1109/ACCESS.2019.2916718.
- [14] Chen, L., Sima, C., Li, Y., Zheng, Z., Xu, J., Geng, X., Li, H., He, C., Shi, J., Qiao, Y., and Yan, J. *PersFormer: 3D Lane Detection via Perspective Transformer and the OpenLane Benchmark*. 2022. DOI: 10.48550/ARXIV.2203.11089. URL: <https://arxiv.org/abs/2203.11089>.
- [15] Chenghao Sun, Pengpeng Sun, Lingfeng Wan, and Xiangmo Zhao. “Background Extraction and Objects Segmentation with 3D Roadside LiDAR under Snowy Weather”. In: (2022).
- [16] Choi, S., Kim, T., and Yu, W. “Performance evaluation of RANSAC family”. In: vol. 24. Jan. 2009. DOI: 10.5244/C.23.81.
- [17] Chum, O. and Matas, J. “Matching with PROSAC - progressive sample consensus”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 1. 2005, 220–226 vol. 1. DOI: 10.1109/CVPR.2005.221.
- [18] Chum, O., Matas, J., and Kittler, J. “Locally Optimized RANSAC”. In: *DAGM-Symposium*. 2003.
- [19] Creß, C., Zimmer, W., Strand, L., Lakshminarasimhan, V., Fortkord, M., Dai, S., and Knoll, A. *A9-Dataset: Multi-Sensor Infrastructure-Based Dataset for Mobility Research*. 2022. DOI: 10.48550/ARXIV.2204.06527. URL: <https://arxiv.org/abs/2204.06527>.
- [20] Cui, Y., Xu, H., Wu, J., Sun, Y., and Zhao, J. “Automatic Vehicle Tracking With Roadside LiDAR Data for the Connected-Vehicles System”. In: *IEEE Intelligent Systems* 34.3 (2019), pp. 44–51. DOI: 10.1109/MIS.2019.2918115.
- [21] Deng, Y., Wang, D., Cao, G., Ma, B., Guan, X., Wang, Y., Liu, J., Fang, Y., and Li, J. “BAAI-VANJEE Roadside Dataset: Towards the Connected Automated Vehicle Highway technologies in Challenging Environments of China”. In: *CoRR abs/2105.14370* (2021). arXiv: 2105.14370. URL: <https://arxiv.org/abs/2105.14370>.
- [22] Du, X., Ang, M. H., Karaman, S., and Rus, D. *A General Pipeline for 3D Detection of Vehicles*. 2018. DOI: 10.48550/ARXIV.1803.00387. URL: <https://arxiv.org/abs/1803.00387>.
- [23] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. “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*. KDD’96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.
- [24] Fischler, M. A. and Bolles, R. C. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: <https://doi.org/10.1145/358669.358692>.

- [25] Fleck, T., Daaboul, K., Weber, M., Schörner, P., Wehmer, M., Doll, J., Orf, S., Sußmann, N., Hubschneider, C., Zofka, M. R., Kuhnt, F., Kohlhaas, R., Baumgart, I., Zöllner, R., and Zöllner, J. M. “Towards Large Scale Urban Traffic Reference Data: Smart Infrastructure in the Test Area Autonomous Driving Baden-Württemberg”. In: *Intelligent Autonomous Systems 15*. Ed. by Strand, M., Dillmann, R., Menegatti, E., and Ghidoni, S. Cham: Springer International Publishing, 2019, pp. 964–982. ISBN: 978-3-030-01370-7.
- [26] Freeman, H. and Shapira, R. “Determining the Minimum-Area Encasing Rectangle for an Arbitrary Closed Curve”. In: *Commun. ACM* 18.7 (July 1975), pp. 409–413. ISSN: 0001-0782. DOI: 10.1145/360881.360919. URL: <https://doi.org/10.1145/360881.360919>.
- [27] Fu, C., Dong, C., Zhang, X., and Dolan, J. M. *Low-cost LIDAR based Vehicle Pose Estimation and Tracking*. 2019. DOI: 10.48550/ARXIV.1910.01701. URL: <https://arxiv.org/abs/1910.01701>.
- [28] Gabb, M., Digel, H., Müller, T., and Henn, R.-W. “Infrastructure-supported Perception and Track-level Fusion using Edge Computing”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 1739–1745. DOI: 10.1109/IVS.2019.8813886.
- [29] Gandhi, R. “Support Vector Machine — Introduction to Machine Learning Algorithms”. In: *Towards Data Science* (June 7, 2018). URL: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a44fca47> (visited on 09/13/2022).
- [30] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. “Vision meets robotics: The KITTI dataset”. In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237. DOI: 10.1177/0278364913491297. eprint: <https://doi.org/10.1177/0278364913491297>. URL: <https://doi.org/10.1177/0278364913491297>.
- [31] Geiger, A., Lenz, P., and Urtasun, R. “Are we ready for autonomous driving? The KITTI vision benchmark suite”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 3354–3361. DOI: 10.1109/CVPR.2012.6248074.
- [32] Gkioxari, G. “Intersection Over Union of Oriented 3D Boxes: A New Algorithm”. In: (). URL: <https://pytorch3d.org/docs/iou3d> (visited on 08/31/2022).
- [33] Gong, Z., Wang, Z., Zhou, B., Liu, W., and Liu, P. “Pedestrian Detection Method Based on Roadside Light Detection and Ranging”. In: *SAE International Journal of Connected and Automated Vehicles* 4.4 (Nov. 2021), pp. 413–422. ISSN: 2574-0741. DOI: <https://doi.org/10.4271/12-04-04-0031>. URL: <https://doi.org/10.4271/12-04-04-0031>.
- [34] Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., and Bennamoun, M. *Deep Learning for 3D Point Clouds: A Survey*. 2019. DOI: 10.48550/ARXIV.1912.12033. URL: <https://arxiv.org/abs/1912.12033>.
- [35] Hantén, R., Kuhlmann, P., Otte, S., and Zell, A. “Robust Real-Time 3D Person Detection for Indoor and Outdoor Applications”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 2000–2006. DOI: 10.1109/ICRA.2018.8461257.
- [36] He, C., Li, R., Li, S., and Zhang, L. *Voxel Set Transformer: A Set-to-Set Approach to 3D Object Detection from Point Clouds*. 2022. DOI: 10.48550/ARXIV.2203.10314. URL: <https://arxiv.org/abs/2203.10314>.
- [37] Hermosilla, P., Ritschel, T., and Ropinski, T. *Total Denoising: Unsupervised Learning of 3D Point Cloud Cleaning*. 2019. DOI: 10.48550/ARXIV.1904.07615. URL: <https://arxiv.org/abs/1904.07615>.

- [38] Himmelsbach, M., Hundelshausen, F. v., and Wuensche, H.-J. “Fast segmentation of 3D point clouds for ground vehicles”. In: *2010 IEEE Intelligent Vehicles Symposium*. 2010, pp. 560–565. DOI: 10.1109/IVS.2010.5548059.
- [39] Himmelsbach, M. and Wuensche, H.-J. “Tracking and classification of arbitrary objects with bottom-up/top-down detection”. In: *2012 IEEE Intelligent Vehicles Symposium*. 2012, pp. 577–582. DOI: 10.1109/IVS.2012.6232181.
- [40] Horváth, E., Pozna, C., and Unger, M. “Real-Time LIDAR-Based Urban Road and Sidewalk Detection for Autonomous Vehicles”. In: *Sensors* 22.1 (2022). ISSN: 1424-8220. DOI: 10.3390/s22010194. URL: <https://www.mdpi.com/1424-8220/22/1/194>.
- [41] Hu, J. S. K., Kuai, T., and Waslander, S. L. *Point Density-Aware Voxels for LiDAR 3D Object Detection*. 2022. DOI: 10.48550/ARXIV.2203.05662. URL: <https://arxiv.org/abs/2203.05662>.
- [42] Hung, W.-C., Kretschmar, H., Casser, V., Hwang, J.-J., and Anguelov, D. *LET-3D-AP: Longitudinal Error Tolerant 3D Average Precision for Camera-Only 3D Detection*. 2022. DOI: 10.48550/ARXIV.2206.07705. URL: <https://arxiv.org/abs/2206.07705>.
- [43] Jin, X., Yang, H., Liao, X., Yan, Z., Wang, Q., Li, Z., and Wang, Z. “A Robust Gaussian Process-Based LiDAR Ground Segmentation Algorithm for Autonomous Driving”. In: *Machines* 10.7 (2022). ISSN: 2075-1702. DOI: 10.3390/machines10070507. URL: <https://www.mdpi.com/2075-1702/10/7/507>.
- [44] Kampker, A., Sefati, M., Rachman, A. S. A., Kreisköther, K., and Campoy, P. “Towards Multi-Object Detection and Tracking in Urban Scenario under Uncertainties”. In: *Proceedings of the 4th International Conference on Vehicle Technology and Intelligent Transport Systems*. SCITEPRESS - Science and Technology Publications, 2018. DOI: 10.5220/0006706101560167. URL: <https://doi.org/10.5220/0006706101560167>.
- [45] Kim, D., Jo, K., Lee, M., and Sunwoo, M. “L-Shape Model Switching-Based Precise Motion Tracking of Moving Vehicles Using Laser Scanners”. In: *IEEE Transactions on Intelligent Transportation Systems* 19.2 (2018), pp. 598–612. DOI: 10.1109/TITS.2017.2771820.
- [46] Klasing, K., Wollherr, D., and Buss, M. “A clustering method for efficient segmentation of 3D laser data”. In: *2008 IEEE International Conference on Robotics and Automation*. 2008, pp. 4043–4048. DOI: 10.1109/ROBOT.2008.4543832.
- [47] Kloeker, L., Geller, C., Kloeker, A., and Eckstein, L. “High-Precision Digital Traffic Recording with Multi-LiDAR Infrastructure Sensor Setups”. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–8. DOI: 10.1109/ITSC45102.2020.9294543.
- [48] Koestler, L., Yang, N., Wang, R., and Cremers, D. *Learning Monocular 3D Vehicle Detection without 3D Bounding Box Labels*. 2020. DOI: 10.48550/ARXIV.2010.03506. URL: <https://arxiv.org/abs/2010.03506>.
- [49] Konstantinidis, K., Alirezai, M., and Grammatico, S. “Development of a Detection and Tracking of Moving Vehicles system for 2D LIDAR sensors”. In: ().
- [50] Krämer, A., Schöller, C., Gulati, D., Lakshminarasimhan, V., Kurz, F., Rosenbaum, D., Lenz, C., and Knoll, A. *Providentia – A Large-Scale Sensor System for the Assistance of Autonomous Vehicles and Its Evaluation*. 2019. DOI: 10.48550/ARXIV.1906.06789. URL: <https://arxiv.org/abs/1906.06789>.
- [51] Leng, Z., Li, S., Li, X., and Gao, B. “An Improved Fast Ground Segmentation Algorithm for 3D Point Cloud”. In: *2020 Chinese Control And Decision Conference (CCDC)*. 2020, pp. 5016–5020. DOI: 10.1109/CCDC49329.2020.9164787.



- [52] Li, D., Wong, K., Hu, Y. H., and Sayeed, A. "Detection, classification, and tracking of targets". In: *IEEE Signal Processing Magazine* 19.2 (2002), pp. 17–29. DOI: 10.1109/79.985674.
- [53] Li, L., Yang, F., Zhu, H., Li, D., Li, Y., and Tang, L. "An Improved RANSAC for 3D Point Cloud Plane Segmentation Based on Normal Distribution Transformation Cells". In: *Remote Sensing* 9.5 (2017). ISSN: 2072-4292. DOI: 10.3390/rs9050433. URL: <https://www.mdpi.com/2072-4292/9/5/433>.
- [54] Lin, C., Liu, H., Wu, D., and Gong, B. "Background Point Filtering of Low-Channel Infrastructure-Based LiDAR Data Using a Slice-Based Projection Filtering Algorithm". In: *Sensors* 20.11 (2020). ISSN: 1424-8220. DOI: 10.3390/s20113054. URL: <https://www.mdpi.com/1424-8220/20/11/3054>.
- [55] Liu, H., Lin, C., Gong, B., and Wu, D. "Extending the Detection Range for Low-Channel Roadside LiDAR by Static Background Construction". In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), pp. 1–12. DOI: 10.1109/TGRS.2022.3155634.
- [56] Liu, Z., Li, Q., Mei, S., and Huang, M. "Background Filtering and Object Detection with Roadside LiDAR Data". In: *2021 4th International Conference on Electron Device and Mechanical Engineering (ICEDME)*. 2021, pp. 296–299. DOI: 10.1109/ICEDME52809.2021.00069.
- [57] Luo, Z., Habibi, S., and Mohrenschildt, M. v. "LiDAR Based Real Time Multiple Vehicle Detection and Tracking". In: *International Journal of Computer and Information Engineering* 10.6 (2016), pp. 1125–1132. ISSN: eISSN: 1307-6892. URL: <https://publications.waset.org/vol/114>.
- [58] Lv, B., Xu, H., Wu, J., Tian, Y., and Yuan, C. "Raster-Based Background Filtering for Roadside LiDAR Data". In: *IEEE Access* 7 (2019), pp. 76779–76788. DOI: 10.1109/ACCESS.2019.2919624.
- [59] Mehrabi, P. and Taghirad, H. D. *A Gaussian Process-Based Ground Segmentation for Sloped Terrains*. 2021. DOI: 10.48550/ARXIV.2111.10638. URL: <https://arxiv.org/abs/2111.10638>.
- [60] Murphy, K. P. et al. "Naive bayes classifiers". In: *University of British Columbia* 18.60 (2006), pp. 1–8.
- [61] Myatt, D., Torr, P., Nasuto, S., Bishop, J., and Craddock, R. "NAPSAC: High Noise, High Dimensional Robust Estimation - it's in the Bag". In: Jan. 2002. DOI: 10.5244/C.16.44.
- [62] Narksri, P., Takeuchi, E., Ninomiya, Y., Morales, Y., Akai, N., and Kawaguchi, N. "A Slope-robust Cascaded Ground Segmentation in 3D Point Cloud for Autonomous Vehicles". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 497–504. DOI: 10.1109/ITSC.2018.8569534.
- [63] Naujoks, B. and Wuensche, H.-J. "An Orientation Corrected Bounding Box Fit Based on the Convex Hull under Real Time Constraints". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, pp. 1–6. DOI: 10.1109/IVS.2018.8500692.
- [64] Ni, K., Jin, H., and Dellaert, F. "GroupSAC: Efficient consensus in the presence of groupings". In: *2009 IEEE 12th International Conference on Computer Vision*. 2009, pp. 2193–2200. DOI: 10.1109/ICCV.2009.5459241.

- [65] Niedfeldt, P. C. and Beard, R. W. “Recursive RANSAC: Multiple Signal Estimation with Outliers”. In: *IFAC Proceedings Volumes* 46.23 (2013). 9th IFAC Symposium on Nonlinear Control Systems, pp. 430–435. ISSN: 1474-6670. DOI: <https://doi.org/10.3182/20130904-3-FR-2041.00213>. URL: <https://www.sciencedirect.com/science/article/pii/S1474667016316974>.
- [66] Nister. “Preemptive RANSAC for live structure and motion estimation”. In: *Proceedings Ninth IEEE International Conference on Computer Vision*. 2003, 199–206 vol.1. DOI: 10.1109/ICCV.2003.1238341.
- [67] Oksuz, K., Cam, B. C., Kalkan, S., and Akbas, E. *Imbalance Problems in Object Detection: A Review*. 2019. DOI: 10.48550/ARXIV.1909.00169. URL: <https://arxiv.org/abs/1909.00169>.
- [68] Olah, C. “Understanding Convolutions”. In: *colah’s blog* (July 13, 2014). URL: <http://colah.github.io/posts/2014-07-Understanding-Convolutions/> (visited on 08/28/2022).
- [69] Oniga, F. and Nedevschi, S. “A Fast Ransac Based Approach for Computing the Orientation of Obstacles in Traffic Scenes”. In: *2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP)*. 2018, pp. 209–214. DOI: 10.1109/ICCP.2018.8516642.
- [70] *Providentia++: Research for automated driving with external infrastructure*. URL: <http://innovation-mobility.com/en/> (visited on 09/11/2022).
- [71] Qian, R., Lai, X., and Li, X. “3D Object Detection for Autonomous Driving: A Survey”. In: *Pattern Recognition* 130 (Oct. 2022), p. 108796. DOI: 10.1016/j.patcog.2022.108796. URL: <https://doi.org/10.1016%2Fj.patcog.2022.108796>.
- [72] Qu, S., Chen, G., Ye, C., Lu, F., Wang, F., Xu, Z., and Ge, Y. *An Efficient L-Shape Fitting Method for Vehicle Pose Detection with 2D LiDAR*. 2018. DOI: 10.48550/ARXIV.1812.09670. URL: <https://arxiv.org/abs/1812.09670>.
- [73] Rathinam, A. and Dempster, A. “Octree-Based Mascon Model for Small Body Gravity Fields”. In: *Journal of Guidance, Control, and Dynamics* 42 (July 2019), pp. 1–11. DOI: 10.2514/1.G004008.
- [74] Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.-Y., Johnson, J., and Gkioxari, G. “Accelerating 3D Deep Learning with PyTorch3D”. In: *arXiv:2007.08501* (2020).
- [75] Research), U. ( I. A. “Intersections”. In: (). URL: <https://www.udv.de/udv-en/topics/road/intersections> (visited on 08/30/2022).
- [76] Rhodes, A. D., Quinn, M. H., and Mitchell, M. “Fast On-Line Kernel Density Estimation for Active Object Localization”. In: *CoRR* abs/1611.05369 (2016). arXiv: 1611.05369. URL: <http://arxiv.org/abs/1611.05369>.
- [77] Rosenfeld, A. and Pfaltz, J. L. “Sequential Operations in Digital Picture Processing”. In: *J. ACM* 13.4 (Oct. 1966), pp. 471–494. ISSN: 0004-5411. DOI: 10.1145/321356.321357. URL: <https://doi.org/10.1145/321356.321357>.
- [78] Rusu, R. B. “Semantic 3D object maps for everyday manipulation in human living environments”. In: *KI-Künstliche Intelligenz* 24.4 (2010), pp. 345–348.
- [79] Rusu, R. B. and Cousins, S. “3D is here: Point Cloud Library (PCL)”. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 1–4. DOI: 10.1109/ICRA.2011.5980567.
- [80] Schwartz, R., Dodge, J., Smith, N. A., and Etzioni, O. *Green AI*. 2019. DOI: 10.48550/ARXIV.1907.10597. URL: <https://arxiv.org/abs/1907.10597>.

- [81] Seebacher, S., Datler, B., Erhart, J., Greiner, G., Harrer, M., Hrassnig, P., Präsent, A., Schwarzl, C., and Ullrich, M. *Infrastructure data fusion for validation and future enhancements of autonomous vehicles' perception on Austrian motorways: The 8th IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. Piscataway, NJ: IEEE, 2019. ISBN: 9781728101422. DOI: 10.1109/ICCVE45908.2019. URL: <http://ieeexplore.ieee.org/servlet/opac?punumber=8952572>.
- [82] Shen, X., Pendleton, S., and Ang, M. H. "Efficient L-shape fitting of laser scanner data for vehicle pose estimation". In: *2015 IEEE 7th International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*. 2015, pp. 173–178. DOI: 10.1109/ICCIS.2015.7274568.
- [83] Shen, Z., Liang, H., Lin, L., Wang, Z., Huang, W., and Yu, J. "Fast Ground Segmentation for 3D LiDAR Point Cloud Based on Jump-Convolution-Process". In: *Remote Sensing* 13.16 (2021). ISSN: 2072-4292. DOI: 10.3390/rs13163239. URL: <https://www.mdpi.com/2072-4292/13/16/3239>.
- [84] Shi, J., Xu, L., Heng, L., and Shen, S. "Graph-Guided Deformation for Point Cloud Completion". In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 7081–7088. DOI: 10.1109/LRA.2021.3097081.
- [85] Simpson, D. G. "Introduction to Rousseeuw (1984) Least Median of Squares Regression". In: 1997.
- [86] Song, Y., Zhang, H., Liu, Y., Liu, J., Zhang, H., and Song, X. "Background Filtering and Object Detection With a Stationary LiDAR Using a Layer-Based Method". In: *IEEE Access* 8 (2020), pp. 184426–184436. DOI: 10.1109/ACCESS.2020.3029341.
- [87] Spencer, B. "Ouster Lidar aids German V2X plans". In: *ITS International* (Nov. 3, 2021). URL: <https://www.itsinternational.com/its4/its7/its8/news/ouster-lidar-aids-german-v2x-plans> (visited on 08/28/2022).
- [88] Srinivasan, A., Mahartayasa, Y., Jammula, V. C., Lu, D., Como, S., Wishart, J., Yang, Y., and Yu, H. "Infrastructure-Based LiDAR Monitoring for Assessing Automated Driving Safety". In: Mar. 2022. DOI: 10.4271/2022-01-0081.
- [89] Strigel, E., Meissner, D., Seeliger, F., Wilking, B., and Dietmayer, K. "The Ko-PER intersection laserscanner and video dataset". In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. 2014, pp. 1900–1901. DOI: 10.1109/ITSC.2014.6957976.
- [90] Sun, B. and Mordohai, P. *Oriented Point Sampling for Plane Detection in Unorganized Point Clouds*. 2019. DOI: 10.48550/ARXIV.1905.02553. URL: <https://arxiv.org/abs/1905.02553>.
- [91] Sun, P., Kretschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., and Anguelov, D. "Scalability in Perception for Autonomous Driving: Waymo Open Dataset". In: *CoRR* abs/1912.04838 (2019). arXiv: 1912.04838. URL: <http://arxiv.org/abs/1912.04838>.
- [92] Sun, Y., Xu, H., Wu, J., Zheng, J., and Dietrich, K. M. "3-D Data Processing to Extract Vehicle Trajectories from Roadside LiDAR Data". In: *Transportation Research Record* 2672.45 (2018), pp. 14–22. DOI: 10.1177/0361198118775839. eprint: <https://doi.org/10.1177/0361198118775839>. URL: <https://doi.org/10.1177/0361198118775839>.

- [93] Tang, P., Wang, X., Bai, S., Shen, W., Bai, X., Liu, W., and Yuille, A. L. "PCL: Proposal Cluster Learning for Weakly Supervised Object Detection". In: *CoRR abs/1807.03342* (2018). arXiv: 1807.03342. URL: <http://arxiv.org/abs/1807.03342>.
- [94] Tarko, A. P., Ariyur, K. B., Romero, M., Bandaru, V. K., and Jimenez, C. L. "Guaranteed LiDAR-aided Multi-Object Tracking at Road Intersections". In: 2016.
- [95] Tian, H., Chen, Y., Dai, J., Zhang, Z., and Zhu, X. "Unsupervised Object Detection with LiDAR Clues". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 5958–5968. DOI: 10.1109/CVPR46437.2021.00590.
- [96] Torr, P. and Zisserman, A. "MLE-SAC: A New Robust Estimator with Application to Estimating Image Geometry". In: *Computer Vision and Image Understanding* 78.1 (2000), pp. 138–156. ISSN: 1077-3142. DOI: <https://doi.org/10.1006/cviu.1999.0832>. URL: <https://www.sciencedirect.com/science/article/pii/S107731429908329>.
- [97] Velas, M., Spanel, M., Hradis, M., and Herout, A. "CNN for very fast ground segmentation in velodyne LiDAR data". In: *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. 2018, pp. 97–103. DOI: 10.1109/ICARSC.2018.8374167.
- [98] Wang, B., Lan, J., and Gao, J. "LiDAR Filtering in 3D Object Detection Based on Improved RANSAC". In: *Remote Sensing* 14.9 (2022). ISSN: 2072-4292. DOI: 10.3390/rs14092110. URL: <https://www.mdpi.com/2072-4292/14/9/2110>.
- [99] Wang, G., Wu, J., Xu, T., and Tian, B. "3D Vehicle Detection With RSU LiDAR for Autonomous Mine". In: *IEEE Transactions on Vehicular Technology* 70.1 (2021), pp. 344–355. DOI: 10.1109/TVT.2020.3048985.
- [100] Wang, H., Zhang, X., Li, Z., Li, J., Wang, K., Lei, Z., and Haibing, R. "IPS300+: a Challenging multi-modal data sets for Intersection Perception System". In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 2539–2545. DOI: 10.1109/ICRA46639.2022.9811699.
- [101] Wang, L., Zhao, C., and Wang, J. "A LiDAR-Based Obstacle-Detection Framework for Autonomous Driving". In: *2020 European Control Conference (ECC)*. 2020, pp. 901–905. DOI: 10.23919/ECC51009.2020.9143607.
- [102] Wang, L. and Lan, J. "Adaptive Polar-Grid Gaussian-Mixture Model for Foreground Segmentation Using Roadside LiDAR". In: *Remote Sensing* 14.11 (2022). ISSN: 2072-4292. DOI: 10.3390/rs14112522. URL: <https://www.mdpi.com/2072-4292/14/11/2522>.
- [103] Wu, J. "Data Processing Algorithms and Applications of LiDAR-Enhanced Connected Infrastructure Sensing". PhD thesis. University of Nevada, 2018. URL: <http://hdl.handle.net/11714/4871>.
- [104] Wu, J., Lv, C., Pi, R., Ma, Z., Zhang, H., Sun, R., Song, Y., and Wang, K. "A Variable Dimension-Based Method for Roadside LiDAR Background Filtering". In: *IEEE Sensors Journal* 22.1 (2022), pp. 832–841. DOI: 10.1109/JSEN.2021.3125623.
- [105] Wu, J. and Xu, H. "An Automatic Procedure for Vehicle Tracking with a Roadside LiDAR Sensor". In: 2018.
- [106] Wu, J., Xu, H., Sun, Y., Zheng, J., and Yue, R. "Automatic Background Filtering Method for Roadside LiDAR Data". In: *Transportation Research Record* 2672.45 (2018), pp. 106–114. DOI: 10.1177/0361198118775841. eprint: <https://doi.org/10.1177/0361198118775841>. URL: <https://doi.org/10.1177/0361198118775841>.

- [107] Wu, J., Xu, H., Tian, Y., Pi, R., and Yue, R. "Vehicle Detection under Adverse Weather from Roadside LiDAR Data". In: *Sensors* 20.12 (2020). ISSN: 1424-8220. DOI: 10.3390/s20123433. URL: <https://www.mdpi.com/1424-8220/20/12/3433>.
- [108] Wu, J., Xu, H., and Zhao, J. "Automatic Lane Identification Using the Roadside LiDAR Sensors". In: *IEEE Intelligent Transportation Systems Magazine* 12.1 (2020), pp. 25–34. DOI: 10.1109/MITS.2018.2876559.
- [109] Wu, J., Xu, H., and Zheng, J. "Automatic background filtering and lane identification with roadside LiDAR data". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. 2017, pp. 1–6. DOI: 10.1109/ITSC.2017.8317723.
- [110] Wu, J., Xu, H., Zheng, J., and Zhao, J. "Automatic Vehicle Detection With Roadside LiDAR Data Under Rainy and Snowy Conditions". In: *IEEE Intelligent Transportation Systems Magazine* 13.1 (2021), pp. 197–209. DOI: 10.1109/MITS.2019.2926362.
- [111] Wu, J., Xu, H., Zheng, Y., Zhang, Y., Lv, B., and Tian, Z. "Automatic Vehicle Classification using Roadside LiDAR Data". In: *Transportation Research Record* 2673.6 (2019), pp. 153–164. DOI: 10.1177/0361198119843857. eprint: <https://doi.org/10.1177/0361198119843857>. URL: <https://doi.org/10.1177/0361198119843857>.
- [112] Xiao, A., Huang, J., Guan, D., Zhang, X., and Lu, S. *Unsupervised Point Cloud Representation Learning with Deep Neural Networks: A Survey*. 2022. DOI: 10.48550/ARXIV.2202.13589. URL: <https://arxiv.org/abs/2202.13589>.
- [113] Xiao, W., Vallet, B., Schindler, K., and Paparoditis, N. "Simultaneous Detection and Tracking of Pedestrian from Velodyne Laser Scanning Data". en. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume III-3, 2016, XXIII ISPRS Congress, 12–19 July 2016, Prague, Czech Republic*. Vol. III. 3. 2016 ISPRS Annual Congress of the Photogrammetry, Remote Sensing and Spatial Information Sciences; Conference Location: Prague, Czech Republic; Conference Date: July 12-19, 2016. Prague: Copernicus, 2016, pp. 295–302. DOI: 10.5194/isprs-annals-III-3-295-2016.
- [114] Xie, S., Gu, J., Guo, D., Qi, C. R., Guibas, L. J., and Litany, O. *PointContrast: Unsupervised Pre-training for 3D Point Cloud Understanding*. 2020. DOI: 10.48550/ARXIV.2007.10985. URL: <https://arxiv.org/abs/2007.10985>.
- [115] Xu, Q., Zhou, Y., Wang, W., Qi, C. R., and Anguelov, D. "SPG: Unsupervised Domain Adaptation for 3D Object Detection via Semantic Point Generation". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 15426–15436. DOI: 10.1109/ICCV48922.2021.01516.
- [116] Yang, S., Xiang, Z., Wu, J., Wang, X., Sun, H., Xin, J., and Zheng, N. "Efficient Rectangle Fitting of Sparse Laser Data for Robust On-Road Object Detection". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, pp. 1–8. DOI: 10.1109/IVS.2018.8500716.
- [117] Ye, X., Shu, M., Li, H., Shi, Y., Li, Y., Wang, G., Tan, X., and Ding, E. *Rope3D: The Roadside Perception Dataset for Autonomous Driving and Monocular 3D Object Detection Task*. 2022. DOI: 10.48550/ARXIV.2203.13608. URL: <https://arxiv.org/abs/2203.13608>.
- [118] Yu, H., Luo, Y., Shu, M., Huo, Y., Yang, Z., Shi, Y., Guo, Z., Li, H., Hu, X., Yuan, J., and Nie, Z. *DAIR-V2X: A Large-Scale Dataset for Vehicle-Infrastructure Cooperative 3D Object Detection*. 2022. arXiv: 2204.05575 [cs.CV].
- [119] Zeng, N. "An Introduction to Evaluation Metrics for Object Detection". In: *NICKZENG* (Dec. 16, 2018). URL: <https://blog.zenggyu.com/en/post/2018-12-16/an-introduction-to-evaluation-metrics-for-object-detection/> (visited on 09/11/2022).

- [120] Zhang, C., Wang, S., Yu, B., Li, B., and Zhu, H. “A Two-Stage Adaptive Clustering Approach for 3D Point Clouds”. In: *2019 4th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*. 2019, pp. 11–16. DOI: 10.1109/ACIRS.2019.8936035.
- [121] Zhang, J., Xiao, W., Coifman, B., and Mills, J. P. “IMAGE-BASED VEHICLE TRACKING FROM ROADSIDE LIDAR DATA”. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLII-2/W13* (2019), pp. 1177–1183. DOI: 10.5194/isprs-archives-XLII-2-W13-1177-2019.
- [122] Zhang, J., Xiao, W., Coifman, B., and Mills, J. P. “Vehicle Tracking and Speed Estimation From Roadside Lidar”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 13 (2020), pp. 5597–5608. DOI: 10.1109/JSTARS.2020.3024921.
- [123] Zhang, K., Chen, S.-C., Whitman, D., Shyu, M.-L., Yan, J., and Zhang, C. “A progressive morphological filter for removing nonground measurements from airborne LIDAR data”. In: *IEEE Transactions on Geoscience and Remote Sensing* 41.4 (2003), pp. 872–882. DOI: 10.1109/TGRS.2003.810682.
- [124] Zhang, L., Zheng, J., Sun, R., and Tao, Y. “GC-Net: Gridding and Clustering for Traffic Object Detection With Roadside LiDAR”. In: *IEEE Intelligent Systems* 36.4 (2021), pp. 104–113. DOI: 10.1109/MIS.2020.2993557.
- [125] Zhang, M., Fu, R., Morris, D. D., and Wang, C. “A Framework for Turning Behavior Classification at Intersections Using 3D LIDAR”. In: *IEEE Transactions on Vehicular Technology* 68.8 (2019), pp. 7431–7442. DOI: 10.1109/TVT.2019.2926787.
- [126] Zhang, T. and Jin, P. J. “Roadside LiDAR Vehicle Detection and Tracking Using Range and Intensity Background Subtraction”. In: *Journal of Advanced Transportation* 2022 (Apr. 2022). Ed. by Armingol, J. M., pp. 1–14. DOI: 10.1155/2022/2771085. URL: <https://doi.org/10.1155%2F2022%2F2771085>.
- [127] Zhang, T., Jin, P. J., and Ge, Y. *Weighted Bayesian Gaussian Mixture Model for Roadside LiDAR Object Detection*. 2022. DOI: 10.48550/ARXIV.2204.09804. URL: <https://arxiv.org/abs/2204.09804>.
- [128] Zhang, W. “LIDAR-based road and road-edge detection”. In: *2010 IEEE Intelligent Vehicles Symposium*. 2010, pp. 845–848. DOI: 10.1109/IVS.2010.5548134.
- [129] Zhang, X., Xu, W., Dong, C., and Dolan, J. M. “Efficient L-shape fitting for vehicle detection using laser scanners”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 54–59. DOI: 10.1109/IVS.2017.7995698.
- [130] Zhang, Y., Bhattarai, N., Zhao, J., Liu, H., and Xu, H. “An Unsupervised Clustering Method for Processing Roadside LiDAR Data With Improved Computational Efficiency”. In: *IEEE Sensors Journal* 22.11 (2022), pp. 10684–10691. DOI: 10.1109/JSEN.2022.3166957.
- [131] Zhang, Y., Xu, H., and Wu, J. “An Automatic Background Filtering Method for Detection of Road Users in Heavy Traffics Using Roadside 3-D LiDAR Sensors With Noises”. In: *IEEE Sensors Journal* 20.12 (2020), pp. 6596–6604. DOI: 10.1109/JSEN.2020.2976663.
- [132] Zhang, Z., Zheng, J., Wang, X., and Fan, X. “Background Filtering and Vehicle Detection with Roadside Lidar Based on Point Association”. In: *2018 37th Chinese Control Conference (CCC)*. 2018, pp. 7938–7943. DOI: 10.23919/ChiCC.2018.8484040.

- [133] Zhang, Z., Zheng, J., Xu, H., and Wang, X. “Vehicle Detection and Tracking in Complex Traffic Circumstances with Roadside LiDAR”. In: *Transportation Research Record* 2673.9 (2019), pp. 62–71. DOI: 10.1177/0361198119844457. eprint: <https://doi.org/10.1177/0361198119844457>. URL: <https://doi.org/10.1177/0361198119844457>.
- [134] Zhang, Z., Zheng, J., Xu, H., Wang, X., Fan, X., and Chen, R. “Automatic Background Construction and Object Detection Based on Roadside LiDAR”. In: *IEEE Transactions on Intelligent Transportation Systems* 21.10 (2020), pp. 4086–4097. DOI: 10.1109/TITS.2019.2936498.
- [135] Zhao, C., Fu, C., Dolan, J. M., and Wang, J. “L-Shape Fitting-Based Vehicle Pose Estimation and Tracking Using 3D-LiDAR”. In: *IEEE Transactions on Intelligent Vehicles* 6.4 (2021), pp. 787–798. DOI: 10.1109/TIV.2021.3078619.
- [136] Zhao, H., Zhang, Q., Chiba, M., Shibasaki, R., Cui, J., and Zha, H. “Moving object classification using horizontal laser scan data”. In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 2424–2430. DOI: 10.1109/ROBOT.2009.5152347.
- [137] Zhao, J., Xu, H., Liu, H., Wu, J., Zheng, Y., and Wu, D. “Detection and tracking of pedestrians and vehicles using roadside LiDAR sensors”. In: *Transportation Research Part C: Emerging Technologies* 100 (2019), pp. 68–87. ISSN: 0968-090X. DOI: <https://doi.org/10.1016/j.trc.2019.01.007>. URL: <https://www.sciencedirect.com/science/article/pii/S0968090X19300282>.
- [138] Zhao, J., Xu, H., Xia, X., and Liu, H. “Azimuth-Height Background Filtering Method for Roadside LiDAR Data”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 2421–2426. DOI: 10.1109/ITSC.2019.8917369.
- [139] Zheng, J., Yang, S., Wang, X., Xiao, Y., and Li, T. “Background Noise Filtering and Clustering With 3D LiDAR Deployed in Roadside of Urban Environments”. In: *IEEE Sensors Journal* 21.18 (2021), pp. 20629–20639. DOI: 10.1109/JSEN.2021.3098458.
- [140] Zhou, Q.-Y., Park, J., and Koltun, V. *Open3D: A Modern Library for 3D Data Processing*. 2018. DOI: 10.48550/ARXIV.1801.09847. URL: <https://arxiv.org/abs/1801.09847>.
- [141] Zimmer, W., Ercelik, E., Zhou, X., Ortiz, X. J. D., and Knoll, A. *A Survey of Robust 3D Object Detection Methods in Point Clouds*. 2022. DOI: 10.48550/ARXIV.2204.00106. URL: <https://arxiv.org/abs/2204.00106>.
- [142] Zimmer, W., Grabler, M., and Knoll, A. *Real-Time and Robust 3D Object Detection Within Road-Side LiDARs Using Domain Adaptation*. 2022. DOI: 10.48550/ARXIV.2204.00132. URL: <https://arxiv.org/abs/2204.00132>.