SPATIOTEMPORAL ANALYTICS OF LIDAR DATA FOR ENVIRONMENTAL PERCEPTION TO ASSIST AUTONOMOUS DRIVING

Katkoria Dhvani Shaileshbhai

Master of Science by Research Thesis

October 2022

International Institute of Information Technology, Bangalore

SPATIOTEMPORAL ANALYTICS OF LIDAR DATA FOR ENVIRONMENTAL PERCEPTION TO ASSIST AUTONOMOUS DRIVING

Submitted to International Institute of Information Technology,
Bangalore
in Partial Fulfillment of
the Requirements for the Award of
Master of Science by Research

by

Katkoria Dhvani Shaileshbhai
MS2019007

International Institute of Information Technology, Bangalore
October 2022

Dedicated to my family

Thesis Certificate

This is to certify that the thesis titled **Spatiotemporal analytics of LiDAR Data for Environmental Perception to Assist Autonomous Driving** submitted to the International Institute of Information Technology, Bangalore, for the award of the degree of **Master of Science by Research** is a bona fide record of the research work done by **Katkoria Dhvani Shaileshbhai**, **MS2019007**, under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Jaya Sreevalsan Nair

Bangalore,

The 6th of October, 2022.

SPATIOTEMPORAL ANALYTICS OF LIDAR DATA FOR ENVIRONMENTAL PERCEPTION TO ASSIST AUTONOMOUS DRIVING

Abstract

Accurate and reliable perception of nearby objects of interest, such as on-road impediments, ground surfaces, curbs and ditch, navigable surfaces, etc., is crucial for course planning and localization in autonomous driving. Spatio-temporal analysis of autonomous driving data is the task of extracting semantic and geometric structures that approximate spatial and temporal relationships for concise perception tasks from a large set of data. The ability of autonomous vehicles to perceive their surroundings is one of their most crucial functions. Few of the various methods for perceiving the environment and its surroundings that have been suggested utilize semantic information to improve the perception models. This turns finding perception task-appropriate methodologies or developing new approaches with semantic and geometric information into a significant research challenge. This thesis proposes geometry-aware approaches with a high potential for contributing to the classification, detection, and reconstruction tasks. In this project, we aim to incorporate semantic and geometric information into the process for perception tasks of ground and non-ground classification, road edge detection, road surface reconstruction, and 3D object detection.

There is active research on computer vision solutions to the problems of perceptions by proposing novel systems or analysis methods. However, there is still scope for improvement for many of these solutions because of the limitations imposed by the complexity of the components in environments, with a simultaneous representation of stationary and movable environments, the barriers in the way of the autonomous vehi-

cle, and the variety of the external environment. For crucial perceptual tasks, we provide an automated approach with the aid of spatiotemporal analysis and improvement. Our work intends to handle three crucial perception-related tasks for autonomous driving, including (i) classifying ground and non-ground, (ii) extracting the road surface, and (iii) enhancing 3D object detection.

To distinguish between drivable zones and impassable obstacles, the ground from non-ground classification method comprises decoding of the proper height-based features by using neighborhood processing techniques of the data points obtained by the LiDAR sensor in the form of 3D point cloud data. We interpret the ground and non-ground elements separately based on the semantics and geometrical perception of the point cloud. Using a supervised machine learning classifier, we divide the data points into their respective ground and non-ground classes as a first step towards the crucial task of perception.

Then the detected ground elements are heuristically separated and labeled into road edge components. The processing mechanisms for the next important perceptual task include road edge point detection, surface mesh generation, our proposed frame classification for identification of road topological structure, and edge point set smoothing, along with a few preprocessing making our method more efficient than existing methods. Finally, we propose automatic road surface extraction using geometry-based and temporal sequential extraction techniques based on the road type to exploit properties of the semantic structure of road rendered in the range image representation.

We also propose a design methodology for improving 3D object detection with the help of adaptive learning. To build a fresh set of more accurate detections, we propose the adaptive model, built based on two-way matching and using shape descriptors that provide geometrical information encoded in detected objects to select one detection out of an ensemble of detections from different models.

For interpreting the environment during autonomous driving and representing the extracted information as ground data, road surface, and enhancement of object detection, we use both spatial and temporal processing and machine learning approaches in our workflow. The effectiveness of our approaches for each perception task is evaluated qualitatively and quantitatively. Overall, the state-of-the-art methods for significant perceptual tasks to support autonomous driving is improved by our unique methodologies based on spatiotemporal analysis and processing, machine learning, and geometrical computations.

Acknowledgements

"My profound appreciation to my supervisor, Professor Jaya Sreevalsan Nair, for her wise counsel, ongoing inspiration, and support. It has been a pleasure for me to work with her. I would like to thank the Machine Intelligence and Robotics Center at IIIT-Bangalore and Ignitarium Technologies Pvt. Ltd. for their kind financial support. Additionally, I want to express my gratitude to the GVCL team and the Ignitarium team for fostering a positive work atmosphere and sharing their priceless expertise. Talking with them was always enjoyable and calming. Additionally, I would like to thank Ms. Pragyan Mohapatra, my lab partner, for working with me and helping me identify my research passion. Last but not least, I would like to express my gratitude to my family for continuing to believe in me and that I can complete this journey."

— Dhvani Katkoria

List of Publications

[1] **D. Katkoria** and J. Sreevalsan-Nair, "RoSELS: Road Surface Extraction for 3D Automotive LiDAR Point Cloud Sequence," in Proceedings of the 3rd International Conference on Deep Learning Theory and Applications (DeLTA), SciTePress, 2022, 55–67. ISBN: 978-989-758-584-5, DOI: 10.5220/0011301700003277.

Contents

List of Figures

List of Tables

List of Abbreviations

IIITB ..... International Institute of Information Technology Bangalore

LiDAR ..... Light Detection and Ranging

PDF ..... Probability Density Function

RPN ..... Region Proposal Network

ESF ..... Ensemble of Shape Functions

FPFH ..... Fast Point Feature Histogram

VFH ..... Viewpoint Feature Histogram

CVFH ..... Clustered Viewpoint Feature Histograms

GASD ..... Globally Aligned Spatial Distribution

RFC ..... Random Forest Classifier

RFR ..... Random Forest Regressor

BEV ..... Bird's Eye View

CHAPTER 1

INTRODUCTION

The development of self-driving vehicle technology has gradually increased over the past three decades, primarily due to advancements in computer and sensing technologies that have decreased the size and cost of essential hardware. Perception, planning, control, and coordination are the different categories of requirements of autonomous vehicle software [7]. These competencies interact with one another and with how the vehicle interacts with its surroundings. The ability of autonomous cars to perceive their surroundings is one of their most crucial functions. Environment perception is a critical component of autonomous vehicles since it gives the car vital information about its surroundings, including the positions, velocities, and potential future states of nearby barriers as well as the free drivable areas [8]. Depending on the sensors used, LiDARs, cameras, or a combination of these two types of equipment can be used to solve the environment perception task. In the field of intelligent vehicles, where a self-driving vehicle is required to travel in a complex environment while also identifying objects safely, spatio-temporal analysis and evaluation of contextual situation awareness are particularly crucial. The three essential tasks in the perception domain necessary for autonomous driving are ground and non-ground classification, extraction of drivable surfaces, and 3D object detection. There are many state-of-the-art techniques for carrying out these tasks; we investigate these techniques and carry out spatial and temporal analyses that aid in better comprehending the data and the scene, opening up new av-

enues for advancements and state-of-the-art.

Systems for perception must be able to recognize and represent different participants in all environments, including unexpected environments. This is especially true in urban areas with a high volume of traffic and complicated lane and road configurations. These factors, along with various forms of peripheral infrastructure and vegetation, make perception difficult. Perception tasks involve data input from different sensor modalities *i.e.* LiDAR, camera, and radar sensors are widely used.

Light detection and ranging device, often known as LiDAR, transmits millions of light pulses per second in a precise pattern and it can produce a dynamic, three-dimensional map of the environment. Most autonomous vehicles today rely heavily on LiDAR for perception tasks. The points the LiDAR returns in an actual situation are never accurate. The scan point sparsity, missing points, and disorganized patterns present challenges in managing LiDAR points. However, LiDAR data is much more accurate with concise depth information as compared to camera images which suffer due to the different appearance, shape, and attitudes of various objects, as well as the interference of lighting, shielding, and other factors during imaging, in the field of computer vision, using images [9].

The terrain surface is divided into drivable regions and impassable obstacles in the first processing stage. Despite being assumed in many approaches, the ground surface typically has a random structure rather than a single plane. For example, discern a car going on the road even within a tunnel, or beneath a tree, protruding and overhanging structures must be identifiable from the ground surface. Explicit depiction of the ground surface produces useful information regarding drivable locations, but many techniques interpret ground data as noise and concentrate on their removal. Widely used strategies either use grid-based intermediate representations or vertical displacement analysis [10,11]. These methods typically struggle to model closer objects accurately. Others

use scan lines of multibeam LiDAR to categorize the ground surface based on a slope threshold. This can be done either using point pairs as a basis [12, 13], or by accounting for bigger point clusters [14, 15]. These algorithms are occasionally subject to inaccurate measurements.

We extract hand-crafted pertinent height-based features to address the drawbacks of existing approaches while maintaining their modest processing requirements. A technique based on the neighborhood extracts features. After carrying out a proper spatial analysis based on Local Geometric Descriptor (LGD) and selecting an appropriate neighborhood size, the relevant height features are extracted. We precisely differentiate ground and non-ground segments by using multi-scale feature extraction to propagate information about ground-level fluctuations.

Another one of the most crucial processes in the assessment of navigable space is the collection of three-dimensional (3D) information about the road surface. Road surface detection informs the autonomous vehicle on the locations of free space where it can drive without collision. It is the prerequisite for any online path planning and control operations. Many people have historically used visual inspection and interpretation to assess road surface quality [16]. The road environment is highly complex and heterogeneous. The heterogeneous environment makes it challenging to collect and accurately analyze roadway data spanning thousands of kilometers of route. The foundational information for further analysis is high-precision road terrain obtained as point cloud data, which is used for applications like road surface settlement, pavement, and slope collapse [17].

We provide an automated technique for extracting road surfaces, which may be divided into two parts: (I) curb-based methods and (II) global properties of roadways (e.g., topology and smoothness). Point clouds from temporal frames are segmented into the ground and non-ground categories as a starting point. Edge detection for the

curb is done after the ground segment has been separated, and mesh creation is done to extract the road surface. Curb points are discovered by geometrically analyzing nearby locations and range image representation. To locate and follow the curb, the topology, smoothness, and trajectory information of point cloud sequences are taken into consideration. The topological road structure is extracted from temporal frames of point cloud sequences using our suggested innovative frame classification algorithm, which is then used to derive potential road surfaces.

Researchers have been particularly interested in driving in urban settings because of the high vehicle density and different area-specific traffic regulations that must be followed. The surroundings of autonomous vehicle contain a variety of challenges, including sensitive objects like vehicles and pedestrians. The approaches discussed in [18, 19] are not reliable or broad enough to be used with autonomous cars because of the diverse types, appearances, shapes, and sizes of the objects. In object detection, deep learning has demonstrated superior performance compared to traditional learning or feature-based techniques. A collection of proposal bounding boxes must typically be created around the input for deep learning algorithms. Then, each proposal box is routed through the CNN network to identify a classification (including background) and fine-tune its bounding box positions. Selective Search [20] and EdgeBoxes [21], which both rely on affordable hand-crafted features and inference procedures, are the two popular techniques for proposing bounding boxes.

3D objects require segmentation, recognition, and localization as point clusters in space. Each cluster needs to be divided into distinct objects after segmentation. The spatial relationships surrounding the LiDAR point clouds and the shapes of the objects in entire, partial, or obstructed circumstances used in object recognition make up the majority of the information embedded in each cluster. Therefore, most algorithms use the computer vision detection problem through specific fusion methods, as we do for our analysis and detection enhancement. On top of the base 3D detection models, which

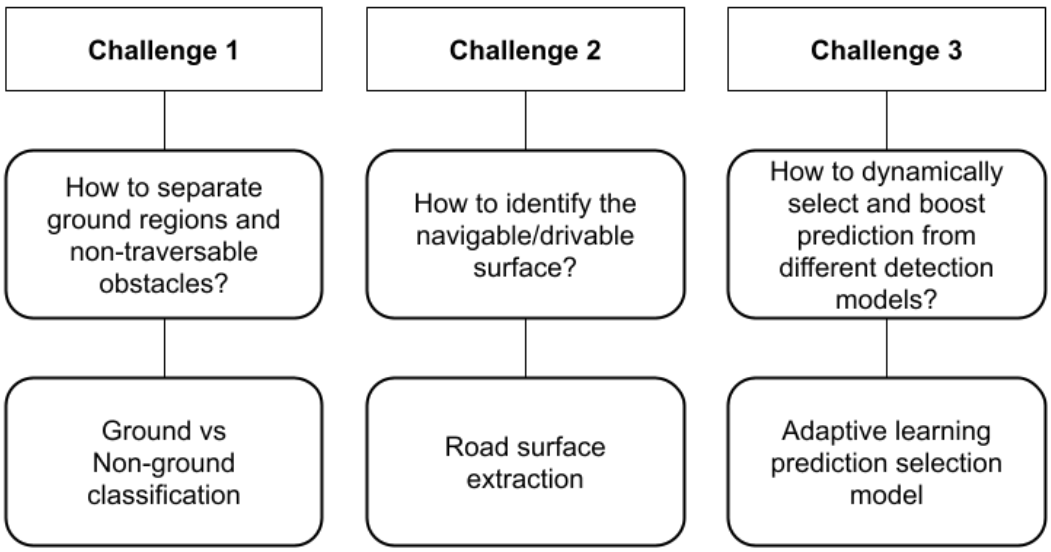


Figure FC1.1: Gaps and challenges in environment perception tasks addressed in our work to assist autonomous driving

would use the shape attributes of detected objects and two-way matching across detections of an ensemble of models, we propose employing an adaptive-learning prediction selection model to boost the 3D object detection.

1.1 Problem Statement

The spatio-temporal analysis of different components of perception tasks in autonomous driving induces point cloud data understanding and extraction of spatio-temporal features helpful in generating new systems and boosting the existing ones. The analysis involves improving and accomplishing computer vision tasks for autonomous driving assistance such as ground and non-ground segmentation, road edge detection, 3D surface extraction for navigable/drivable space, boosting 3D object detection by using an ensemble of models, etc. Our aim is to perform analysis in the spatial as well as the temporal domain of the point cloud data collected by the LiDAR sensor, which decodes different perception tasks based on the different segments the point cloud data

comprises of the ground and non-ground separation, extraction of a drivable surface from ground component and improving the detection of the objects present in the non-ground component. The three main research objectives and the challenges involved are summarized in Figure FC1.1. Our objective is to address the following problems:

- Ground and non-ground classification based on semantics and usage.
  - A point cloud collects several significant components to depict data effectively. These point cloud components are perceived in order and later assembled to form an overall interpretation of underlying information based on the steps involved in the analytical perception process. The separation of ground regions and non-traversable obstacles is crucial in interpreting the scene and extracting the drivable surface data. Existing methods are either insufficiently precise or rely on deep learning networks. Mapping the points that belong to the ground and non-ground requires a significant amount of effort if done manually or a significant amount of processing power if a deep learning network is used. Segmentation using hand-crafted features continues to be the best trade-off between efficiency and accuracy [22,23]. Random forest classifier (RFC) [24] has been known to work effectively for semantic segmentation of the airborne and terrestrial LiDAR point clouds [25, 26]. However, there is a gap in the use of effective use of RFCs for 3D automotive LiDAR point clouds. There is a need to determine an appropriate feature vector and preprocessing steps to address this gap.
- Road surface extraction using 3D LiDAR point cloud sequences
  - Navigable space detection is a difficult problem for robotics and intelligent vehicle technology that requires a combined approach of computational geometry and computer vision. The traditional data processing workflow uses semantic segmentation, which is capable of quickly identifying road points.

However, to identify curb or edge points, the semantic segmentation results must be post-processed. Thus, there is a gap in determining the curb points and extracting road surface using the entire trajectory information from a sequence of LiDAR point clouds. This entails a workflow that works at different scales, *i.e.* at the scale of a point, a point cloud, and the sequence of point clouds.

- Boosting the 3D object detection
  - 3D object detection is the task of localizing and recognizing objects in a 3D scene that are present around the self-driving vehicle. Individual detection model approaches may perform well in a given environment, but a single 3D object detection model may not produce comparable results across all detected objects in all environmental conditions when precise detection is more useful in all cases during autonomous driving. Ensembling features from several sources have been the focus of some works [27–29]. When using ensemble methods based on model output, the typical strategy is to use the main model whose predictions are modified by a secondary model, as done in [30, 31]. The use of methods to remove redundant bounding boxes, such as Non-Maximum Suppression [32], Soft-NMS [33], or WBF [6], is another strategy for combining the output of detection models. There is a gap in the state-of-the-art in use of ensemble models for selecting the output of detection models instead of merging them for 3D object detection. We focus on using pre-trained models to effectively use the state-of-the-art models. Hence, addressing the gap includes a decision-making workflow that includes a post-processing step to adaptively choose the outcomes of these models.

Our work in this thesis aims to explore and analyze diverse methods implemented for

three perception tasks and introduce our approach as an alternative and equally efficient solution for deriving information essential to assist autonomous driving. Our proposed methods work on the point cloud sequences captured from the Velodyne LiDAR sensor mounted on a mobile platform.

1.2 Contributions

Our novel contributions lie in integrating the appropriate existing and proposed methods in spatio-temporal analysis for three important tasks of ground and non-ground classification, 3D road surface extraction, and boosting 3D object detection to assist autonomous driving (Figure FC4.2). Our key contributions and implementations in this thesis include the following:

- We propose the use of point cloud registration as a medium for outlier removal to improve the representation of data using its statistical measures, which gives better feature computation and comparatively less processing time and the use of selected features based on the impact of different features with a supervised machine learning model, namely, the random forest classifier, for ground and non-ground point classification.
- We design and implement a complete automated system using ground points for road surface extraction from the ego-motion in 3D automotive LiDAR point clouds.
- We propose a novel per-frame road-geometry classification, *i.e.* frame classification, using appropriate image representation of the ground points and the process of transfer learning with the help of ImageNet weights instead of training the image classification model from scratch.

- We further propose appropriate point set processing using the range image representation of the 3D points for performing road edge point detection and the point set smoothing of the detected edge points by merging the edge points over continuous temporal frames and appropriate surface mesh generation methods for our road surface extraction.
- We design and implement a novel adaptive learning prediction-selection strategy of selecting appropriate detection from multiple detections obtained from 3D object detection methods. This is to improve the accuracy of 3D object detection of MLS point clouds using an effective application of two-way matching for preserving the bounding boxes based on the spatial locality of the detected bounding box from an ensemble of detection models,
- We propose a novel use of random forest regression to predict the distance between the shape descriptors of the detected object points and ground truth object points to select the detection from an ensemble of 3D detection models.

1.3 Thesis Structure

The thesis structure is as follows: in Chapter 2, we provide a literature survey on analysis and approaches taken in the areas of ground classification, segmentation, and extraction of road edge points and drivable surface, scene classification, and different models for 3D object detection and improvements on the same, which are essential tasks in autonomous driving, emphasizing the variety of implementations and their corresponding limitations. Chapter 3 discusses our first important step of ground and non-ground classification, which is widely adopted to form a basis in many use cases for object detection, tracking, and 3D surface extraction and reconstruction involving analysis of point neighborhood for better understanding of scene captured by LiDAR, outlier removal, inclusion of selected hand-crafted features, and point set classification, along

with the experiments conducted. In Chapter 4, we propose an algorithm for an automated system for 3D road surface extraction from sequences of 3D LiDAR point cloud data utilizing our ground segmentation approach and semantically inferring the road edge points to generate the 3D meshes corresponding to the road surface and the experimental results compared with existing methods. Chapter 5 focuses on boosting the 3D object detection task, another vital task in autonomous driving, using a decision-making model to adeptly pick one of the detected objects from the corresponding predictions of an ensemble of 3D detection models, along with experimentation of weighted box fusion for comparing and assessing the improvements. The potential applications of our work are concluded in Chapter 6, along with a comment on entire systems that can employ our algorithms as a module for each perception task discussed and the range of potential future work aimed at enhancing our methods.

CHAPTER 2

LITERATURE SURVEY

A significant amount of study has been done over the years on different perception tasks involved in the autonomous driving scenario to help the automated systems better understand the underlying components such as performing segmentation, localization, mapping, and object detection. The binary segmentation of point cloud involves revealing the ground and non-ground points, which form the basis for other tasks in many cases. The road surface extraction provides us with the extracted surface of the drivable area, a key component for a self-driving vehicle, which calls for ground segmentation, road edge point detection, road plane estimation, and scene understanding. 3D object detection is also one of the crucial tasks in autonomous driving. Different models provide different detections, all aiming for improvement in the detections. Boosting the 3D object detection would result in more efficient driving.

This chapter provides a thorough summary of different approaches and implementations involved in analyzing essential perception tasks throughout the previous few years, including various related methods. The state-of-the-art in 3D automotive LiDAR point cloud processing on the these topics is discussed in this chapter.

2.1 Pointwise Analysis

This section describes the various analysis performed on each point in the point cloud that are relevant to the various methods used in our work. It includes works on point-wise ground and non-ground classification, as well as various works on shape descriptors based on object points. This section provides a summary about literatures that address the challenges in the ground and non-ground classification and 3D object detection.

2.1.1 Ground and Non-Ground Classification

Ground point segmentation, comparable to classifying each point into "ground" and "non-ground points," serves as the starting point for our spatio-temporal analysis. Since the middle of the 2000s, this field of study has been active [22, 34]. There are two concurrent strategies: (i) using height-based hand-crafted features and conventional machine learning; and (ii) using convolutional neural networks (CNNs) or convolutional encoder-decoder, either with image representation for its projection (e.g., sparse pseudo image, bird's eye view (BEV), range image, etc.) or with 3D points directly [35]. The CNNS can be used directly for binary road segmentation; for example, [36]. Although CNNs perform best in trained contexts, they are expensive for training and less general in other environments. However, depending on the need, ground point segmentation has continued using height-based feature extraction [22,23]. For example, either geometry-based filtering [23] or elevation map image processing [37]) is carried out. Through voting [22], such images at different resolutions are used as input to an ensemble edge detection for probabilistic ground point segmentation.

A different approach is the fine-grained multi-class semantic segmentation [35], in which the relevant classes can be joined functionally to form a "ground" class [22, 34,

37]. Range pictures are frequently utilized for projection-based approaches employing deep learning, which belong to a class of semantic segmentation methods widely employed [38, 39]. Another group of networks, such as RandLA-Net [40], SCSSnet [41], etc., directly use 3D point sample sets.

After some testing, we recommend employing supervised learning with custom features to partition the ground at a reduced cost.

2.1.2 Shape Descriptors

The methods for extracting 3D point cloud descriptors widely in use are thoroughly examined in this survey. These methods can be broadly divided into local-based descriptors, global-based descriptors, and hybrid-based descriptors. Fast Point Feature Histogram (FPFH) [42] is introduced to simplify and reduce the computational complexity of the PFH descriptor and is used mainly for object registrations. It is a local descriptor that uses the relationships between point pairs in the support region and estimates surface normals to represent the geometric properties. By integrating extra perspective variance, the Viewpoint Feature Histogram (VFH) [43] expands on the concept and characteristics of FPFH. It is a global descriptor comprising of a surface form component and a viewpoint direction component used for object detection purposes. The Clustered Viewpoint Feature Histogram descriptor [44], a local/regional descriptor, can be considered an extension of VFH, which takes into account the advantage from stable object regions obtained by applying a region growing algorithm after removing the points with high curvature. It is used for object detection with pose estimation and detection of partial objects. Globally Aligned Spatial Distribution (GASD), a novel global descriptor proposed by Lima et al. [45], is based on a reference frame estimated for the entire point cloud model representing an object and is constructed based on Principal Component Analysis (PCA).

2.2 Road Surface Estimation

This section discusses various feature extraction methods, such as manual feature extraction based on analysis and feature extraction using deep learning networks, both of which are useful in tasks such as road surface extraction, scene classification, and ground plane estimation. It provides a summary of state-of-the-art that are relevant to the task of road surface extraction.

2.2.1 Road Edge Extraction

For mobile laser scanning (MLS)/LiDAR point clouds [46, 47], monocular pictures captured by moving vehicles [48], and elevation maps from 2D laser scanners [49], curb extraction has been examined. To find potential points or locations, all of these techniques employ elevation filtering and the proper line fitting algorithms. Our method is most similar to road boundary extraction for MLS point clouds [47], which is extracted by searching outward from the vehicle trajectory to discover edge points. The MLS point clouds differ, though, in that the search is carried out in the candidate point set, and we take advantage of the range image view of the vehicle LiDAR point cloud on which the search is carried out.

The curb points are designated as "sidewalk" in the benchmark SemanticKITTI dataset for vehicle LiDAR point clouds, where the points are first labeled in tiles by human annotator [2] and the road boundary/curb points are subsequently specifically refined [50]. The IoU (intersection over union) score for sidewalk is 75.5% in SCSSnett [41] and 75.2% with RangeNet++ [38] in the baseline techniques in the benchmark test of semantic segmentation performed using deep learning architectures.

As a result, we see that while deep learning algorithms for semantic segmentation are good at identifying road points, they fall short in detecting road edge points. The

class disparity can be used to explain this. Therefore, we provide a ground-based road edge recognition method that considers structure. Our strategy involves employing height-based hand-crafted features in supervised learning techniques, such as [22], to find road edge points.

2.2.2 Scene Classification

We look at the state-of-the-art in the area of scene categorization, which is closest to our novel frame classification. Transfer learning has been used for coarse scene classification on satellite or aerial images [51], where ResNet (Residual Network) has shown close to accurate performance [3]. ResNet with 50 layers (ResNet-50) offers the best performance and value for classifying the land cover in remote sensing images [52]. The KITTI vision benchmark suite has been used to classify the types of roads based on their functionality as "highways" and "non-highways" with the help of AlexNet [53]. We choose to use ResNet-50 for our frame classification rather than AlexNet since it performed better on aerial images. Our road extraction system requires a deep learning architecture that performs well for the top-view of the road. Perceptually, top views clearly distinguish between different road geometry classes, namely "straight" and "curved" roads.

2.2.3 Ground Plane Estimation

According to recent research on road extraction, the geometry extraction process is preceded by ground plane estimation and segmentation [34, 41]. To estimate ground elevation, GndNet creates a pseudo-image using 2D voxelization or pillars, which is then sent to a convolutional encoder-decoder [34]. SCSSnet conducts a precise ground plane estimate and employs semantic segmentation to locate ground points [41]. Since the direct extraction of coarse geometry is what we are after, we locate edge points

and triangulate them with trajectory points. The *mesh map*, which is a triangulation of an automotive LiDAR using surface normals produced from range images [54], is also different from our proposed 3D surface extraction method.

2.3 3D Object Detection Models

An autonomous driving system consists of perception, planning, decision, and control. Object detection is an essential aspect of many robotics applications and autonomous driving. 3D Object detection is a task of localization and classification of objects. Object detection in point clouds is an intrinsically three-dimensional problem. Recently, with advances in deep networks for point cloud data, several works [55, 56] have shown state-of-the-art 3D detection results with point cloud as the only input. One line of work [57, 58] uses mature 2D detectors to provide initial proposals in the form of frustums. This limits the 3D search space for estimating 3D bounding boxes. A new deep architecture [59] proposed for fusing camera and LiDAR sensors for 3D object detection combines the camera and LiDAR features using the cross-view spatial feature fusion strategy. The method [60] proposed a novel 3D object detector, KDA3D, which achieves high-precision and robust classification, segmentation, and localization with the help of key-point densification and multi-attention guidance. This section provides a summary of literatures that addresses the challenge in the 3D object detection task with ensemble models as mentioned in Chapter 1.

2.3.1 Adaptive Prediction-Selection Model

The most relevant work is used in the framework [61] for object tracking tasks. When the car is in an intersection, an adaptive strategy is used to enhance prediction and analysis outcomes by employing several prediction models. The various parts of the Gaussian Sum Filter (GSF), which correspond to the degree of uncertainty regard-

ing the direction in which the vehicle will travel, are provided to various turning models as the vehicle approaches the intersection. The findings demonstrate that, compared to employing a single, constant velocity model, employing several models has lowered the analysis RMSE in tracking and provided a better assessment of the analysis uncertainty. Another similar work is taking advantage of the spatial location of the features during adaptive multiscale feature extraction [62] for the decision-making process to discover the optimum multiscale approach. When using a particular multiscale method while employing the hybrid features and choosing voxels, the classification results demonstrate an improvement in overall accuracy.

2.3.2 Weighted Boxes Fusion

Weighted boxes fusion (WBF) is a novel method for combining predictions of object detection models. A 3D object detection pipeline [10] using point cloud sequence data uses the WBF for the test time augmentation by rotating the point cloud and ensembling predictions with weighted boxes fusion. The object detection approach in [11] also uses the test time augmentation with different augmentations, and the other usage is to merge the results of different models across different sub-designs.

2.4 Summary

We build upon the state of the art on feature extraction, ground plane estimation, and 3D object detection discussed in this chapter. Our novel contributions are elaborated in Chapters 3, 4, and 5, respectively.

CHAPTER 3

GROUND AND NON-GROUND CLASSIFICATION

In any outdoor environment autonomous system, the ground classification is an important preprocessing task for the local environment perception. Ground classification for LiDAR point cloud is a crucial procedure to ensure driving safety in an autonomous vehicle. Many tasks, like object detection, classification, and 3D reconstruction, can be carried out after the ground and non-ground components have been separated. The primary goal of ground classification is to separate ground points and positive obstacles above the ground from the 3D point cloud acquired by LiDAR, allowing for the further identification of road and obstacle types. The vehicle experiences changes in pitch, roll, and suspension when driving. The space between measurement points is also significant due to the uneven distribution of 3D LiDAR data, which causes the laser measurement points close to the LiDAR to be distributed relatively dense while the distribution of laser measurement points farther from the LiDAR to be distributed relatively sparse. The goal of this spatial analysis is to separate ground points and non-ground points by analyzing the neighborhood point distribution for each class.

Mapping the points that belong to ground and non-ground requires a lot of effort if manually done or a lot of processing power if a deep learning network is used. The proposed method segments ground data efficiently and accurately using machine learning as compared to other methods without much processing power. We first experiment with

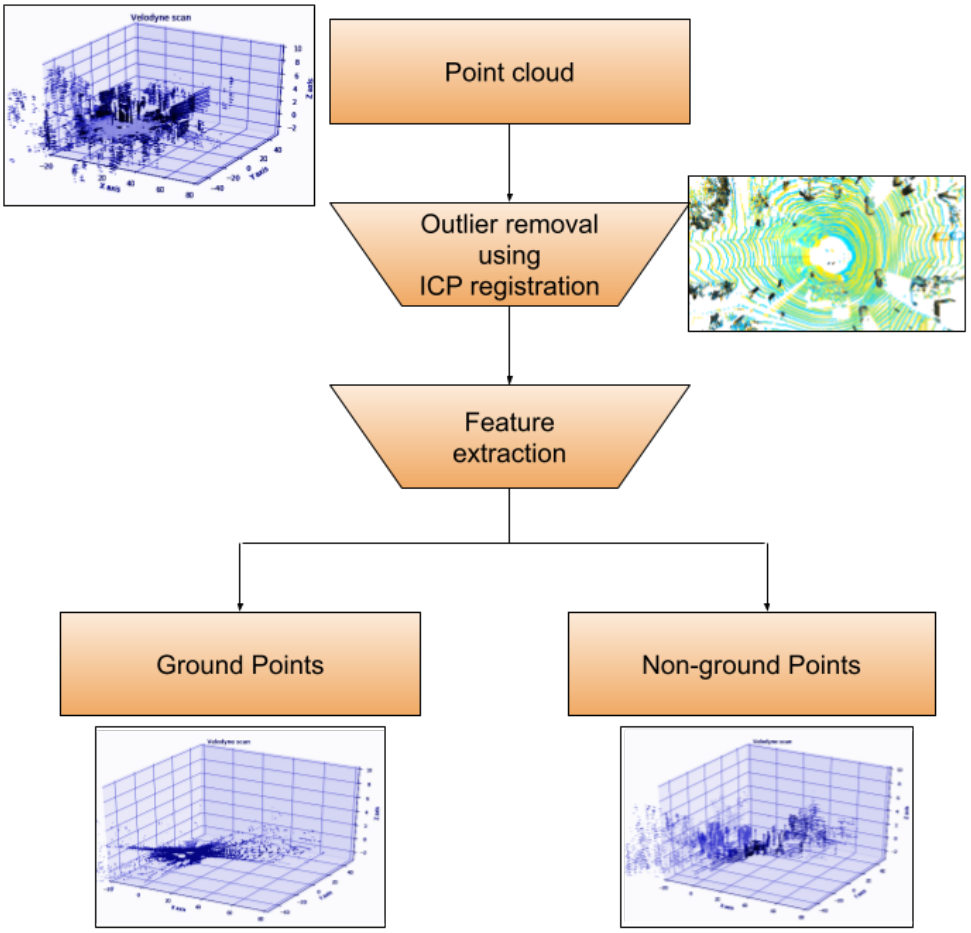


Figure FC3.1: Summary of our proposed classification system, for ground and non-ground classification. Our system includes two significant intermediate processes of feature extraction and registration.

the simple height filter approach for ground and non-ground points classification. To improve the classification results we utilize the neighborhood search on the point cloud. We compare the LGD (Local Shape Descriptor) for different neighborhood searches on the point cloud to identify what type of neighborhood would be suitable to generate features for ground and non-ground classification. The ground surface on which a vehicle can travel, which is mostly the road surface, is referred to as *navigable/drivable* space. The "ground" point class covers various fine-grained classes, such as "road," "parking," "sidewalk," "terrain," and so on [34]. A summary of ground and non-ground classification is shown in Figure FC3.1.

3.1 Height Filter Based Classification

A simple geometrical consideration may provide us with the means to greatly simplify or even automate the processing of ground point filtering. The ground surface is actually a collection of points that form a geometrical pattern of the plane with no vertical component. The point cloud consists of a large number of points. It is not enough to get to small details for basic tasks from such a large number of points. The ground points do not have a vertical component as opposed to the non-ground points. Utilizing this fact we use a simple height filter to classify the ground and non-ground points.

The data is separated into ground points and non-ground points based on the height filter. We analyze the SemanticKitti [2] point cloud data and observe that the ground points have z values (values in a vertical direction) below or close to 0. The range of z-values approximately is [-27.7, 3.02]. Based on this height filter is set to (max(z)-min(z))/100 . The normalization factor/value is chosen as '100' as the z values have a variation of 0.01 and also it sets the height filter close to 0 (around 0.03).

The points are separated into ground and non-ground points using the height (z-value) threshold. In some regions where the terrain is rugged the vehicle will experience changes in pitch, roll, and suspension when driving and cause the uneven distribution of 3D LiDAR data, due to which the ground points could be higher than the z-threshold in some regions. To address the issue of misclassifying ground points as non-ground, we divide x-y plane containing the entire point cloud into cells. A region is defined by the subset of the point cloud contained in a cell through the entire height of the point cloud. We determine the minimum z-value in each such region and use the sum of the minimum z-value and the z-threshold value as the new z-threshold value for the region. Thus, we use an adaptive height filter to identify ground points.

3.2 Analysis of Local Geometric Descriptors

The structure of the local neighborhood is identified using local geometric descriptors [63]. The covariance tensor is used as the local geometric descriptor in this case. The descriptor is derived from a local neighborhood. In general, many techniques can be used to locate nearby neighborhoods for points in a 3D point cloud. These neighborhood types, such as a spherical neighborhood with a radius, a spherical neighborhood with the number of closest neighbors in relation to the Euclidean distance in 3D space, or a hybrid search of a neighborhood with the maximum k-nearest neighbors inside a given radius, are used in particular. These neighborhood types are parameterized with a single scale parameter which is represented by either a radius or the number of nearest neighbors, and they allow describing the local 3D structure at a specific scale. Multi-scale neighborhoods can be used to represent the local 3D structure at several scales instead of utilizing a single neighborhood to describe it at one scale.

A k-d tree (short for k-dimensional tree) is a data structure that organizes points in a k-dimensional space [64]. k-d trees are a useful data structure for a variety of applications, including multidimensional search key searches (e.g. range searches and nearest neighbor searches). The k-d tree is used to perform three different searches for each point in the point cloud. The four searches include :

- Radius search: For each point P, it searches the neighbor points in the given radius with point P at its center.
- KNN search: For each point P, it searches k-nearest neighbor points with given k.
- Hybrid search: It returns at most k-nearest neighbors that have distances to the search point less than a given radius.
- Optimal search: We use a multi-scale neighborhood and find the optimal scale for

each point based on homogeneity score and Shannon entropy [65].

For each neighborhood type, we extract a set of geometric features describing the spatial arrangement of points within the local neighborhood of each considered point. The neighborhood shape can be classified into “line-”, “surface-” and “point-” type features, which are the geometric classes [66]. Each point is represented by the local 3D shape features given by linearity, sphericity, and planarity. For each point, the covariance tensor is generated considering variance in the points lying in the neighborhood of the search point. Next, the eigenvalue decomposition is performed for all the covariance tensors generated for each point. The likelihood of the point belonging to these classes, C_l, C_s, C_p, respectively, are computed using eigenvalues of the descriptors at multiple scales [66]. The tuple C_l, C_s, C_p, called as saliency map, gives the probabilistic geometric classification of the point [66].

Every point can be defined by a combination of any two of the barycentric coordinates, C_l (linear metric), C_s (spherical metric), or C_p (planar metric), which are calculated from the eigenvalues using following equations:

$$C_l = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2 + \lambda_3}$$

$$C_s = \frac{3\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3}$$

$$C_p = \frac{2(\lambda_2 - \lambda_3)}{\lambda_1 + \lambda_2 + \lambda_3}$$

Shannon entropy [65], H, is calculated for each point based on C_l, C_s, C_p values as:

$$H = -C_l \log_2(C_l) - C_s \log_2(C_s) - C_p \log_2(C_p)$$

We analyze the probability density function (PDF) for each of the geometric feature values for each class based on different neighborhoods for different frames. Based on

our analysis of PDFs and k-distributions for all classes, objects like trunks, traffic-sign, buildings, and poles with smaller radial neighborhood values around 1 & 2 and k-nn values around 50 & 100 gave somewhat expected characteristics like high C_l value for trunk, traffic sign, and k-distributions for optimal entropy values and high homogeneity values were also within the afore-mentioned range for these objects. Thus, we choose radius as 1 and k-nn as 100 as ideal for the neighborhood to get appropriate height features for all classes and which will help in determining better neighborhoods for ground and non-ground classification, including correct classification of smaller objects like a trunk, traffic sign, etc. to non-ground class. Example of PDF plots with different neighbourhood size for class car, road and trunk are as shown in Figures FC3.2, FC3.3, FC3.4. The PDF of road class shown in FC3.3 shows characteristics of high C_p indicating a planar neighborhood and PDF of trunk class shown in FC3.4 shows high C_l values, whereas PDF of car shows mix of C_s and C_p characteristics. These properties are more noticeable with a hybrid neighborhood of radius 1 and k-nn 100.

The point cloud is classified into "ground" and "non-ground" points for ground point detection. $\mathbf{S_1}$ involves two sequential substeps, namely, outlier removal and semantic classification. Here, we exploit the temporal and spatial locality of the points.

3.2.1 Outlier Removal

Point cloud registration or scan matching is done for consecutive frames to find a spatial transformation (e.g., scaling, rotation and translation) in order to align two point clouds. Rigid transformation refers to the transformation that does not change the distance between any two points and consists of translation and rotation, unlike non-rigid transformation, which consists of scaling and shear mapping. The iterative closest point (ICP) algorithm [67] is used in this work which performs rigid registration (yielding rigid transformation) in an iterative fashion by alternating in the initial transformation,

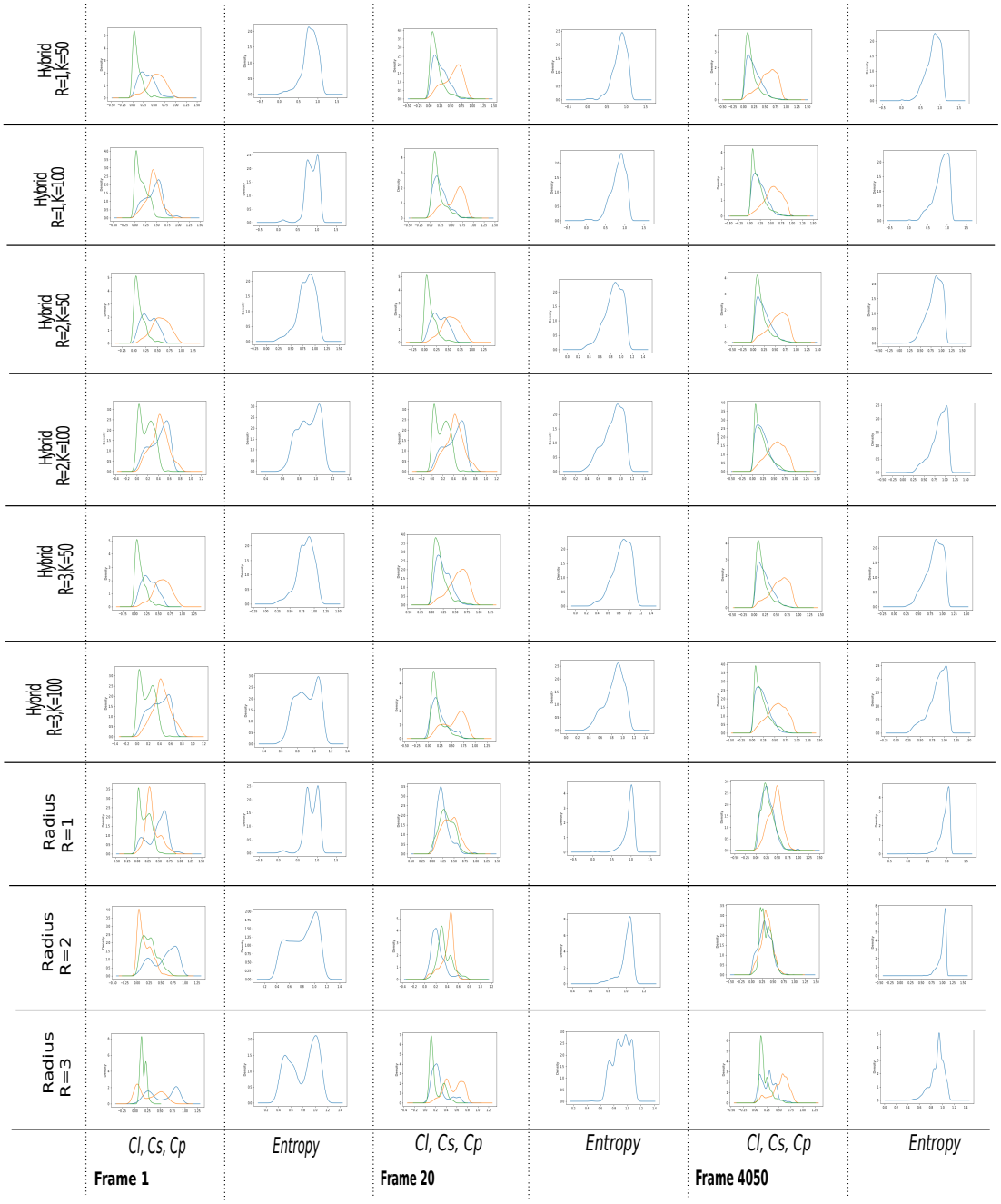


Figure FC3.2: PDF of car C_l, C_s, C_p, Entropy values based on different neighbourhoods for 3 different frames of sequence 08 indicating high density with C_l and C_p plots. Blue, Green, and Orange line plots show the C_l, C_s, C_p values, respectively.

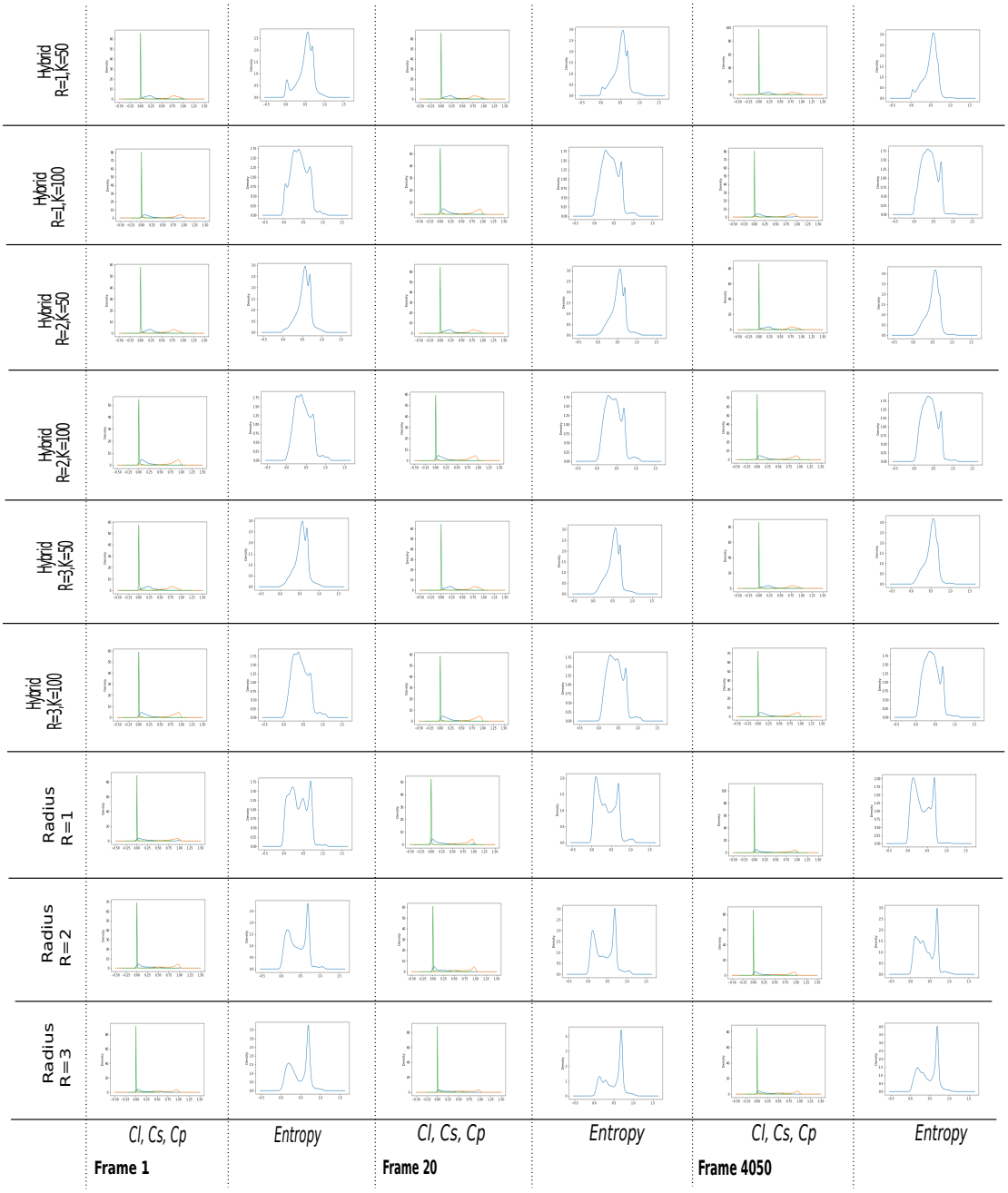


Figure FC3.3: PDF of road C_l, C_s, C_p, Entropy values based on different neighbourhoods for 3 different frames of sequence 08 indicating high density with C_l and C_p plots and minimal density with C_s value around 0. Blue, Green, and Orange line plots show the C_l, C_s, and C_p values, respectively.

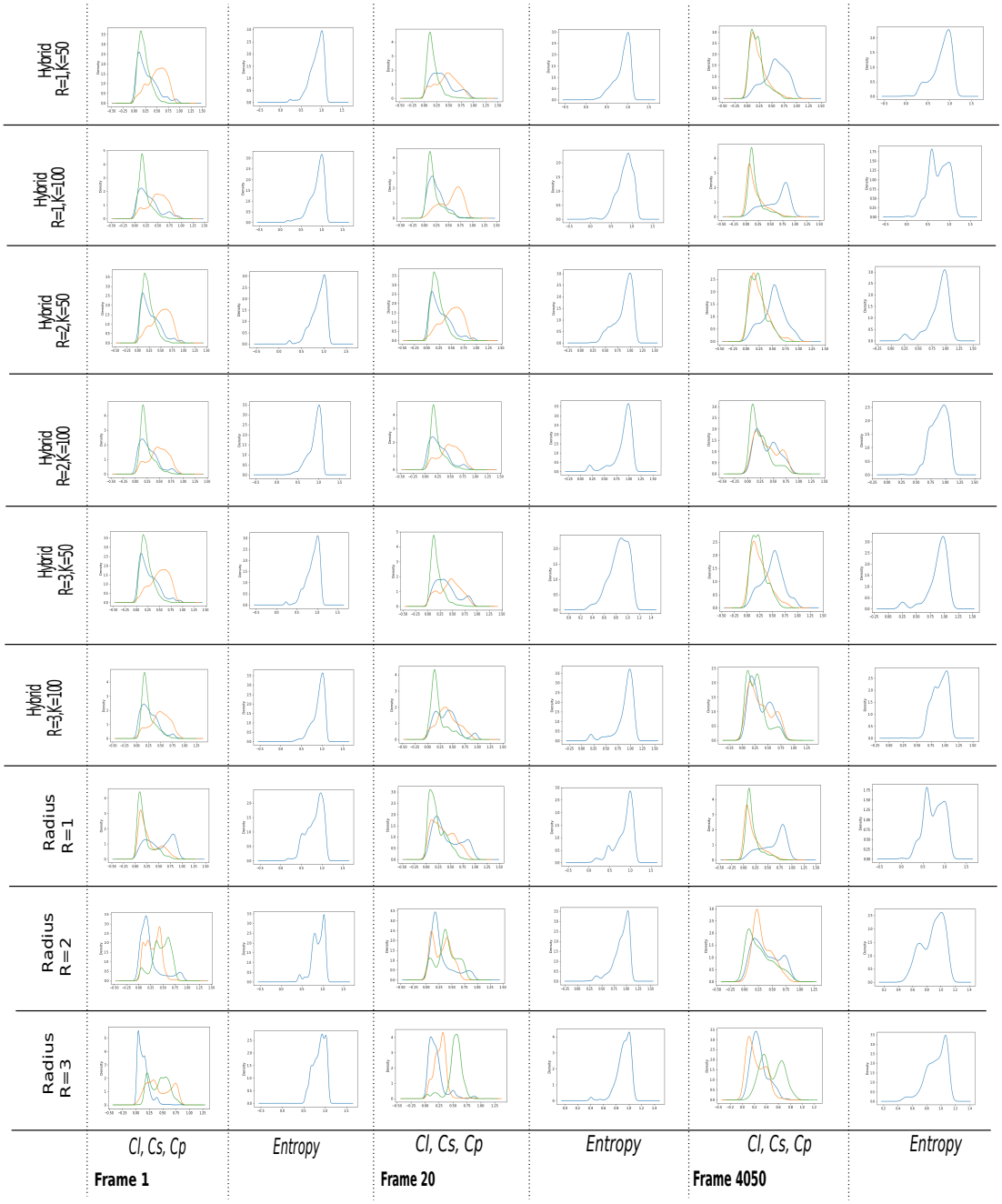


Figure FC3.4: PDF of trunk C_l, C_s, C_p, Entropy values based on different neighbourhoods for 3 different frames of sequence 08 indicating high density with C_l and C_s plots. Blue, Green, and Orange line plots show the C_l, C_s, and C_p values, respectively.

finding the closest point in the source point cloud for every point in the target point cloud. A correspondence pair denotes the existence of an affine transformation (e.g., scaling, rotation, and translation) that transforms a point in the source point cloud to another point in the target point cloud.

Different variants of ICP use different objective functions $E(T)$ for finding the optimal rigid transformation T. The most common is point-to-point ICP which uses least squares as the objective function:

$$E(\mathbf{T}) = \sum_{(\mathbf{p},\mathbf{q})\in\mathcal{K}} \|\mathbf{p} - \mathbf{T}\mathbf{q}\|_2^2. \tag{Eqn 3.1}$$

Another variant of ICP is the point-to-plane one which uses the following objective function:

$$E(\mathbf{T}) = \sum_{(\mathbf{p},\mathbf{q})\in\mathcal{K}} \big((\mathbf{p} - \mathbf{T}\mathbf{q}) \cdot \mathbf{n_p}\big)^2, \tag{Eqn 3.2}$$

where $\mathbf{n_p}$ is normal of point $\mathbf{p}$. The point-to-plane ICP algorithm has been proven to be a faster convergence speed than the point-to-point ICP algorithm [68]. Hence, we choose the point-to-plane ICP algorithm for registration.

To improve the registration outcomes, a multiway approach is beneficial. The practice of aligning many bits of geometry in a global location is known as multiway registration. Here, we use multiway registration, which internally uses an ICP registration with pose graph optimization [69].

The registration uses temporal locality, *i.e.* points in a frame must be preserved in consecutive frames. With the idea that data points in the consecutive frames should be close in the spatial domain, we find corresponding pairs of points in consecutive frames. The ICP registration is done on the point set. Registration is done based on the maximum correspondence distance i.e. distance between two sets of points. Following registration, we obtain the correspondence set consisting of pair of points between

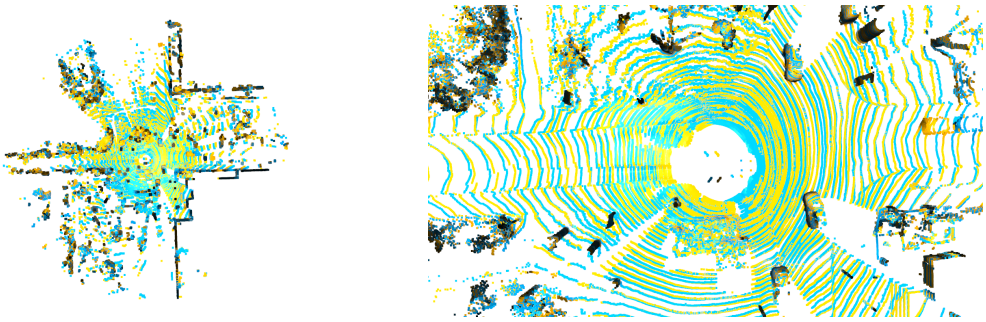


Figure FC3.5: Point cloud from SemanticKITTI dataset [2] after multiway registration using point-to-plane ICP algorithm. Points in yellow and blue represent consecutive point clouds at the current and previous frames respectively. The left image is the 3D top view of the point cloud and the right image is the zoom-in 3D top view of the point cloud.

pair of frames. The points which are not present in the correspondence set after the registration are considered outlier points. We discard those points and then perform the classification. Owing to the continuity of motion across frames, experiments are performed on iterative registration in different combinations of consecutive frames for outlier removal. The registration of points is shown in Figure FC3.5.

3.3 Random Forest Classifier

We improve the classification based on a height filter to separate ground and non-ground points by training a Random Forest Classifier (RFC) [24] model with a set of relevant features. We extract the height-related features and feed them to the RFC to test the classification accuracy between the ground and non-ground points. We also experimented by adding the local height features extracted based on the hybrid neighborhood search in both scenarios, *i.e.* at single scale and at multiple scales.

3.3.1 Semantic Classification

The elevation (z) of several of the objects corresponding to non-ground locations is higher than the ground. As a result, height-based features are optimal for distinguish-

Table TC3.1: Point-wise features at each scale for ground detection ($\mathbf{S_1}$)

Local Features	**Global Features**	**Global Features**
Point-based		Frame-based
Height-based – Difference from max. – Difference from mean – Standard deviation	*Height-based* – Value (z-coordinate) – Difference from mean (*of frame*) *Position-based* – Distance from sensor – Elevation angle θ	*Height-based* – Difference from mean – Standard deviation

ing ground locations from others, according to [22]. The "ground" class is a coarser class that combines numerous fine-grained semantic classes relevant to the ground. We extract local and global spatial handmade features and utilize them to divide the point cloud into the ground and non-ground classes using the RFC.

The feature extraction is implemented on the point cloud in each frame, after outlier removal. Here, we compute multi-scale local height features, for three scales. Multi-scale features, *i.e.* features captured at different spatial resolutions, are known to work better than a single scale for LiDAR point classification using RFC [70]. Here, for each point, we select a hybrid neighborhood search that combines the criteria of the spherical and the k-nearest neighborhoods (knn). Thus, we identify at most k-nearest neighbors (knn) of a point that are within a given distance, r, from the point. The height features used for ground point detection are listed in Table TC3.1. These extracted features are computed and used in an RFC for both training and testing.

3.4 Experiments & Results

Our method requires an input dataset with enough annotations to generate machine learning solutions. In that regard, SemanticKITTI [2] serves well as our test data.

3.4.1 Datasets

The SemanticKITTI dataset [2] has been published primarily for three benchmark tasks, namely semantic classification and scene completion of point clouds using single and multi-temporal scans.

The SemanticKITTI dataset comprises over 43,000 scans of which over 21,000 are training sequences with IDs from 00 to 10. We have used sequence 08 as the validation/test set, as prescribed by the data providers, thus training our model on the remaining training sequences for our classifier model, *i.e.* RFC model for ground point detection. We only used every 10^{th} frame of the SemanticKITTI sequences because the frame-wise data is acquired every 0.1 seconds, and our subsampling ensures that significant variations in the scene are captured without incurring high computing costs.

Overall, the dataset has annotations for 28 distinct classes for the semantic classification benchmark task. We consider five such classes, namely "road," "parking," "sidewalk," "other ground," and "terrain," together as the "ground" class in our approach. Thus, the "non-ground" class implies the remaining classes, *i.e.* movable objects, such as "car," "bicycle," etc., and stationary objects, such as, "building," "vegetation," etc.

We test our method on another dataset called the nuScenes dataset [5], which has been used for a variety of benchmarks such as LiDAR segmentation, object detection, and tracking, etc. This dataset contains 1000 scenes from 15 hours of driving data with 20-second long sequences for training, validation, and testing (700, 150, 150). The dataset is diverse, with data from two cities, left versus right-hand traffic, interesting driving maneuvers, common traffic situations, and unexpected behavior. nuScenes' 32-lane LiDAR generates approximately 30k points per frame.

We test on the mini dataset which is a subset of trainval consisting of 10 scenes. The nuScenes-lidarseg dataset includes annotations for 32 different segmentation tasks.

We combine the classes to form ground, which includes “driveable surface,” “other flat,” “sidewalk,” “terrain,” and non-ground, which includes “barrier,” “bicycle,” “bus,” “car,” and “construction vehicle,” “motorcycle,” “pedestrian,” “traffic cone,” “trailer,” “truck,” “vegetation,” as we did with the SemanticKITTI dataset for the LiDAR classification.

3.4.2 Experimental Setup

For multi-scale feature extraction for ground point detection using RFC, hybrid criteria for neighborhood determination have been used for three different scales. We have commonly used the constraint of r of $1m$ for the spherical neighborhood in all the scales, and variable k values for the knn neighborhood, *i.e.* $k = 50, 100, 200$ neighbors. We have systematically experimented with several combinations of neighborhood criteria to arrive at this parameter setting as described earlier.

We have used all sequences from the training dataset for training except sequence 08 which is used for validation for SemanticKITTI dataset while for nuScenes we use 8 sequences for the training and 2 sequences for validation from the mini dataset. The outlier removal using registration and geometric feature computation takes approximately $\sim$11 seconds and $\sim$122 seconds respectively per frame. Training the RFC model for complete data takes approximately $\sim$2 hours and inference time after feature extraction takes $\sim$20 seconds. The computation time stated are based on the approach implemented on a system with an Intel core i7 CPU with 12 GB of RAM.

3.4.3 Results

For our workflow, we perform both qualitative analysis using visualization and appropriate quantitative evaluation. For our initial approach of classification using a height filter, we obtain an accuracy of 76.82%. We also perform experiments with different feature selections and outlier removal.

3.4.3.1 Feature Selection

To analyze the results we carried out 10 different experiments by including and excluding different features and their impact on the results. We also extracted additional multi-scale local height features by considering hybrid search with radius: 1, k-nearest neighbor: 50 and hybrid search with radius: 1, k-nearest neighbor: 200 to the existing local feature set with hybrid search with radius: 1, k-nearest neighbor: 100. We use this additional features as instead of relying on a single scale however appropriate, multi-scale features would provide additional information. We also added remission and azimuth angle as features if they provide any improvement. The results of these experiments are shown in Table TC3.2.

From the above experiments, we can observe that the highest accuracy is obtained with Exp 9 with multi-scale features added. This feature set in Exp 9 also coincides with the feature set mentioned in Table TC3.2. The remission feature shows the least importance for all the experiments included. The inclusion of multi-scale features shows improvement in accuracy; however, including all features decreases the accuracy, possibly due to overfitting the data. For the next set of experiments, we consider the feature set mentioned in Exp 9.

Next, we experiment with different scenarios along with the outlier removal process done before the classification and feature set mentioned in Exp 9 and Exp 10 in Table TC3.2. At each frame, we perform iterative registration using three consecutive frames, *i.e.* x, (x-1), and (x-2), where x is the current frame. x is registered with respect to (x-1), and again, (x-1) with respect to (x-2). Thus, two sets of registrations are performed. Points preserved from the first set of registration are considered as input for the next registration step. The points finally preserved after outlier removal using two-step registration are considered for the classification step. Results of different registrations experiments are in Table TC3.3. The highest accuracy *i.e.* 96.91% results of classifica-

tion are obtained with the multi-scale feature set and registration and the same setting is used as the initial step for our next perception task of road surface extraction described in Chapter 4.

The classification report with Precision, Recall, F-1 score, and IoU score for the highest accuracy *i.e.* 96.91% results of ground and non-ground points obtained with the multi-scale feature set and registration are shown in TableTC3.4. The micro-average precision, recall, and F1 score is calculated from the individual classes' true positives (TPs), true negatives (TNs), false positives (FPs), and false negatives (FNs) of the model. The macro-average precision, recall, and F1 score is calculated as an arithmetic mean of individual classes' precision and recall scores.

Figures FC3.6 and FC3.7 show the ground segmentation. Although GndNet [34] has reported a mIoU of 83.6%, this number is not comparable because their mIoU was determined by combining the ground and non-ground classes. Similar to this, the ground segmentation in [22] reported a mIoU of 78.46% for the "ground" class, but this is not comparable because their "ground" class also includes "vegetation". These mIoU scores suggest that, despite the fact that we cannot directly compare, our method for classifying ground points exhibits a really high level of accuracy.

We also tested our approach on the mini subset of nuScenes dataset with the same multi-scale feature settings of radius of 1m and k-nn with $k = 100$, along with registration-based outlier removal, which yielded better results on the SemanticKITTI dataset. Table TC3.5 displays the classification report and IoU score of ground and non-ground points from the mini subset of nuScenes [5]. We can see that the classification results for the nuScenes dataset show high accuracy.

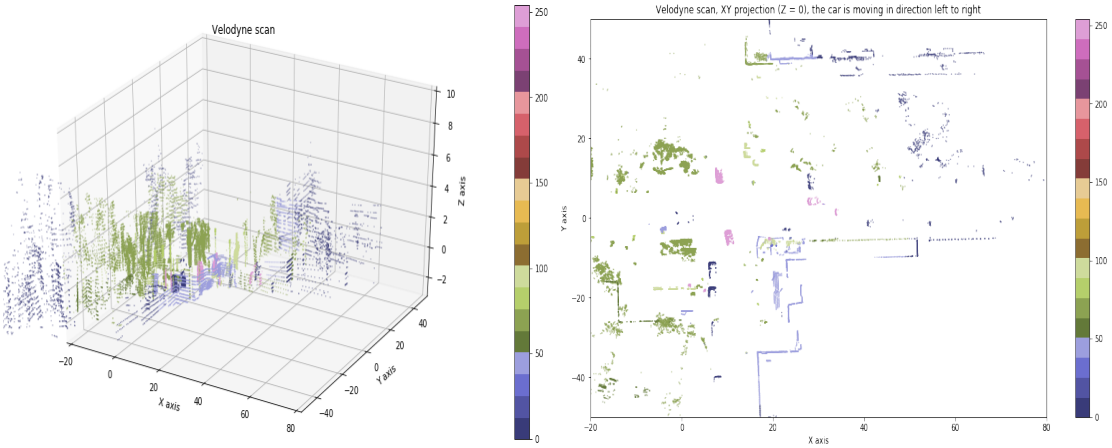


Figure FC3.6: Non-Ground points obtained after classification for frame 20 of sequence 08 of SemanticKITTI dataset [2]. Points are colored based on semantic labels in (Left) the 3D view of the point cloud, and (Right) the 2D top view in the x-y plane of the point cloud.

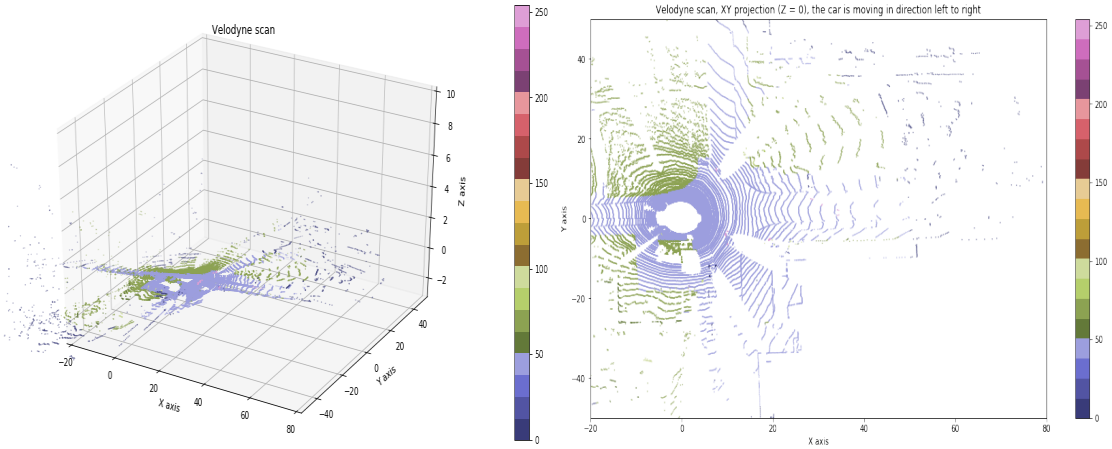


Figure FC3.7: Ground points obtained after classification for frame 20 of sequence 08 of SemanticKITTI dataset [2]. Points are colored based on semantic labels in (Left) the 3D view of the point cloud, and (Right) the 2D top view in the x-y plane of the point cloud.

Table TC3.2: Experimental results for ground and non-ground classification on SemanticKITTI dataset [2]

Index	**Remission**	**Azimuth Angle ϕ**	**Variance of z-value**	**Common Frame-wise Features**	**Multi-scale Point-wise Features**	**Accuracy (%)**
Exp 1	*No*	*No*	*Yes*	*Yes*	*No*	*96.38*
Exp 2	*No*	*Yes*	*No*	*No*	*No*	*95.65*
Exp 3	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*95.20*
Exp 4	*Yes*	*Yes*	*No*	*No*	*Yes*	*96.58*
Exp 5	*Yes*	*Yes*	*No*	*No*	*No*	*95.46*
Exp 6	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*96.17*
Exp 7	*No*	*Yes*	*Yes*	*Yes*	*No*	*95.07*
Exp 8	*No*	*No*	*Yes*	*Yes*	*Yes*	*96.60*
Exp 9	*No*	*No*	*No*	*Yes*	*Yes*	***96.63***
Exp 10	*No*	*No*	*No*	*Yes*	*No*	*96.37*

Common point-wise features in all experiments at single scale:
1. z coordinate value **2.** range value **3.** elevation angle θ
4. Height difference (local) : max(local z coordinates) - z coordinates(at a point)
5. Mean difference (local) : z coordinates(at a point) - mean(local z coordinates)
6. Variance (local) : var(local z coordinates)
7. Standard deviation : standard deviation(local z coordinates)
Common frame-wise features in experiments :
1. Mean difference : z coordinate value - mean (all z coordinates)
2. Standard deviation : standard deviation(all z coordinates)
Multi-scale point-wise features :
Below features are added for 3 different neighbor searches:
Hybrid search with r - 1 & k - 100 (Added in first model)
Hybrid search with r - 1 & k - 50 (Added for experiments 1 to 8)
Hybrid search with r - 1 & k - 200 (Added for experiments 1 to 8)

Table TC3.3: Results of our classification model with and without registration process on SemanticKITTI dataset [2]

Points from Point Cloud considered for Classification	**Feature Set**	**Registration of Point Clouds**	**Accuracy of RFC model**
All points	1 set of hybrid neighbours -based local features	No registration performed	96.37%
All points	3 set of hybrid neighbours -based local features	No registration performed	96.63%
Points preserved between registrations	1 set of hybrid neighbours -based local features	Registration 1 : Frame x, frame x-1 Registration 2 : Frame x, frame x-2	96.53%
Points preserved in consecutive registrations	1 set of hybrid neighbours -based local features	Registration 1 : Frame x-1, frame x-2 Registration 2 : Frame x, frame x-1	96.58%
Points preserved in consecutive registrations	3 set of hybrid neighbours -based local features	Registration 1 : Frame x-1, frame x-2 Registration 2 : Frame x, frame x-1	**96.91%**

Table TC3.4: Classification report with Precision, Recall, F1-score, Support, Accuracy and IoU for SemanticKITTI dataset [2]

	Precision	**Recall**	**F1-score**	**Support**	**IoU(%)**
Non-Ground	0.93	1.00	0.97	27560594	94.7%
Ground	1.00	0.91	0.95	21618194	93.0%
Micro Average	0.96	0.96	0.96		
Macro Average	0.97	0.95	0.96		
Weighted Average	0.96	0.96	0.96		
Overall Accuracy(%): 96.91%					

Table TC3.5: Classification report with Precision, Recall, F1-score, Support, Accuracy and IoU for nuScenes dataset [5]

	Precision	**Recall**	**F1-score**	**Support**	**IoU(%)**
Non-Ground	0.957	0.954	0.955	1331065	91.4%
Ground	0.943	0.947	0.945	1080143	89.58%
Micro Average	0.95	0.95	0.95		
Macro Average	0.951	0.951	0.951		
Weighted Average	0.951	0.951	0.951		
Overall Accuracy(%): 95.07%					

3.5 Summary

We have proposed and implemented ground and non-ground classification, an alternative system for classification which works for all scenarios, rough surfaces, and even on different datasets for 3D automotive LiDAR point clouds. We analyze the neighborhood of the points using the local geometric descriptor technique to determine an appropriate neighborhood for feature extraction. We also experimented with the outlier removal process and multi-scale feature extraction, which improved the results. Supervised learning using RFC is used for ground and non-ground classification. Our experiments on SemanticKITTI and nuScenes yielded promising results, which have been qualitatively and quantitatively verified.

CHAPTER 4

ROAD SURFACE EXTRACTION USING 3D LIDAR POINT CLOUD SEQUENCES

Robotics and intelligent vehicle technology face a difficult problem called navigable space detection, which calls for a combined approach of computational geometry and computer vision. After the initial ground classification in this Chapter, we focus on the road surface extraction challenge which utilizes the ground points from classification as described in Chapter 3. In processing automotive LiDAR point clouds in three dimensions (3D), the ground surface on which a vehicle can travel, which is mostly the road surface, is referred to as *navigable* space. The "ground" point class covers various fine-grained classes, such as "road," "parking," "sidewalk," "terrain," and so on. [34]. The state-of-the-art methods perform ground point segmentation/detection followed by ground plane estimation motivated as a precursor to road surface extraction [34,41]. However, even for a single point cloud, we find that ground plane estimate must be done piece-wise, resulting in a coarse approximation of the surface geometry. Piece-wise estimation necessitates a systematic geometric study to establish the number, position, and orientation of planes required to form a watertight surface when used together. This is a difficult point-set processing task, especially given the unstructured point clouds. Instead, we propose extracting the surface mesh directly from the road points. However, creating a fine mesh with all of the road points takes time. This can be alleviated by selecting a selection of road sites that adequately sample the surface.

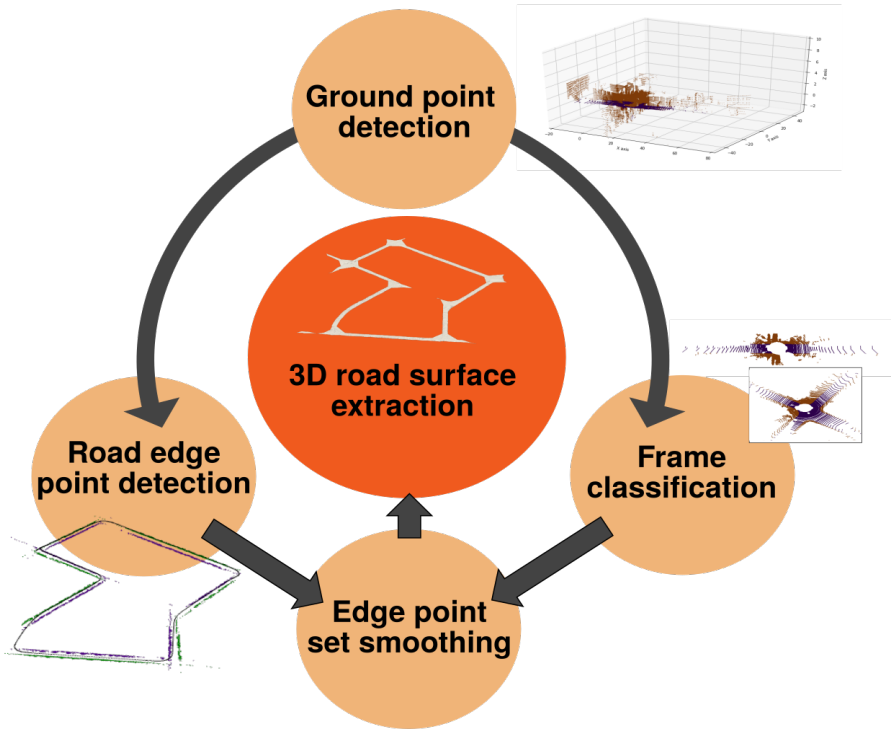


Figure FC4.1: Summary of our proposed system [1], road surface extraction, for 3D road surface extraction using ground points detected from an automotive LiDAR point cloud sequence. Our system includes two novel and significant intermediate processes of road edge point detection and frame classification.

Here, we propose the road edge points and vehicle positions as this desired sample set.

Given we are using the positions of the ego-motion of the vehicle, we can now expand the surface extraction across all frames in a sequence. This leads to creating a watertight road surface for the entire sequence, which is as seen from the point of view of the ego-vehicle. Such a process requires all the point clouds in the sequence, which improves the utilization of the complete dataset.

Semantic segmentation, which is used in the traditional data processing workflow for 3D automotive LiDAR point clouds, is capable of quickly identifying road points. However, the semantic segmentation results have to be post-processed to identify curb or edge points [50]. At the same time, ground point filtering using local height differences is a reliable solution in the LiDAR point cloud analysis [22]. Binary clustering in

point clouds can be done here by classifying the points as “edge” or “non-edge” using highly statistically significant handmade features such as height differences. Since it was discovered that the expectation-maximization (EM) approach is useful for binary grouping of LiDAR point clouds, [71], we propose a road edge point detection method using binary clustering of ground points. We also use the spatio-temporal locality of the points for outlier removal to improve ground point detection.

In the presence of turnings and complex topology, such as junctions, extracting road geometry becomes difficult. We start with the workflow for straight roads because our work is unique in terms of obtaining surface mesh geometry for roads. This mesh generating procedure can then be used for curved roads, such as turns and junctions. Alternatively, our suggested method may extract contiguous parts of straight roadways and use surface correction to fill the gaps between them for short curved segments. For most sequences with a significant number of contiguous straight highways, this technique works. The point cloud geometry for each frame must be classified in order to meet our criteria for recognizing consecutive straight roadways. We propose a novel frame-wise point cloud geometry classification, referred to as *frame classification*, using an appropriate image representation of the geometry. We choose an intermediate image representation specifically, as is done in state-of-the-art deep learning classifiers for semantic segmentation [35]. Here, transfer learning is used for frame classification.

Thus, our proposed approach is to detect ground points, on which both edge detection and frame classification are performed. We further smooth the edge point set to improve the sample set for surface mesh generation and finally extract the road surface using geometry algorithms. A system based on this approach [1] is implemented as shown in (Figure FC4.1). Our novel contributions are in integrating appropriate methods in our proposed system, for road surface extraction from LiDAR point cloud Sequence for its implementation (Figure FC4.2).

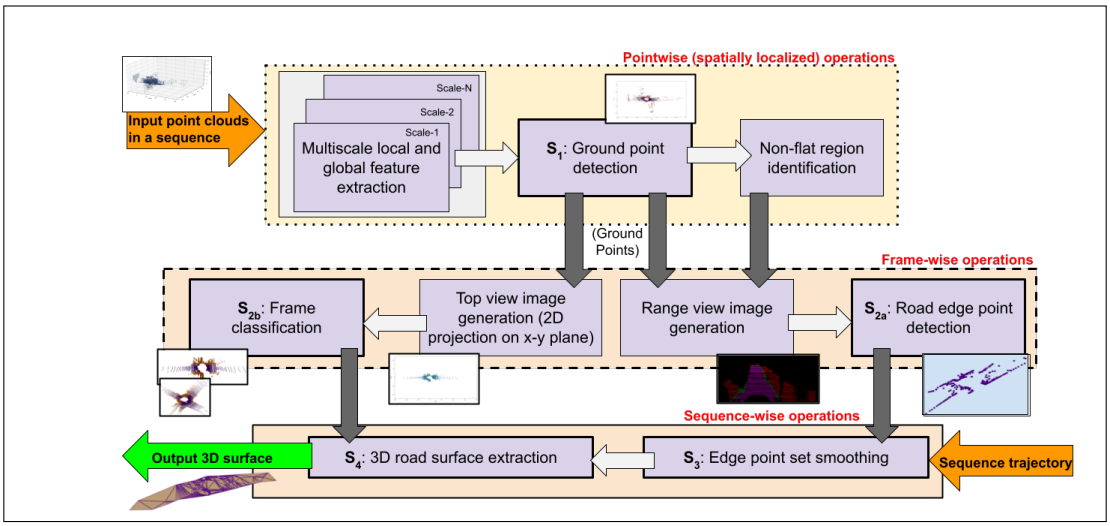


Figure FC4.2: Our proposed workflow of road surface extraction [1], for generating 3D road surface, from input 3D LiDAR frame-wise point clouds in a sequence and its trajectory information (position and pose of the vehicle). Our workflow proceeds from point-, frame-, to sequence-wise processing.

We propose a novel workflow to extract approximate road geometry, for straight roads. The workflow of road surface extraction consists of five key steps, namely, ($\mathbf{S_1}$) ground point detection, ($\mathbf{S_{2a}}$) frame classification implicitly giving the road geometry, ($\mathbf{S_{2b}}$) road edge point detection, ($\mathbf{S_3}$) edge point set smoothing, and ($\mathbf{S_4}$) 3D road surface extraction. As shown in Figures FC4.1 and FC4.2:

- $\mathbf{S_1}$ is a point-wise operation, *i.e.* it is implemented on each point in the point cloud, *i.e.* a frame.
- The frame-wise operations, $\mathbf{S_{2a}}$ and $\mathbf{S_{2b}}$, are decoupled and implemented in parallel.
- $\mathbf{S_3}$ and $\mathbf{S_4}$ are sequence-wise operations, and hence require the trajectory information of the vehicle for the entire sequence.

The overall workflow of road surface extraction, *i.e.* $\mathbf{S_1}$ to $\mathbf{S_4}$, is captured in Algorithm 2. The partial workflows of the point-wise classification process ($\mathbf{S_1}$) and the sequence-wise road edge extraction ($\mathbf{S_{2a}}$, $\mathbf{S_{2b}}$, $\mathbf{S_3}$) are given in Algorithms 1 and 3, respectively.

Our proposed road surface extraction gives the surface as visible from the point-of-view of the vehicle. Hence, the vehicle is called an *ego-vehicle* [41].

4.1 $\mathbf{S_1}$ – Ground Point Detection

The point cloud is classified into "ground" and "non-ground" points for ground point detection. $\mathbf{S_1}$ involves two sequential substeps, namely, outlier removal and semantic segmentation. Here, we exploit the temporal and spatial locality of the points. This step is the classification process described in Chapter 3.

4.1.1 Outlier Removal

The Iterative Closest Point (ICP) registration [67], method as described in Chapter 3 is used to do point cloud registration or scan matching on two separate point clouds to determine the correspondence pairs of points between the two.

The registration process relies on temporal locality, which means that points in one frame must remain in succeeding frames. As a result, we identify point correspondence pairings in successive frames and designate the remaining points as "outliers" that need to be filtered out. Iterative registration is used on three successive frames at a time due to the consistency of motion across frames. In two steps, we conduct registration for a given current frame x. Using registration between frames $(x-1)$ and $(x-2)$, outliers are taken out of frame $(x-1)$ in the first phase. After executing registration between (x) and $(x-1)$, the identical procedure is then repeated on frame (x).

4.1.2 Semantic Segmentation

The elevation (z) of several of the objects corresponding to non-ground locations is higher than the ground. As a result, height-based features are optimal for distinguishing ground locations from others, according to [22]. The "ground" class is a coarser class that combines numerous fine-grained semantic classes relevant to the ground. We extract local and global spatial handmade features and utilize them to divide the point cloud into the ground and non-ground classes using the Random Forest Classifier (RFC) [24].

After removing outliers from the point cloud in each frame, feature extraction is applied. Here, we compute three-scale multi-scale local height features. For LiDAR point classification using RFC, multi-scale features, *i.e.* features recorded at various spatial resolutions, are known to perform better than a single scale [70]. Here, for each point, the same neighborhood with a hybrid search that combines the criteria of the spherical and the k-nearest neighborhoods (k-nn) is used. The height features used for ground point detection are the same as listed in Table TC3.1. These extracted features are computed and used in an RFC for both training and testing.

4.2 S_{2a} – Road Edge Point Detection

The ground points that physically interface with the curb or sidewalk are those spots along the borders of the road [50]. Both the left and right banks of the road must be extracted for road surface extraction. Edge detection is a well-researched issue in image processing where gradient information is used to identify image borders. It is typically accomplished using the three-step method of differentiation, smoothing, and labelling. [72]. The height gradient is employed as a distinguishing feature to pinpoint the places on road boundaries using the same strategy as in image processing.

Input : Point cloud $P(f)$ at a frame f
Output: Set of ground points $G_p(f)$

$G_p(f) \leftarrow \{\}$ `// Set of ground points`
for *point p in point cloud* $P(f)$ **do**
  `// Extraction of all features,`
  `// as given in Table TC3.1`
  for $0 \leq i < n_{scales}$ **do**
    $N_i \leftarrow$ find-local-neighborhood(p, *neighborhood_size*)
  end
  $F_{\text{p}} \leftarrow$ compute-features ($P_i, N_1, \ldots, N_{n_{scales}}$)

  `// Classification of points as`
  `// ground or non-ground points`
  $type(p) \leftarrow$ classify-using-Random-Forest-Classifier(F_{p})

  `// Add ground points to the output`
  if $type(p)$ *is "ground"* **then**
    $G_p(f) \leftarrow G_p(f) \cup \{p\}$
  end
end
return $G_p(f)$

Algorithm 1: Ground point detection per frame, *i.e.* $\mathbf{S_1}$

However, our road edge identification method differs from picture edge detection in two significant respects. To begin, unlike image smoothing used for edge detection in photos, smoothing is required only for road edge points, not the complete point cloud. On ground points, we execute edge smoothing and labeling procedures, which are now collectively referred to as *edge point set smoothing*. Secondly, in our case, the road extraction depends on the road geometry information, *i.e.* determined during $\mathbf{S_{2b}}$. Additionally, the edge point set smoothing $\mathbf{S_3}$ requires knowledge of the sequence trajectory for coordinate system translation, unlike the differentiation step, which is performed on a frame. As a result, the three steps do not follow successively, here. Thus, $\mathbf{S_{2a}}$ is solely for differentiation, while $\mathbf{S_3}$ implements edge point set smoothing and labelling.

4.2.1 Height Gradient-based Differentiation

Here, only the ground points are used to compute the first-order derivatives or gradients of height values. For this work, we recommend point clustering with two particular demands. First, we group road sites into "flat" and "non-flat" zones, which are described as having low and high height gradients, respectively. Second, height differences are used to compute the characteristics required for clustering. Of the custom characteristics deduced by hand-crafted features that are utilized to semantically segment 3D aerial and terrestrial LiDAR point clouds [70], we choose the two appropriate height-difference (Δ_z) features, namely, in a local neighborhood, and in the 2D accumulation map. The 2D accumulation map generates local neighborhoods of points projected to xy-plane, within a square of fixed length (*e.g.* 0.25m), centered at the point. For the clustering process, we observe that the clear clusters do not exist in vehicle LiDAR point clouds. We note that the vehicle LiDAR point clouds do not have distinct clusters for the clustering process. In these circumstances, it has been demonstrated that the Expectation-maximization (EM) algorithm [73] performs superior to the k-means clustering. The Figure depicts flat and non-flat regions identified using the k-means and EM algorithms across different road structures, indicating that the EM algorithm distinguishes between regions more precisely than the k-means algorithm. The basic presumption of the EM algorithm is the presence of a Gaussian Mixture Model (GMM) in the data. We then utilize the EM technique to identify two clusters of points that belong to the flat and non-flat regions, assuming a bimodal data distribution in the 2D feature space.

4.2.2 Projection to Range Images

The edge points are located in the non-flat zones, where the preferred edge points are those that are closest to the centerline or the trajectory. The range picture of the frame is

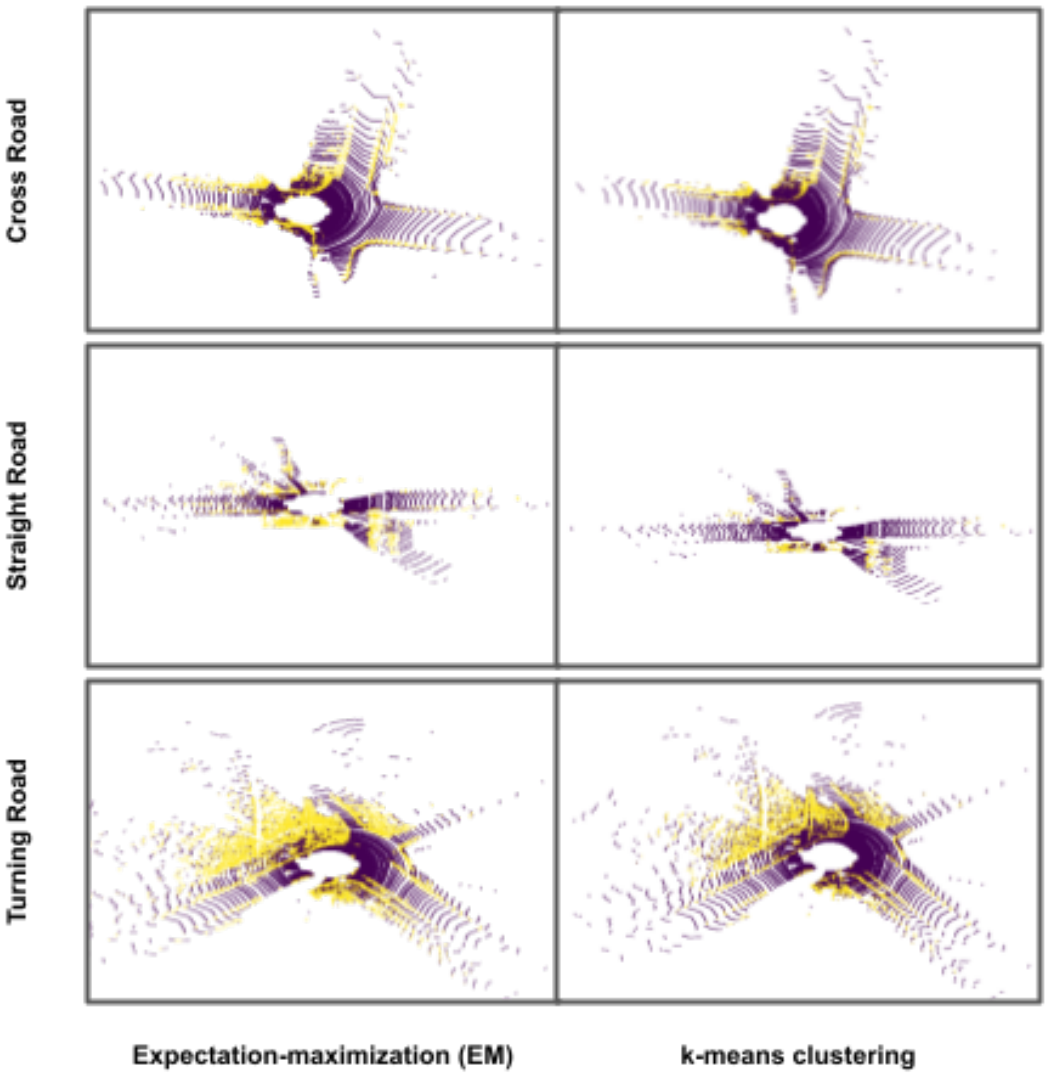


Figure FC4.3: Flat and non-flat regions identified for different road structures using the k-means and EM algorithms. Flat and non-flat regions are highlighted in purple and yellow respectively.

the ideal frame format to employ for a frame-by-frame process of centerline detection. The column of the range image where the sensor, or the ego-vehicle, is positioned, is referred to as the centerline. A dense rasterized representation of the obscured view from the ego-vehicle position is referred to as a range image. As a result, it is created as a spherical projection of the points closest to the ego-vehicle, with the pixels colored according to the attribute of the closest point in the pixel. The angular resolution in the elevation and azimuthal angles determines the image resolution. For instance, the angular resolution for the Velodyne HDL-64E S2[1] that was used for SemanticKITTI data [2] acquisition has 64 angular subdivisions (*i.e.* $\approx 0.4^o$) in elevation angle spanning for 26.8^o, and similarly 0.08^o angular resolution for 360^o azimuthal angle, which gives a 64×4500 resolution of range images.

4.2.3 Edge Detection

We propose the use of a *scanline algorithm* on the range image in order to determine the edge points. We first scan the image of size $H \times W$ row-wise, where the key positions relative to the ego-vehicle, in the pixel space, are:

- at P_{cf}, *i.e.* $(0, \frac{W}{2})$, which indicates the centerline column in the front;
- at P_{cbL}, *i.e.* $(0,0)$, which indicates the centerline column in the back (rear), but on the left-side of the ego-vehicle; and
- at P_{cbR}, *i.e.* $(0,W)$, which indicates the centerline column in the back, but on the right-side.

Keeping in mind that the ego-left vehicle's and right sides are in relation to its front face. In order to scan the pixels in each row in the appropriate direction until a pixel carrying a non-flat region point is met, the pixels on the centerline columns are utilized

[1] This information is from the sensor specification sheet as published by the sensor manufacturer.

as the reference at each row. For front left and right pixels for non-flat region points, we traverse from P_{cf} towards P_{cbL} and P_{cbR}, respectively. Similarly, in the rear side, for left side, we traverse from P_{cbL} to P_{cf}, and for right side, from P_{cbR} to P_{cf}. After locating these pixels, their corresponding 3D LiDAR points are to be determined. We refer to these row-wise points as p_{fL}, p_{bL} on the left side, and p_{fR}, p_{bR} on the right side. These points are added to the side-specific sets, EP_L and EP_R, for the left and right sides, respectively in each frame.

In this step, the height differences pertaining to other surface variations on the road, *e.g.* potholes, are disregarded. We visualize the points in EP_L and EP_R and ensure that no additional surface artifacts are designated as road edge points. This is significant because artifact spots would have a negative impact on road surface extraction.

4.3 S_{2b} – Frame Classification

The surface extraction technique is influenced by the underlying road shape as expected. The road edge points are used in our suggested method [1] to create triangulated (surface) meshes. The edge locations along the perimeter of the road must be sampled sufficiently for precise edge extraction. The curvature of the road influences this sampling.

We first consider a broad classification of “straight” and “curved” roads as shown in Figure FC4.5. For three reasons, we currently limit our work to straight roads. To begin with, curved roads require more samples as edge points in order for the edges to be retrieved accurately, and the sample quantity is decided using geometric techniques. Second, from the perspective of the ego-vehicle, the wider road topology is not sufficiently recorded to extract the curved road edges piece-wise. The existing procedure is unable to capture the road topology for turnings and crossroads, which involves T- and X-intersections that must be recorded. Thirdly, more inner road points are required

Input : A sequence S of frame-wise point clouds $\{P(f_i) : 0 \leq i < n_{frames}\}$ with frame f_i at index i
Input : Trajectory information of the sequence $T(S)$
Output: 3D surface mesh of the extracted road R_m

$P_{edge} \leftarrow \{\}$ `// Set of edge points in` S
for *frame* f *in* S **do**
    `// Ground point detection`
    `// using Algorithm 1`
    $G_p(f) \leftarrow$ ground-point-detection($P(f)$)

    `// Frame classification using`
    `// top-view image as straight- or`
    `// curved-road`
    $TV_{\text{img}}(f) \leftarrow$ projection-xy-plane($G_p(f)$)
    $road\_type(f) \leftarrow$ classify-using-transfer-learning($TV_{\text{img}}(f)$)

    `// Straight-road edge detection`
    `// using Algorithm 3`
    if $road\_type(f)$ *is "straight-road"* **then**
        $E_p(f) \leftarrow$ straight-road-edge-detection($G_p(f)$)
        $P_{edge} \leftarrow P_{edge} \cup E_p(f)$ `// Merging all`
                                                `// edge points`
    end
end
$R_m \leftarrow$ generate-triangulated-mesh(P_{edge})
return R_m

Algorithm 2: The complete workflow of road surface extraction for road surface extraction from a sequence

in order to correctly extract curved road surfaces. These three issues call for a thorough investigation, which is outside the purview of our current work. Consequently, we demonstrate a proof-of-concept for our workflow for straight roads exclusively.

4.3.1 Transfer Learning Using ResNet-50 Architecture

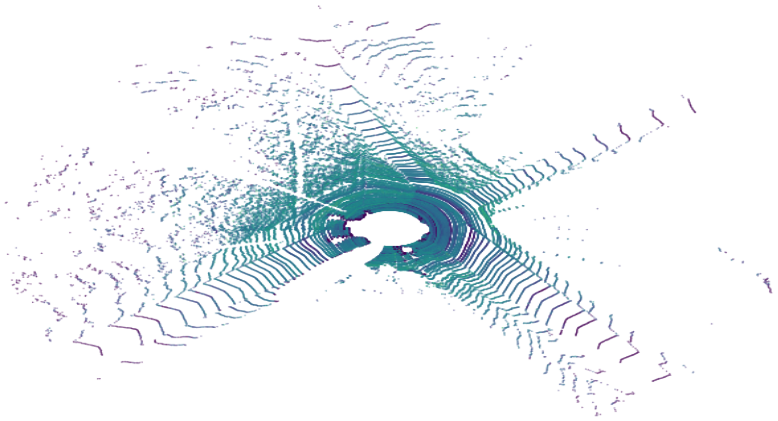
We see that each frame in the 3D LiDAR point cloud sequence clearly illustrates the road geometry from the top-view or a 2D projection of the points on the x-y plane. We suggest the use of transfer learning with ResNet-50 in order to take advantage of the perceptual variations across frames. Effective scene classification of perceptually distinguishable images has been achieved for other works using ResNet-50 [3]. We

generate the top view image of ground points for classification. Given that between 40 and 50 percent of the points in the point cloud are ground points, the point densities for ground points used for the top view projection image are not significantly different. Also, the point density is not a barrier for ResNet images of a given size at this time, however, managing the point density may help to improve accuracy in the future. Sample image for each road type is shown in Figure FC4.4.

The 2D top-view RGB image is rendered utilizing the attribute values of the points using the perceptually uniform sequential colormap, or Viridis colormap. The sequential colormap is then further discretized into, let's say, 5 bins. The color map is used to render the ground points found in $\mathbf{S_1}$ while taking into account their remission values. In these images, we apply transfer learning using the ResNet50 model (Figure FC4.5). Here, pre-trained weights for image classification of ImageNet are used, as per the de facto standard in transfer learning on images.

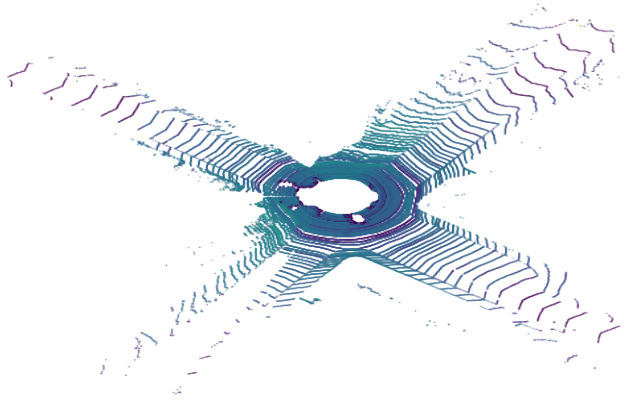
4.4 S_3 – Edge Point Set Smoothing

Now, the road edge points identified in $\mathbf{S_{2a}}$ are fitted to form edges. Due to the noise in the edges, these edges have to be smoothed. To prevent filtering out essential locations, the smoothing is carried out individually for the left and right sides of the road. Edge labeling is the process of identifying edges and removing false positives. Thus, $\mathbf{S_3}$ includes both smoothing and labeling. The global coordinate system, which contains the whole trajectory of the series, is used to implement the edge processing. The transformation of the coordinate system is therefore the initial substep.



(a) Turning road

(b) Straight road



(c) Cross road

Figure FC4.4: Sample top view images for straight road, turning road and crossroad generated with ‘viridis’ perceptually sequential colormap with remission values

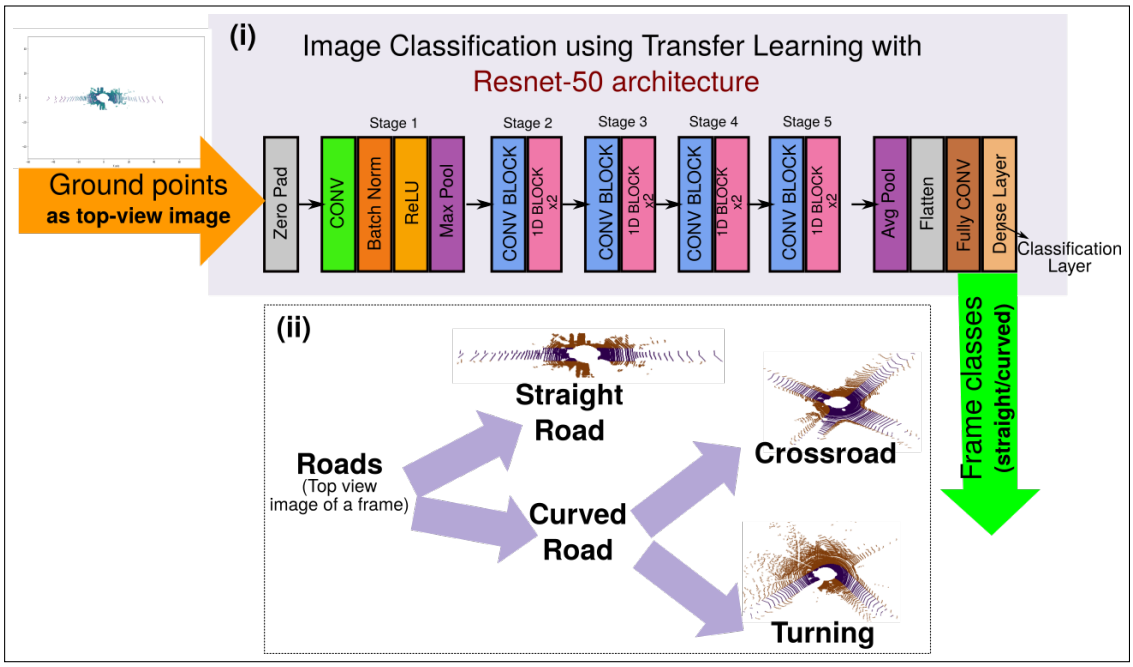


Figure FC4.5: Frame classification [1] implemented on top-view images of ground points in each frame, using (i) transfer learning using ResNet-50 architecture [3]. The possible class hierarchy for frames is given in (ii), of which we currently focus on the first level of straight and curved road classes.

4.4.1 Local to World Coordinate System Transformation

This transformation makes sure the smoothed edge is present in the 3D world space exactly as it is. The smoothing and transformation procedures are also *non-commutative* meaning that the sequence in which they are carried out must be rigorously preserved. Hence, we now add the trajectory information as an input to the workflow (Figure FC4.2. This input contains the position and poses of the ego-vehicle at each frame of the sequence. Transformation matrices are used to represent the shift in position and posture. Each frame's edge points are subjected to these matrices in order to translate them into the 3D world space.

4.4.2 Point Set Smoothing and Labeling

Using the converted coordinates, the straight road edges are smoothed. First, we identify the subsequences of frames that make up consecutive stretches of straight roads. Each side's implementation of this is done individually. The random sample consensus (RANSAC) line fitting model [74] is applied to each such subsequence. As a result, both sides of the road have disconnected smooth line segments resembling dashed lines.

4.5 S_4 – 3D Road Surface Extraction

Following smoothing, each continuous segment of the straight road's left and right edge points is used to create a triangulated mesh that represents the surface of the road. Here, a constrained Delaunay tetrahedralization [75] is implemented and is followed by the extraction of the outer/external surface of the tetrahedral mesh. When opposed to using projections of the 3D points for 2D Delaunay triangulation, this produces triangles of higher quality *i.e.* better shaped triangle closer to an equilateral triangle.

4.6 Experiments & Results

Sequences of LiDAR point clouds that follow the path of the vehicle must be included in the input dataset for road surface extraction, together with sufficient annotations to enable machine learning algorithms. SemanticKITTI [2] works well as our test data in this regard.

Input : Set of ground points $G_p(f)$ at a frame f with label "straight-road"
Input : Trajectory information of the sequence $T(S)$
Output: Set of road edge points $E_p(f)$

$NF_p(f) \leftarrow \{\}$ `// Set of non-flat region points`
for *point p in* $G_p(f)$ **do**
  $A_m \leftarrow$ compute-accumulation-map(p)
  $N_g \leftarrow$ find-local-neighborhood(p, *neighborhood_size*)
  $F_{\Delta z} \leftarrow$ compute-local-height-difference-features(G_p, N_g, A_m)
  $region\_type(p) \leftarrow$ classify-using-GMM($F_{\Delta z}$) `// Classify as flat or non-flat region`
  if *region_type*(p) *is "non-flat"* **then**
    $NF_p(f) \leftarrow NF_p(f) \cup \{p\}$
  end
end

`// Generate range image of size (H,W) using non-flat region points`
$R_{img}(f) \leftarrow$ range-image-generation($NF_p(f), G_p(f), H, W$)
$EP_L \leftarrow \{\}$
$EP_R \leftarrow \{\}$

`// Detect edge points from range image`
for $0 \leq row < H$ **do**
  $P_{cf} \leftarrow (row, \frac{W}{2})$ `// Determine centerline pixel in the front side using the column for sensor`
  $P_{cbL}, P_{cbR} \leftarrow (row, 0), (row, W)$ `// Determine centerline pixel in the back (rear) side using the columns for sensor`
  $p_{fL} \leftarrow$ point-in-pixel-furthest-from-centerline-in-pixel-interval($P_{cf}, \left[P_{cf}, P_{cbL}\right]$)
  $p_{fR} \leftarrow$ point-in-pixel-furthest-from-centerline-in-pixel-interval($P_{cf}, \left[P_{cf}, P_{cbR}\right]$)
  $p_{bL} \leftarrow$ point-in-pixel-furthest-from-centerline-in-pixel-interval($P_{CRL}, \left[P_{cbL}, P_{cf}\right]$)
  $p_{bR} \leftarrow$ point-in-pixel-furthest-from-centerline-in-pixel-interval($P_{CRR}, \left[P_{cbR}, P_{cf}\right]$)
  $EP_L \leftarrow EP_L \cup \{p_{fL}, p_{bL}\}$
  $EP_R \leftarrow EP_R \cup \{p_{fR}, p_{bR}\}$
  `// Correct the selected edge points in 3D world space`
  for *point p in* $\{p_{fL}, p_{fR}, p_{bL}, p_{bR}\}$ **do**
    $p \leftarrow$ transform-using-trajectory-information(p, $T(S)$)
  end
end

`// Postprocessing edges to remove outliers`
$EP_L \leftarrow$ smooth-edge-using-RANSAC-line-fitting(EP_L)
$EP_R \leftarrow$ smooth-edge-using-RANSAC-line-fitting(EP_R)
$E_p(f) \leftarrow EP_L \cup EP_R$
return $E_p(f)$

Algorithm 3: Straight road edge detection per frame, followed by collation of edge points from all frames (**S2a**, **S2b**, **S3**)

4.6.1 Implementation of Road Surface Extraction

The road surface extraction has been implemented on Intel core i7 CPU with 12 GB of RAM. We have used Open3D library APIs [76] for point cloud registration in $\mathbf{S_1}$. For neighborhood computation in $\mathbf{S_1}$ and $\mathbf{S_3}$, Open3D KDTree has been used. The scikit-learn library APIs [77] have been used for implementing the RFC and GMM models in $\mathbf{S_1}$ and $\mathbf{S_{2a}}$, respectively. Frame classification model in $\mathbf{S_{2b}}$ has been implemented using Keras APIs [78] and the model has been trained for five epochs. Edge point set smoothing in $\mathbf{S_3}$ has used the RANSAC model from the scikit-image library. PyVista library APIs [79] has been used for geometry computation in $\mathbf{S_4}$.

4.6.2 Dataset

We test our approach using the SemanticKITTI dataset [2] as described in Section 3.4.1. Since our work is different from the benchmark tasks, validation is not readily available for the dataset. Given its fit as input to road surface extraction, we use SemanticKITTI for our experiments and provide an appropriate qualitative and quantitative assessment.

The SemanticKITTI dataset comprises of the training sequence IDs, 00 to 10. We have used sequence 08 as the validation/test set, as prescribed by the data providers, thus training our model on the remaining training sequences for our classifier models, *i.e.* RFC model for ground point detection ($\mathbf{S_1}$), and transfer learning model for frame classification ($\mathbf{S_{2b}}$). We have only used every 10^{th} frame of training sequences of SemanticKITTI since frames are captured in 0.1s and our subsampling ensures significant variations in the vehicle environment are captured without incurring high computational costs. We have found that including more overlapping data resulted in increased computation without adding new information.

We group the classes to form the ground and non-ground classes as described in 3. The curbs of the road are labeled as sidewalk [50] and are important in our evaluation.

For the frame classification, we have manually annotated all frames in all training sequences, *i.e.* from 00 to 10, into "straight," "crossroad," and "turning".

4.6.3 Experimental Setup

For multi-scale feature extraction for ground point detection using RFC, hybrid criteria for neighborhood determination have been used for three different scales. We have commonly used the constraint of r of $1m$ for the spherical neighborhood in all the scales, and variable k values for the knn neighborhood, *i.e.* $k = 50, 100, 200$ neighbors. We have systematically experimented with several combinations of neighborhood criteria to arrive at this parameter setting as described in Chapter 3. Similarly, we have used similar hybrid criteria, *i.e.* $r = 1m$ and $k = 50$ for finding the local neighborhood of ground points for computing height-difference features to be used in the GMM for detecting flat and non-flat regions.

For road surface extraction, we employed the training dataset sequences 01, 05, 07, and 08. Additionally, we put our suggested approach [1] to the test using sequence 15 from the test dataset. Our results for all the sequences are given in Figure FC4.7. The performance of our edge point set smoothing in $\mathbf{S_3}$ in sequence 07 is demonstrated in Figure FC4.6.

The outlier removal using registration, geometric feature computation, and ground points classification takes approximately $\sim$11 seconds, $\sim$122 seconds, and $\sim$20 seconds respectively, per frame. The computation time for height feature extraction from ground points and edge extraction per frame takes approximately $\sim$34 seconds and $\sim$4 seconds respectively. The computation time for the final step of mesh generation for surface extraction for the entire trajectory is approximately $\sim$1 second and may vary depending

on the length of the trajectory.

4.6.4 Results

We perform both suitable quantitative evaluation and qualitative analysis using visualization for each step in our operation.

S_1: Ground Detection: The details of the sequences used in our models are given in Table TC4.2. The percentage of road points kept as ground points do not drop for any training sequences after the S_1 outlier removal. This demonstrates the effectiveness of our method [1] for removing outliers while keeping the "road" points. Table TC4.3 gives the findings of our ground detection utilizing RFC and other experiments we carried out by incorporating multi-scale characteristics and registrations. The results in Table TC4.3 coincides with the results in Table TC3.3 which are utilized for road surface extraction. The Table TC4.3 displays the mean IoU (mIoU) and average accuracy for the ground class for all frames of test sequence 08. Although GndNet [34] has reported a mIoU of 83.6 percent, this number is not comparable because their mIoU was determined by combining the ground and non-ground classes. Similar to this, the ground segmentation in [22] reported a mIoU of 78.46 percent for the "ground" class, but this is not comparable because their "ground" class also includes "vegetation". These mIoU scores suggest that, despite the fact that we cannot directly compare, our method for classifying ground points exhibits a really high level of accuracy.

S_{2a}: Frame Classification: As a test, we have developed two distinct frame classification models that are matched to the various levels of the road geometry class hierarchy (Figure FC4.5). The details of the sequences and number of frames for different road geometry class used for training and validation/testing is shown in Table TC4.1 The first model is used to categorize roads as straight or curved, and the second is used to categorize roads as straight, crossroads, and turnings. We also experimented with the grayscale

Table TC4.1: Details of the SemanticKITTI [2] sequences used for training and validation/testing in frame classification [1]

Seq. ID	**# Frames**				
Training	**Total**	**Used**	**Straight**	**Crossroad**	**Turning**
0	4,541	455	238	205	12
1	1,101	111	69	23	19
2	4,661	467	195	134	138
3	801	81	17	48	16
4	271	28	25	3	0
5	2,761	277	172	102	3
6	1,101	111	54	57	0
7	1,101	111	54	57	0
9	1,591	160	42	39	79
10	1,201	121	64	32	25
All	**19130**	**1922**	**930**	**700**	**292**
Testing					
8	4071	408	261	124	23

images using binary colormap for comparison with RGB images using Viridis colormap along with different bins to form sequential + quantitative colormap. We also verified the results across two models *i.e.* ResNet-50 as described in Section 4.3.1 and similarly used transfer learning with the same ImageNet weights with the MobileNet architecture. We perform this experiment only on 1st hierarchy *i.e.* "straight" and "curved" roads. The results of this experiment are shown in Table TC4.4. The results show that ResNet-50 performs better than MobileNet.

Table TC4.5 displays the validation accuracy for both frame classification models on sequence 08 for each model. We utilized the model for "straight" and "curved" roads because it had a greater level of categorization accuracy in this instance. Future research can further enhance these accuracy scores by addressing the class imbalance at both hierarchical levels (Table TC4.2).

S_3: Edge Point Set Smoothing: Following **S_3**,Table TC4.6 displays the class distribution of our road edge points. We note that the majority of the edge points primarily belong to the "road" class, followed by the "sidewalk" class. This demonstrates that, as expected, our method [1] finds edge points marked as "road" or "sidewalk". This

Table TC4.2: Specifications of the SemanticKITTI [2] sequences used for training and validation/testing in road surface extraction [1]

Seq. ID	**Ground truth (GT)**			**Filtered after outlier removal in S1**	
Training	**#Points**	**#"Ground" points**	**"Road" * (%)**	**# "Ground" points**	**"Road" * (%)**
0	55,300,603	21,242,723	45.2	20,928,740	45.2
1	11,737,924	6,684,753	**71.8**	6,425,398	**72**
2	58,678,800	26,955,344	42.8	26,568,086	42.8
3	10,038,550	4,563,802	**48.8**	4,485,650	**49**
4	3,518,075	1,816,228	**65.9**	1,779,528	**66.4**
5	34,624,816	14,025,815	40.5	13,802,511	40.5
6	13,567,503	8,417,991	34.1	8,223,230	34.1
7	13,466,390	5,301,837	**48.1**	5,233,937	**48.2**
9	19,894,193	9,313,682	**45**	9,159,419	**45.1**
10	15,366,254	5,608,339	**43.7**	5,487,403	**43.9**
All	**236,193,108**	**103,930,514**	**45.3**	**102,093,902**	**45.3**
Testing				Filtered and Classified in S 1	
8	50,006,369	21,943,921	**40.3**		**41.1**

* Our annotation of "ground" combines five classes, namely, "road," "parking," "sidewalk," "terrain," and "other ground," as given in SemanticKITTI dataset [2]. Percentage values in columns 9 and 11 give the fraction of ground points in columns 8 and 10, respectively, that are annotated as "Road" in SK. Boldface indicates improvement in retaining road points, thus demonstrating the efficiency of processes in $\mathbf{S_1}$.

Table TC4.3: Ground detection using random forest classifier (RFC) [1]

Set of points to be classified	# Scales for local features	Classification accuracy (%-age)	mIoU (%-age)
All points	1 (single scale)	96.37	89.38
	3 (multi-scale)	96.63	90.63
Filtered* points	1 (single scale)	96.58	89.47
	3 (multi-scale)	**96.91**	**90.79**

* Filtered points are those that were retained after outlier removal in $\mathbf{S_1}$.

demonstrates the effectiveness of $\mathbf{S_{2a}}$, $\mathbf{S_{2b}}$, and $\mathbf{S_3}$. The results of the edge point set smoothing for sequence 07 are shown in Figure FC4.6, which demonstrates a significant reduction in edge point noise and smooth road edges on the left and right of the trajectory.

The root means square error between the edge point sets computed using "road" (GT) and "ground" (detected in $\mathbf{S_1}$) points are calculated after performing these three steps on the "road" points in the ground truth (GT). We perform this analysis due to the lack of ground truth of edge points and retrieved surface. In Figure FC4.7 the RMSE values for Sequences 01, 05, 07, and 08 are provided. We notice that the RMSE errors

Table TC4.4: Frame classification results using 'Viridis' colormap for RGB image and 'binary' colormap for gray scale image with different number of bins for sequential+quantitative colormap and two models on SemanticKITTI dataset [2].

Image type	**No. of bins (Binning values to get sequential + quantitative colormap)**	**Resnet50**	**Mobilenet**
RGB (with local maxima & minima) - Using Viridis colormap	5	**88.50%**	74.50%
	10	85.37%	71.13%
	20	84.13%	70.13%
	No binning	85.25%	70.88%
Grayscale (with local maxima & minima) - Using binary colormap	5	79.37%	79.37%
	10	76.13%	73.37%
	20	73.25%	60.00%
	No binning	79.50%	78.37%

Table TC4.5: Results of frame classification [1] using transfer learning on SemanticKITTI dataset [2]

#Class hierarchy* **levels**	**Class outcomes**	**Classification accuracy** (%age)
1	Straight road, Curved road	82.35
2	Straight road, Crossroad, Turning	78.51

* Frame class hierarchy is as shown in Figure FC4.2).

are rather small, given that each frame has an extent of 51.2 meters in front of the car and 25.6 meters on either side [2].

S_4: 3D Road Surface Extraction: Figure FC4.7 shows the results of the 3D extracted surface for trajectories of various sequences. Our qualitative findings demonstrate that road surface extraction is effective on closed trajectories, complex trajectories, and straight roads with lots of contiguous straight road segments. For the purpose of illustrating their similarities, our results of surfaces produced using detected ground points and "road" (as per ground truth (GT)) points are overlaid. It is clear that the triangulated meshes have adequately covered brief turning segments and connections between straight road segments.

On test sequence 15, road surface extraction has also been fully accomplished. As the GT for the test sequences is not available, we use surface and point rendering to qualitatively compare our recovered road surface with the reference trajectory (Fig-

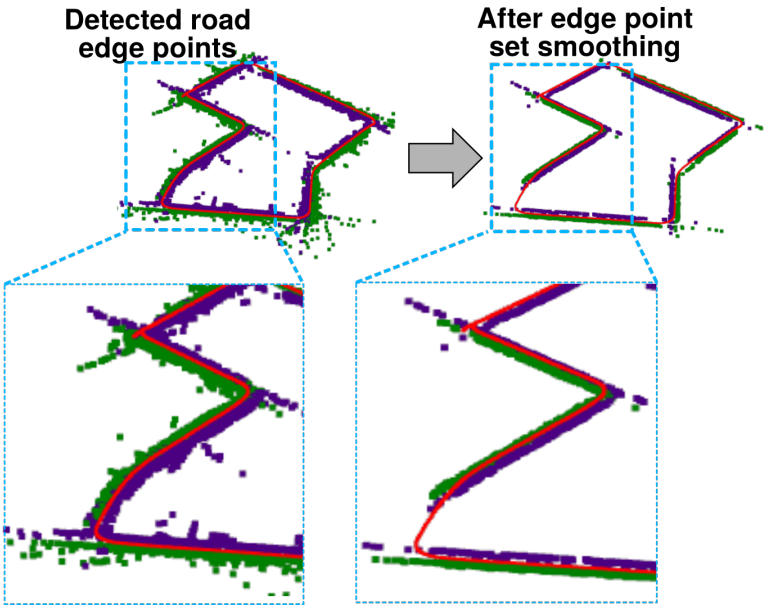


Figure FC4.6: Edge point set smoothing [1] results on sequence 07 data of SemanticKITTI dataset [2] trajectory (Top row) with zoomed-in region inset (Bottom row). Red, purple, and green points show the trajectory, left and right edge points, respectively.

Table TC4.6: Class distribution of road edge points in extracted surface

GT Class ↓	# Points (**% age**)			
Seq. ID →	01	05	07	08
Road	12,437 (**94.0**)	10,093 (**63.4**)	3,864 (**76.5**)	19,856 (**84.3**)
Parking	0	408 (**2.6**)	448 (**8.9**)	710 (**3.0**)
Sidewalk	6 (**0.0**)	5,243 (**32.9**)	695 (**13.8**)	2,599 (**11.0**)
Terrain	519 (**3.9**)	161 (**1.0**)	42 (**0.8**)	393 (**1.7**)
Other-ground	276 (**2.1**)	18 (**0.1**)	0	0
Non-ground	0	0	0	7 (**0.0**)

* Underlined %-age values show that road edge points in the extracted surface belong to "road" and "sidewalk" classes, predominantly, and as desired.

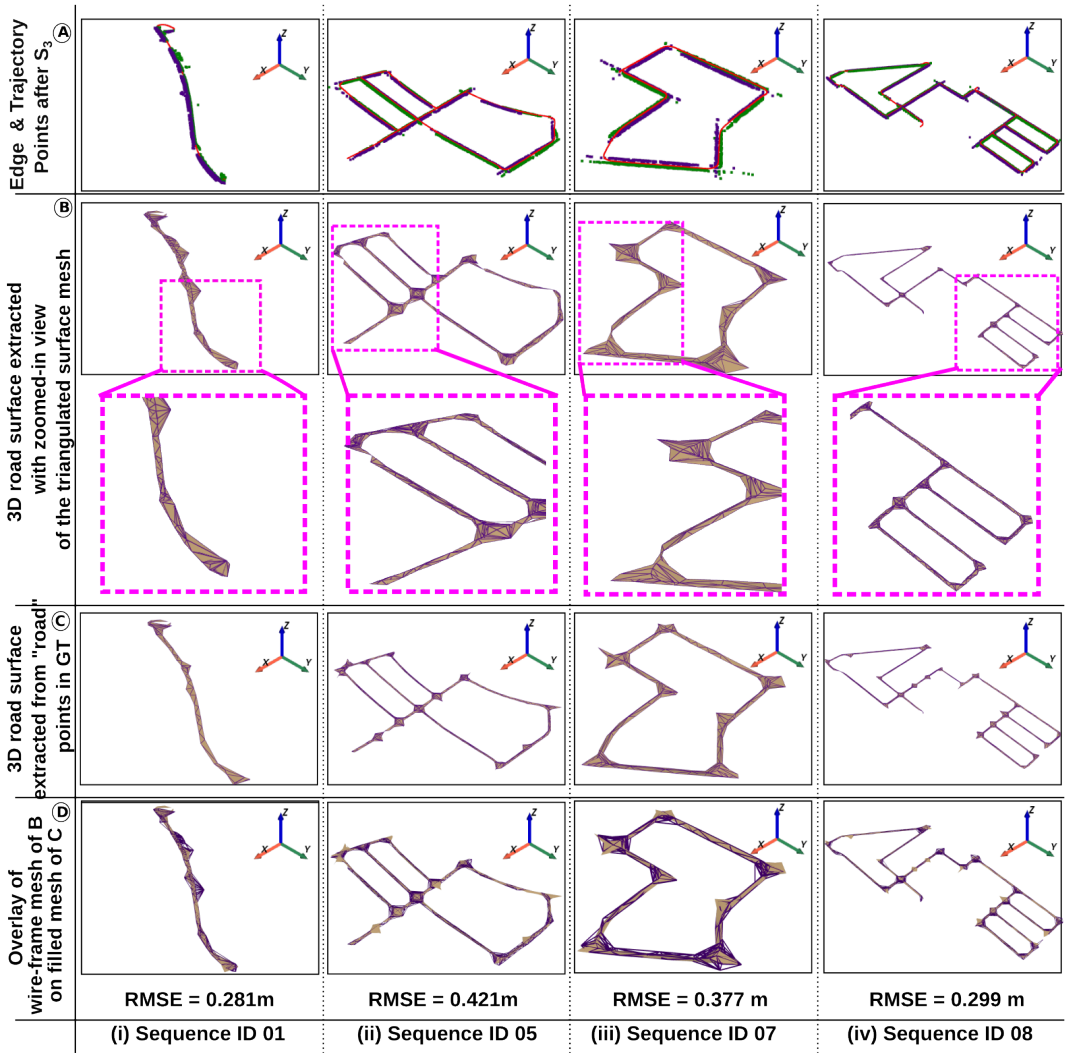


Figure FC4.7: Results of implementing road surface extraction [1] on training sequences of SemanticKITTI [2], where 01, 05, and 07 have been used for training our learning models, and 08 have been used for validation/testing. Row A follows the color scheme mentioned in Figure FC4.6. For rows B, C, and D, wireframe meshes are shown in indigo, and filled meshes are shown in tan color.

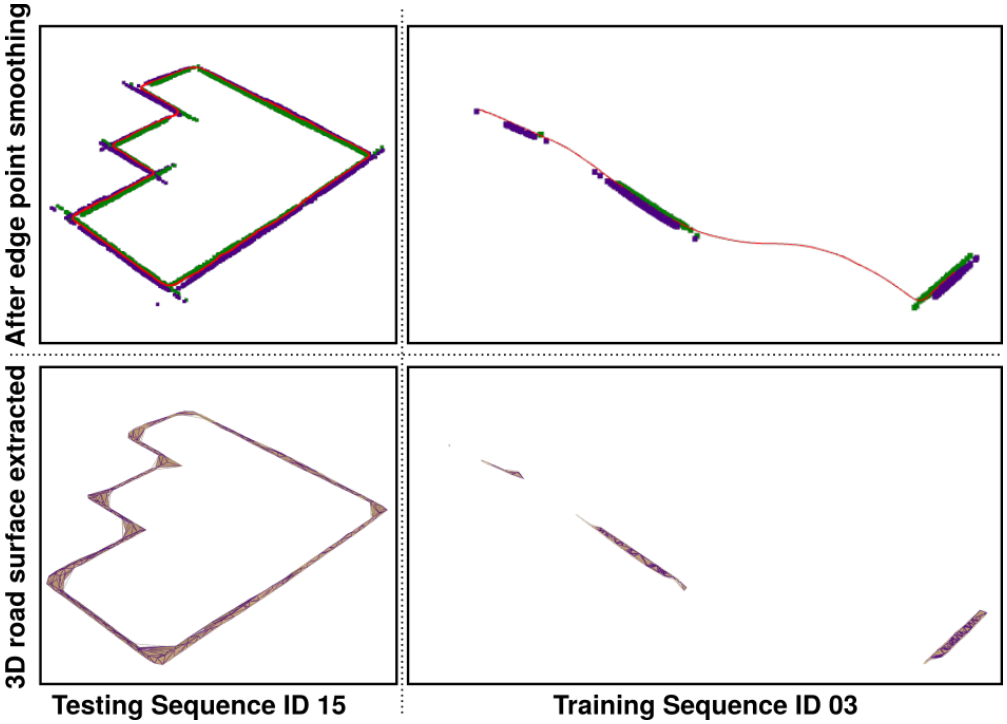


Figure FC4.8: Results of road surface extraction [1] on (Left) a sample sequence from the test set of SemanticKITTI [2], for which annotations for semantic segmentation have not been published; and (Right) a sample sequence where the surface is only partially extracted.

ure FC4.8 (Left)). We see that the road surface mesh maintains the trajectory as its *medial skeleton* [80] in all of the sequences, including 15.

Road surface extraction, however, is unable to extract the road surface over the whole trajectory in cases where (a) straight road segments are substantially fragmented and (b) subsequences contain portions of curved and straight roads that are comparable in length. Road surface extraction only extracts a portion of the road surface when edge points are not detected for significant portions of the road, as is the case in training sequence 03 (Figure FC4.8 (Right)). For such cases, surface extraction calls for a thorough investigation of curving roads, which is outside the scope of our current work.

4.7 Summary

For a series of 3D automobile LiDAR point clouds, we have suggested and implemented road surface extraction, a unique approach for automating 3D road surface extraction. A five-step workflow is used. First, ground point detection is accomplished via outlier elimination, multiscale feature extraction, and supervised learning with RFC. Second, the EM algorithm detects road edge points based on height variations in ground points. In addition, our frame classification uses transfer learning using ResNet-50 on top-view images to determine the road geometry. The edge point set in the sequence is smoothed in the fourth step. The 3D road surface is then retrieved as a triangulated mesh using 3D Delaunay tetrahedralization as the final step. Our studies on four SemanticKITTI sequences with varied degrees of geometry complexity produced promising results that have been both qualitatively and quantitatively confirmed. Therefore, road surface extraction primarily works on trajectories with contiguous straight roads.

Our method fails to extract road surfaces for the full trajectory when the segments do not have contiguous straight road geometry, despite the fact that road surfaces across different trajectories are extracted with a high level of visual similarity using the suggested methodology. Thus, extending our algorithm to curved roads is in the scope of future work. Other unresolved issues for the road surface extraction application include a more reliable measure and ground truth for validation.

CHAPTER 5

BOOSTING 3D OBJECT DETECTION

Recognizing and locating items in a 3D environment is a crucial first step toward scene understanding which can be used for augmented reality and autonomous navigation applications. The next important perception task for autonomous driving assistance after ground and non-classification and road surface extraction is 3D object detection. Chapter 3 and Chapter 4 proposes the spatio-temporal analysis and approach for ground detection and road surface extraction. This chapter presents the existing approaches and our proposed approach for boosting 3D object detection. 3D object detection is a task of localization and recognition of objects present around the self-driving vehicle in a 3D scene. Monocular and stereo 3D object identification techniques frequently use camera-produced images [81]. Since images have dense pixels and a clear appearance and texture, image-based detectors can easily identify the object. Object detection has always been a difficult challenge in the field of computer vision using images because of the variable look, shape, and attitude of numerous objects as well as the interference of lighting, shielding, and other elements during imaging. This prompted many researchers to design an effective framework to integrate features from different perspectives to achieve accurate 3D object detection. Early multi-sensor feature fusion methods take RGB image, front view, and bird's eye view (BEV) as input and then directly combine and merge the features by cropping and resizing to generate 3D candidate boxes, such as MVF [82] and AVOD [83], but they ignore the different perspectives

of image and BEV.

LiDAR point cloud data do not have sufficient sampling to detect or classify small-scale vulnerable road users such as pedestrians, cyclists, and motorcyclists, since they are represented by fewer points compared to other objects like cars, trucks, and trailers. Sometimes it is observed that an object is misclassified or misrepresented. However, many state-of-the-art 3D object detectors, such as those in [84], generally utilize point clouds produced by LiDAR as the primary input modality because they provide exact depth information, and are resistant to changing weather conditions, and illumination. However, the sparsity of point clouds grows with distance because of laser-ray divergence. The far-off objects have fewer points, making them highly sparse. Their object boundaries and semantic classes are difficult to predict. The LiDAR-based 3D object detectors are still accurate and robust as compared to camera images [85].

A few of the state-of-the-art approaches fuse the LiDAR and camera sensor modalities to utilize multi-modality input for boosting the 3D object detection [85]. One such way of fusion is depth completion which refers to completing relatively dense depth images that contain missing values, such as those utilizing exemplar-based depth inpainting, low-rank matrix completion, object-aware interpolation, tensor voting, Fourier-based depth filling, background surface extrapolation, learning-based approaches using deep networks, and alike. The depth maps are produced by the framework [86] by extracting both global and local information based on the confidence maps in a late fusion approach. In another approach, the camera and LiDAR features are fused using the 3D ROI-based gated fusion network where a region of interest (ROI), provides samples within a data set identified for the fusion. [59].

Individual detection model approaches may work well in a given environment, but when that environment changes over time or in specific scenarios such as obstructions, objects overlap, occlusion, unexpected scenarios, etc. the model may not provide the

desired results due to differences in the underlying changes in data captured in different environments. We propose a novel adaptive selection algorithm to improve 3D object detection by choosing detection from various existing methods. Our method employs an adaptive model selection mechanism to provide more accurate detection by selecting one of several object detections from various models in various environments.

5.1 Base Model for 3D Object Detection

Three different point cloud representations are widely used as input to the 3D detector. (1) Since the original point cloud may be processed directly using PointNet++ [87] based on the point representation, this method preserves as much of the original geometric data as possible without the need for transformation. The application of PointNet++ to 3D detection based on the cropped point cloud of a 2D image bounding box is proposed by F-PointNet [57]. The first point-based 3D object detection technique that exclusively uses point clouds as network input is proposed in point-RCNN [56]. A point-based, single-stage 3D object identification framework with an excellent balance of accuracy and speed was proposed by 3DSSD [88]. (2) This approach transforms point clouds into regular grids based on the voxel representation so that 3D CNN can use this representation directly. The point cloud is divided into voxel representations by SECOND [89], which then creates 3D bounding boxes by learning voxel characteristics using sparse convolution. By converting point clouds into pseudo images, PointPillars [90] replaces laborious 3D convolution processes with 2D convolutions. The attention mechanism is introduced by Fast-PointRCNN [91] to improve the replacement capabilities of the network. The candidate box is honed and the 3D detection precision is increased by the ROI-aware pooling suggested in voxel representation methods. Although the voxel-based technique has excellent perceptual abilities, the voxelization procedure will result in information loss. (3) Point-voxel joint represen-

tation method takes points and voxels as inputs and fuses the features of points and voxels at different stages of the network for 3D object detection, such as Part-A_2 [92] and PV-RCNN++ [93].

When direct 3D point clouds are used for object detection, approaches based on different point cloud representations may unnecessarily increase the computational burden and introduce a large number of potential false-positive detections. The CenterPoint method represents objects as points, greatly simplifying 3D recognition [4]. This method demonstrates that a simple switch from box to center-based representation yields a 3-4 mAP increase in 3D detection under different backbones [89,90,94,95]. As a result, using the center-based approach as the base model for our adaptive prediction-selection model seems more appropriate.

5.1.1 Center-based 3D Object Detection and Tracking

In this study, a two-stage 3D detector called CenterPoint [4] uses a keypoint detector to discover the centers of objects and their attributes, and a second step refines these predictions. In particular, CenterPoint creates a representation of the input point cloud using a typical LiDAR-based backbone network, such as VoxelNet [94] or PointPillars [90]. It then flattens this representation into an overhead map-view and locates item centers using a common image-based keypoint detector. It descends from a point feature at the central location to all other object properties for each detected center, such as 3D size, orientation, and velocity. Furthermore, to fine-tune the item placements, a lightweight second stage is employed. We refer to this representation as a *center-based* one.

The second stage extracts point features from the 3D bounding box of the estimated object at the 3D centers of each face. It gives an additional 2 mAP boost at a relatively low cost by recovering the lost local geometric information as a result of striding and a

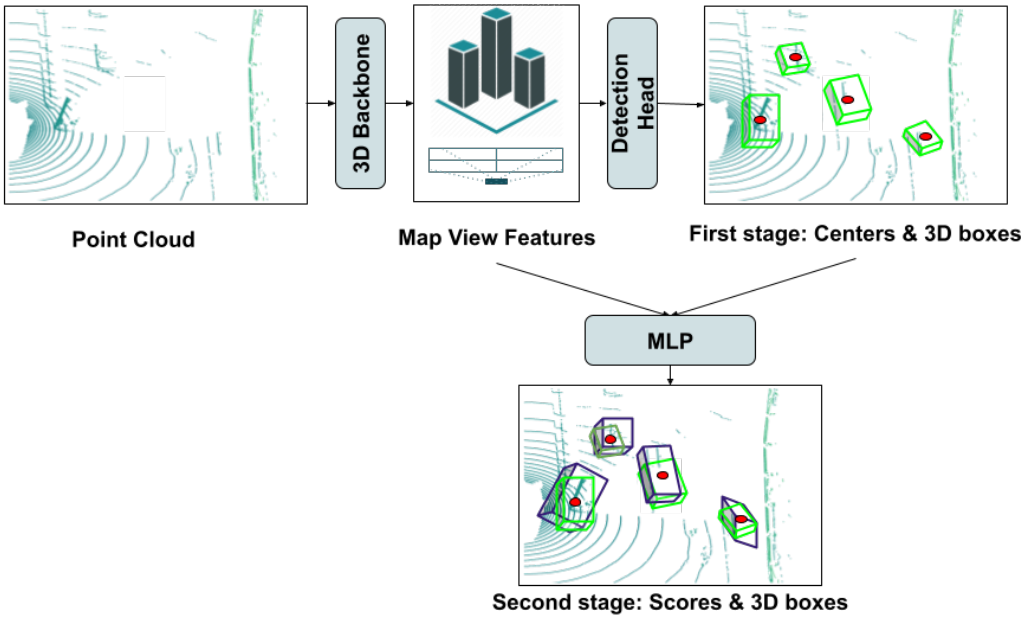


Figure FC5.1: Overview of CenterPoint [4] framework. It relies on a standard 3D backbone to extract the map-view feature and a 2D CNN architecture detection head to get a full 3D bounding boxes. Then, box prediction is used to extract point features at the 3D centers, which are used to predict confidence score and box regression refinement. The image is created and inspired from Figure 2 in CenterPoint approach [4] using point cloud from a frame of nuScenes dataset [5].

constrained receptive field.

The center-based representation has the following major benefits: First, points lack intrinsic orientation, in contrast to bounding boxes. In addition to enabling the backbone to learn about the rotational invariance of objects and the rotational equivariance of their relative rotation, this significantly shrinks the search space for the object detector. Second, a center-based representation makes subsequent activities, like tracking, simpler. Tracklets are pathways in space and time if objects are points. The relative offset of objects between two frames is predicted by CenterPoint and then greedily linked. Thirdly, point-based feature extraction makes it possible to create an efficient, two-stage refinement module considerably more quickly than with earlier methods.

The CenterPoint network architecture utilizes two common backbone architectures VoxelNet [94] and PointPillars [90]. VoxelNet architecture separates a point cloud into

evenly spaced 3D voxels, and through the voxel feature encoding (VFE) layer, converts a set of points within each voxel into a single feature representation and transforms points within each voxel to a vector representation characterizing the shape information. The space is represented as a sparse 4D tensor. Thus, a descriptive volumetric representation of the point cloud is created, which is subsequently coupled to an RPN to provide detections. This efficient algorithm benefits both from the sparse point structure and efficient parallel processing on the voxel grid.

The PointPillars approach for 3D object detection [90] allows end-to-end learning using only 2D convolutional layers. In order to forecast 3D oriented boxes for objects, PointPillars use a new encoder that learns features on the pillars (vertical columns) of the point cloud. This strategy has a number of benefits. First, PointPillars can take advantage of the whole information provided by the point cloud by learning features rather than depending on fixed encoders. Additionally, since pillars are used instead of voxels, manual vertical direction binning tuning is not necessary. Last but not least, pillars are quick due to the fact that all critical operations can be expressed as 2D convolutions, which are very effective to calculate on a GPU.

On nuScenes dataset [5] CenterPoint model outperforms the challenge winner CBGS with multi-scale inputs and multi-model ensemble [95]. This gives us a good justification for employing this model as the foundation for boosting 3D object detection.

5.2 Boosting Models

Boosting is an ensemble method for improving the model predictions of any given learning algorithm which generally involves training multiple models in succession so that each one tends to fix the mistakes made by the one before it. We experiment with two different approaches for boosting the 3D object detection by utilizing detections from existing deep learning models. Model detections are the bounding boxes of ob-

jects predicted by the model. We use the CenterPoint model with VoxelNet and PointPillars backbone as base models for both approaches. The first approach is of merging the detections by taking the weighted parameters of bounding boxes based on the confidence of the prediction. Another approach we propose is of adaptive selection model which adeptly chooses one from different detections based on the shape of the detected objects.

5.2.1 Weighted Boxes Fusion (WBF)

A new technique for merging the predictions of object detection algorithms is called weighted boxes fusion [6]. In order to achieve high accuracy, ensemble models are developed (often during competitions), where all of the predictions from many models are combined to provide the final forecasts. For the purpose of combining all of the predictions from several models, weighted boxes fusion is utilized. Non-maximum suppression (NMS) a crucial component of the process for object detection takes the detection box with the highest score and all additional detection boxes that have a considerable overlap (using a predefined threshold) with it are suppressed whereas Soft-NMS, continuously decays the detection scores of all other objects as they overlap. In contrast to NMS (Non-Maximum Suppression) or Soft-NMS approaches, which merely eliminate a portion of the predictions, the WBF method constructs the weighted average of bounding box coordinates and properties using the confidence scores of all proposed bounding boxes. The comparison between NMS/soft-NMS and WBF results for a group of incorrect predictions is shown schematically in Figure FC5.2. The quality of the combined predictions improves as a result of this. We evaluate the performance of the weighted boxes fusion across the CenterPoint-VoxelNet and CenterPoint-PointPillars models on the filtered bounding box based on the match across the models.

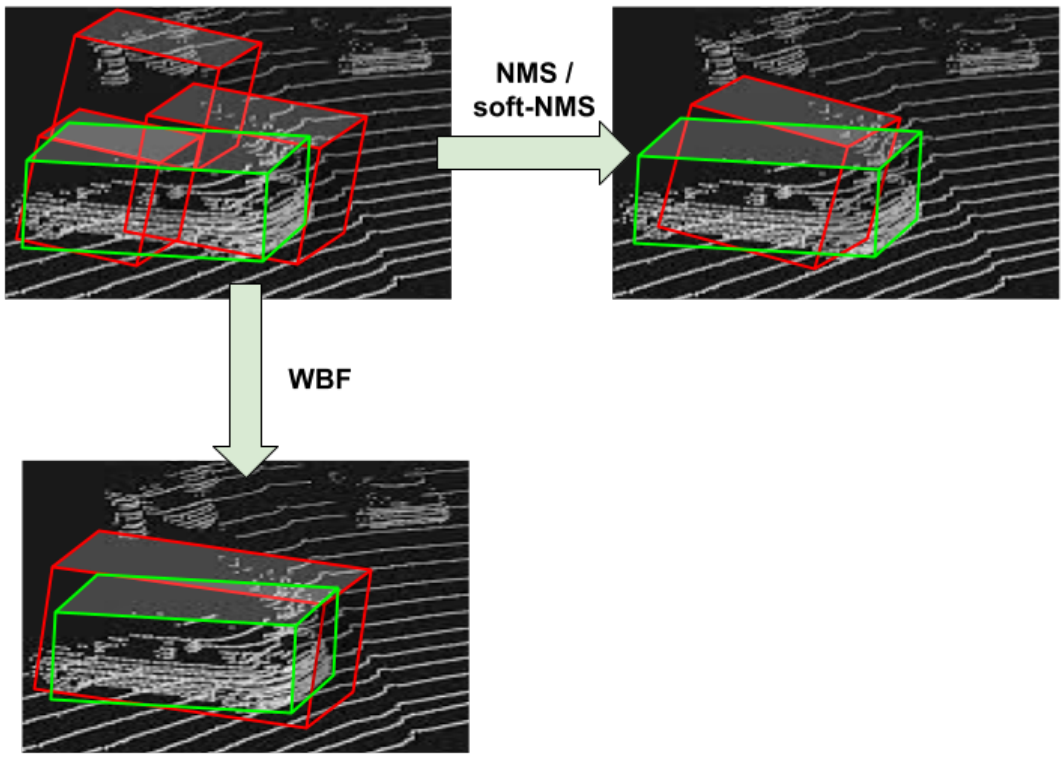


Figure FC5.2: The schematic illustration of NMS/soft-NMS vs. WBF outcomes for an ensemble of inaccurate predictions. The red color box represents different models' predictions and the Green color box represents ground truth. The image is created based on the Figure 2 in Weighted Boxes Fusion approach [6] using a partial point cloud from a frame of nuScenes dataset [5].

5.2.2 Adaptive Selection Model

Extensive research work has been done for building various deep learning and supervised models for 3D object detection tasks. An individual model may work well under a certain operating condition or degradation pattern, but when the degradation status of the model changes over time or in certain scenarios, it may not provide desired results due to the difference in the underlying changes in data captured in a different environment. An adaptive model selection mechanism is necessary to be able to provide more accurate detection information by capturing the characteristics of various detected objects. In some situations, an adaptive model selection mechanism can be achieved by changing the parameters of an individual model, but it may be very difficult to achieve

satisfactory accuracy due to the limitations of the model itself. Multiple models are more desirable and promising for different detections. The adaptive selection model adeptly chooses one from different detections based on the shape of the detected objects. We use a shape descriptor histogram to get the characteristics of the detected object and predict the closest histogram distance to ground truth and choose the more accurate detection.

5.3 Our Proposed Adaptive Model Selection Method

By selecting detection from various existing methods, we propose a novel adaptive selection algorithm to improve 3D object detection. To provide more accurate detection, our method employs an adaptive model selection mechanism that selects one of several detections from various models in various environments. The objective of the adaptive learning-based prediction model selection is to obtain a mapping from each state/detection pair, which provides a "look-up table" for corresponding detections from each model accessible for selection. The state/detection pair indicates the preference of selecting an individual detection under a certain state or from a certain model. After training the simple supervised machine learning framework with the detected object characteristics defined by the shape descriptors, the "look-up table" along with the closest distance of shape descriptor to ground truth descriptor characteristics will be capable of giving recommendations of the most appropriate detection from an ensemble of 3D detection models.

When the car is in an intersection, one similar method is used for object tracking tasks in [61] to enhance prediction and analysis outcomes by employing several prediction models. The findings demonstrate that, compared to employing a single, constant velocity model, the employment of several models has lowered the analysis RMSE in tracking and provided a better assessment of the analysis uncertainty.

We propose a novel framework for adaptive prediction-selection based on the shape descriptor using a random forest regression algorithm. 3D object detection models with different architectures can be adopted as the prediction models in this framework. For the time being, our approach only involves two models at a time, as we propose two-way matching on detections across two models to map the corresponding objects. We use the CenterPoint with PointPillars as $Model_1$ and CenterPoint with VoxelNet as $Model_2$ as base prediction models in this framework for our spatial analysis to boost the 3D object detection. As described in Section 5.1.1 the CenterPoint network provides good justification for being used as base model and the architecture utilizes two common backbone architectures, VoxelNet [94] and PointPillars [90]. To show the effectiveness of our approach for adaptive selection among the output of detection models, we use the same center-based CenterPoint approach but with two different backbone architectures, VoxelNet [94] and PointPillars [90], as base detection models.

Our approach functions as a model stacked on the ensemble of detection models to learn and select an accurate prediction. This is different from using the stacking ensemble approach, the idea of which is to learn from several different weak learners and combine them by training a meta-model to output predictions based on the multiple predictions returned by these weak models. Here, instead of combining predictions from an ensemble of models, we select one of the predictions from the ensemble of models as the final output. The proposed novel adaptive prediction model selection scheme can autonomously decide on appropriate detection to select according to the distance of its shape descriptors from the ground truth. Our proposed framework is shown in Figure FC5.3.

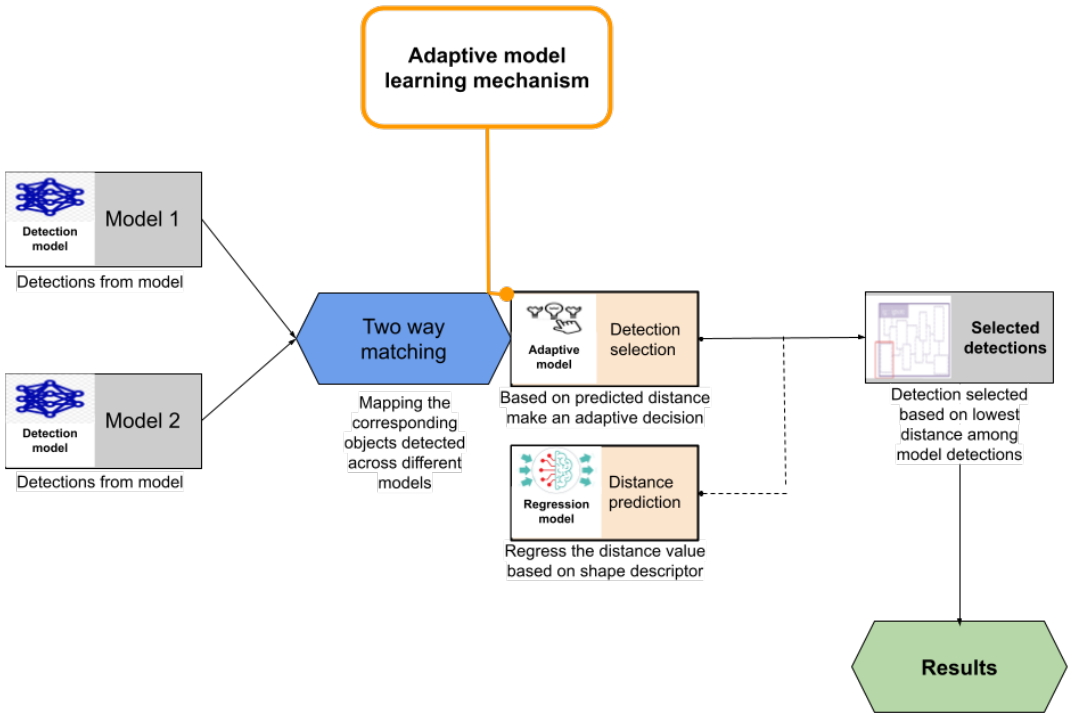


Figure FC5.3: Our proposed workflow for adaptive model strategy. We perform two-way matching for mapping corresponding objects across predictions from 3D object detection models. For each detection box match, the detection corresponding to the closest(lowest) distance is selected for a final set of detections based on the predicted distance of the shape descriptor histogram.

5.3.1 Mapping the Corresponding Objects Detected across Different Models

Obtaining a mapping of detections from several models is the first stage in an adaptive learning based prediction-selection model. We consider only two models $Model_1$ and $Model_1$, as stated in Section 5.3 at a time for mapping. Detections from the model represent the bounding boxes of objects $\mathcal{B}$ predicted by the model. This mapping creates a "look-up table" for corresponding detections from each model that can be chosen. The decision to choose a single detection under a particular state or from a particular model is influenced by the detection pair. There is a strong likelihood that two bounding boxes belong to the same object if their spatial relationship indicates their closest match in the spatial domain. Thus, the closest detected bounding boxes are taken into

consideration for our current framework, which uses the bird-eye-view center distance (xy-only) between two detected bounding boxes.

We first register the equivalent match among the detections of $Model_2$ for each of the detections of $Model_1$, *i.e.* mapping each bounding box from $\mathcal{B}_1$ to $\mathcal{B}_2$ which is a one-way match. Similar to this, we register the match for each of $Model_2$ detections across $Model_1$ detections, *i.e.* mapping each bounding box from $\mathcal{B}_2$ to $\mathcal{B}_1$ which is again a one-way match but the order of finding an equivalent match is reversed. The bounding boxes mapped in each one-way match *i.e.* ($\mathcal{B}_1$,$\mathcal{B}_2$) and ($\mathcal{B}_2$,$\mathcal{B}_1$) are stored in a look-up table. Two-way matching is about registering ($\mathcal{B}_1$,$\mathcal{B}_2$) from $Model_1$ to $Model_2$ as R_1, and registering ($\mathcal{B}_2$,$\mathcal{B}_1$) from $Model_2$ to $Model_1$ as R_2, and only if $R_1(\mathcal{B}_1,\mathcal{B}_2)$ == $R_2(\mathcal{B}_2,\mathcal{B}_1)$, *i.e.* both the registered pairs have same bounding boxes, the matching is retained. The identical matches from the two-way match strategy are preserved and the remaining matches are thrown away. Because the number of detections from different models varies, two-way matching is preferable to one-way matching. As a result, there is always a probability that a false detection from one model will not be present in the detections of another model, and thus such false detections can be removed using two-way matching. We use two-way matching in the spatial domain between two detection models to determine the match across detections for each frame.

We only use one-way matching in the spatial domain with the same bird-eye-view center distance (xy-only) to be used for mapping the detected bounding boxes of the prediction model and bounding boxes of objects in the ground truth while training the adaptive learning selection model. There is no need for two-way matching to find a match among the ground truth bounding boxes because the ground truth bounding boxes do not require finding a match among detections. We employ two-way matching across detection models to validate our proposed model.

Input : List $\mathcal{B}_1$ and $\mathcal{B}_2$ of detections from 3D object detection model $Model_1$ and $Model_2$
Output: Selected 3D object detections by adaptive model $\mathcal{B}_f$

$M_1 \leftarrow \{\}$ // Set of matched bounding boxes for detection in $\mathcal{B}_1$ to detection in $\mathcal{B}_2$
$M_2 \leftarrow \{\}$ // Set of matched bounding boxes for detection in $\mathcal{B}_2$ to detection in $\mathcal{B}_1$
$M \leftarrow \{\}$ // Set of matched bounding boxes after two-way matching
// match bounding boxes
Function matchDetections(B_1,B_2)**:**
  for *bounding boxes* b_1 *in* B_1 **do**
    for *bounding boxes* b_2 *in* B_2 **do**
      if $center(b_1)$ *is closest to* $center(b_2)$ ***and***
      Euclidean distance between $center(b_1)$ *&* $center(b_2) <$ *distance threshold* **then**
        $M_b \leftarrow$ matched-detection (b_1, b_2)
      end
    end
  end
  return M_b
End Function
// match bounding boxes of $\mathcal{B}_1$ to bounding boxes in $\mathcal{B}_2$
$M_1 \leftarrow$ matchDetections$(\mathcal{B}_1, \mathcal{B}_2)$
// match bounding boxes of $\mathcal{B}_2$ to bounding boxes in $\mathcal{B}_1$
$M_2 \leftarrow$ matchDetections$(\mathcal{B}_2, \mathcal{B}_1)$
// keep matched bounding boxes after two-way matching
for *match* m_1 *in* M_1 **do**
  for *match* m_2 *in* M_2 **do**
    if $m_1 == m_2$ **then**
      $M \leftarrow$ two-way matched-detection (b_1, b_2)
    end
  end
end
// compute GASD shape descriptor for matched bounding boxes from $Model_1$ and $Model_2$
for *match m in M* **do**
  $desc_1(d_1) \leftarrow$ compute shape descriptor
  $desc_2(d_2) \leftarrow$ compute shape descriptor
  // predict distance for shape descriptors using Random Forest Regressor (RFR)
  distance($desc_1(d_1)$) $\leftarrow$ RFR(b_1)
  distance($desc_2(d_2)$) $\leftarrow$ RFR(d_2)
  if *distance(*$desc_1(b_1)$*) ¡ distance(*$desc_2(b_2)$ **then**
    $\mathcal{B}_f \leftarrow$ add bounding boxes b_1
  end
  else
    $\mathcal{B}_f \leftarrow$ add bounding boxes b_1
  end
end
return $\mathcal{B}_f$

Algorithm 4: The complete workflow of adaptive learning based prediction-selection model

5.3.2 Shape Descriptor Computation

Extraction of 3D features or descriptors is one of the extremely important processes in 3D applications that has a big impact on how well descriptive outcomes work overall. The geometric structure should be captured by a strong and discriminative descriptor that is also simultaneous translation, scaling, and rotation invariant. It is one of the research areas of study that merits in by providing many useful 3D descriptor [96] from 3D point clouds, particularly in environments with occlusion and clutter.

The three primary categories of 3D descriptors now used in literature are global-based descriptors, local-based descriptors, and hybrid-based descriptors. In general, the first method estimates a single descriptor vector that encodes the entire input 3D object. The success of global descriptors depends on the ability to observe the full geometry of object point clouds, which proves to be a little more challenging. While utilizing appropriate keypoint-extraction methods to extract geometrical data from the local vicinity of each key point in the point cloud, local descriptors generate features. And the local descriptors are robust to occlusion and clutter [97] which the global descriptors are not. However, the local descriptors are sensitive to the changes in the neighborhoods around keypoints [98]. The hybrid-based descriptors combine the fundamental theorem of local and global descriptors or combine both types of descriptors to maximize the benefits of both local and global features.

As global descriptors, which demand comparatively less computing time and memory footprint, encode the geometric data of the entire 3D point cloud of the object and in general, 3D object identification, geometric classification, and shape retrieval all require global descriptors we choose the global descriptor as a feature for input to our adaptive learning based prediction-selection model. We propose to use the Globally Aligned Spatial Distribution (GASD) [45] global descriptor for efficient object recognition and pose estimation. The wide application of the GASD descriptor is for the task

of 3D object recognition and it outperforms ESF, VFH, and CVFH descriptors.

5.3.2.1 Globally Aligned Spatial Distribution

In order to align an entire point cloud representing an instance of an object with the canonical coordinate system, GASD relies on the estimation of a reference frame. The aligned point cloud is then given a descriptor based on the spatial distribution of its 3D points. The color distribution over the aligned point cloud may also be added to the descriptor. For computing object pose, global alignment transforms of matching point clouds are applied.

A 3D point cloud that represents a partial perspective of an item is the input for the GASD global description method. To align the point cloud to the canonical coordinate system and make the descriptor pose invariant, the initial step entails estimating a reference frame for the point cloud. Principal Component Analysis (PCA) is used to estimate the reference frame. The first stage entails computing the centroid of the set of 3D points, or P_i, which is the origin of the reference frame and represents a partial perspective of an object with i=1,..., n. Then, using P_i and $\overline{\mathbb{P}}$, a covariance matrix $\mathbf{C}$ is generated with x and z axis are the eigenvectors v_1, v_3 corresponding to the minimal and maximal eigenvalues, respectively and y-axis is $v_2 = v_1 \times v_3$. Then, the whole point cloud is aligned with this reference frame. Next, an axis-aligned bounding cube of point cloud is partitioned into $m_s \times m_s \times m_s$ cells. Concatenating the histograms created by adding the number of points that fall into each grid results in the global descriptor. The shape descriptor computed can also be combined with color information based on HSV space to increase discriminative power. However, we do not use color information for our shape descriptor computation.

5.3.3 Random Forest Regressor

We compute the GASD shape descriptor for the points lying inside each of the bounding boxes. We use the GASD descriptor as the independent variable and the statistical distance between the GASD descriptors of predicted and ground truth bounding boxes as the dependent variable based on which the bounding box selection is done. Random forest regressor (RFR) algorithm is used to regress the statistical distance value. For training the RFR model, we perform the one-way matching between the predicted bounding box of a single model and the ground truth object bounding box. For each match, we compute the GASD descriptor consisting of a 512 vector histogram for both the predicted bounding box of the model and the ground truth object bounding box. We further compute the distances between the two shape descriptor histograms of detected and ground truth bounding boxes and between the detections of two models as targets for training and testing our regression model. We also calculate statistical measures directly between the detections instead of shape descriptors histograms. The distances used between the two detections are as follows:

- L^1, Manhattan, or City Block Distance between histograms

$$D_{L1} = \sum_i |h_1(i) - h_2(i)|$$

- L^2 or Euclidean Distance between histograms

$$D_{L2} = \sqrt{\sum_i (h_1(i) - h_2(i))^2}$$

- L_∞ or Chebyshev Distance between histograms

$$D_{L\infty} = \max_i |h_1(i) - h_2(i)|$$

- Direct bird-eye view center distance of detections (distance between x-y only)

$$D_C = \sqrt{\sum_i (c_x y1(i) - c_x y2(i))^2}$$

- Difference between the volume of bounding boxes *i.e.* v=l*w*h of detections

$$D_V = |v_1(i) - v_2(i)|$$

- Bhattacharyya distance between histograms

$$D_{BH} = \sqrt{1 - \sum_i \sqrt{h_1(i) h_2(i)}}$$

- Earth Mover's distance between histograms

$$D_{EM} = \frac{\min_{f_{ij}} \sum_{i,j} f_{ij} A_{ij}}{sum_{i,j} f_{ij}}$$

$$\sum_j f_{ij} \leq h_1(i), \sum_j f_{ij} \leq h_2(j), \sum_{i,j} f_{ij} = \min \left(\sum_i h_1(i) \sum_j h_2(j) \right)$$

- Hellinger distance between histograms

$$D_H = \frac{1}{\sqrt{2}} \cdot \| \sqrt{h_1(i)} - \sqrt{h_2(i)} \|_2$$

The statistical distances for a feature matching is used to evaluate the similarity of two probability distributions between predicted and ground truth bounding box shape descriptors. It aids in quantifying the "closeness" of two statistical samples. We train our random forest regression model with input of the predicted bounding box shape descriptor and the statistical distance between the predicted and ground truth bounding box as the target value.

For our adaptive learning prediction-selection model, we use the trained regression

model to regress the distance value. Based on the closest(lowest) distance the detection corresponding to it is selected and added to the final list of detections. The overall workflow of the adaptive learning based prediction-selection model is captured in Algorithm 4.

5.4 Experiments & Results

LiDAR point clouds with required annotations for object detection for multiple classes must be included in the input dataset for our adaptive model. nuScenes [5] is a good choice for our validation data in that aspect.

5.4.1 Dataset

The nuScenes dataset [5] has been used for multiple benchmarks such as LiDAR segmentation, object detection, tracking, etc. This dataset contains 1000 scenes from 15 hours of driving data with 20-second long sequences with 700, 150, and 150 sequences for training, validation, and testing. This large-scale dataset provides data from the entire sensor suite, including six cameras, one LiDAR, five radars, localization info via GPS, and IMU data. It includes 1.4M camera images, 390k LiDAR sweeps, 1.4M RADAR sweeps, and 1.4M object bounding boxes in 40k keyframes. The dataset showcases good diversity with data from two diverse cities, left versus right-hand traffic, interesting driving maneuvers, everyday traffic situations, and unexpected behaviors. The dataset also ensures data alignment between sensors and cameras by calibrating the extrinsic and intrinsic of every sensor.

The dataset only includes box annotations every ten frames, but each LiDAR frame has calibrated vehicle pose information (0.5s). A 32 lanes LiDAR used by nuScenes generates about 30k points every frame. There are 28k, 6k, and 6k annotated frames

for training, validation, and testing. Ten classes having a long-tail distribution are included in the annotations. An average of the classes serves as the formal measurement of evaluation. Mean Average Precision (mAP) and nuScenes detection score are the primary metrics for 3D detection (NDS). Instead of the usual box-overlap, the mAP uses a bird-eye-view center distance of 0.5m, 1m, 2m, and 4m. Translation, scale, orientation, velocity, and other box properties are included in the NDS, a weighted average of mAP, and other attributes measurements. We train our adaptive selection model on the training data of 700 sequences and evaluate our approach on the 6k frames of 150 sequences of validation data.

5.4.2 Experimental Setup

To test our adaptive model, we need to have a set of models with comparative detection results to select among the detections. Some of the results from good older models have been improved to a great extent by some of the recent models. Many recent models have improved results by utilizing multi-modality input. CenterPoint [4] which takes only LiDAR point cloud data as input, serves as an ideal choice for an adaptive model where we focus on improving the results with only 3D point cloud data input. We use CenterPoint-VoxelNet and CenterPoint-PointPillars as our 3D object detection models for comparative results. CenterPoint-Voxel use a (0.1m, 0.1m, 0.2m) voxel size and CenterPoint-Pillars use a (0.2m, 0.2m) grid. For experiments on nuScenes, CenterPoint sets the detection range to [-51.2m, 51.2m] for the X and Y axis, and [-5m, 3m] for the Z axis.

We use the training and validation data for training and validating our random forest regressor model and validate the results of the adaptive learning based prediction-selection on the validation data. The training time for the random forest regressor model is approximately a few hours while the inference time per frame for our adaptive

learning-based selection model is approximately $\sim$0.8 seconds. The inference times are measured on an RTX 3060 GPU with 12GB memory.

5.4.3 Results

To find the similarity between the probability distributions of shape descriptors, we experiment with different statistical measures to identify the most suitable measure for our Random Forest Regression (RFR) model. We use the R_2 (coefficient of determination) as the regression score function to validate the accuracy and precision of our trained model. The best possible score is 1.0, which can be negative (because the model can be arbitrarily worse). In the general case when the actual y is non-constant, a constant model that always predicts the average y disregarding the input features, would get a score of 0.0. We experiment with L^1 norm, L^2 norm, Chebyshev distance, Direct bird-eye view center distance of detections (distance between x-y only), Difference between the volume of boxes, Bhattacharyya distance, Earth Mover's distance, and Hellinger distance as described in section 5.3.3 between the two shape descriptor histograms of detected and ground truth bounding boxes and between the detections of two models as targets for training and testing our regression model, respectively.

For the matched bounding boxes, we take the shape descriptor vector histogram as input and train and test our regression model with the target statistical measure. Results of our experiments with different statistical measures and its corresponding R_2 for our test data are shown in Table TC5.1. The results show that the highest R_2 (coefficient of determination) score is obtained with Bhattacharyya distance. This shows that the Bhattacharyya distance provides more information on the similarity between the probability distributions of shape descriptors from model detections and ground truth bounding boxes.

We evaluate our adaptive learning based prediction-selection model with CenterPoint-

Table TC5.1: R_2 score on test data with respect to different statistical measures.

Statistical Measure	R_2 score	
	CenterPoint-Voxelnet	**CenterPoint-PointPillars**
L^1-Norm	0.384	0.372
Direct bird-eye view center distance	0.152	0.154
Difference between volumes of boxes	0.014	0.024
Bhattacharyya distance	**0.490**	**0.472**
Earth Mover's Distance	0.355	0.349
L^2-Norm	0.446	0.439
Chebyshev distance	0.385	0.372
Hellinger distance	0.468	0.455

VoxelNet and CenterPoint-PointPillars models as detection models and experiment with a distance threshold of 4.0 for considering a two-way bounding box match and no distance threshold (distance threshold of ∞) used *i.e.* all the closest box matches irrespective of its distance are considered as a match in both one-way and two-way matching. The distance threshold indicates the minimum distance required to consider the closest bounding box match between the center of the two bounding boxes. The bounding box match process with distance threshold is shown in Algorithm 4. The average metrics scores across all classes are shown in Table TC5.2. The metrics used for evaluation of object detection task with ground truth as described in [5] are :

- **mean Average Precision (mAP):** The well-known Average Precision metric is used, but a match is defined by considering the bird-eye-view 2D center distance on the ground plane rather than the intersection over union-based affinities. Specifically, we match predictions with the ground truth objects with the smallest center distance up to a certain threshold. For a given match, average threshold precision (AP) is calculated by integrating the recall vs. precision curve for recalls and precision > 0.1. We finally average over match thresholds of 0.5, 1, 2, 4 meters and compute the mean across classes.
- **Average Translation Error (ATE):** Euclidean center distance in 2D in meters.
- **Average Scale Error (ASE):** Calculated as 1 - IOU after aligning centers and

Table TC5.2: Mean score for different for all classes when using different settings of distance threshold for match consideration for two-way matching to filter out bounding boxes on nuScenes [5] validation data.

Metrics	Detection Model Backbone		Aggregation Strategy	
	CenterPoint-VoxelNet	**CenterPoint-PointPillars**	**Adaptive Selection**	**Weighted Box Fusion**
Filtering matches without any distance threshold				
mAP	**0.547**	0.497	0.529	**0.550**
mATE	**0.280**	0.298	0.287	**0.275**
mASE	**0.251**	0.256	**0.254**	0.423
mAOE	**0.269**	0.365	**0.301**	0.474
mAVE	**0.267**	0.326	0.283	**0.267**
mAAE	**0.196**	0.200	0.198	**0.196**
NDS	**0.647**	0.604	**0.632**	0.612
Filtering matches with distance threshold of 4.0m				
mAP	**0.559**	0.507	0.541	**0.565**
mATE	**0.281**	0.302	0.290	**0.279**
mASE	**0.252**	0.257	**0.254**	0.425
mAOE	**0.275**	0.369	**0.305**	0.479
mAVE	**0.266**	0.326	0.282	**0.268**
mAAE	**0.195**	0.198	0.196	**0.194**
NDS	**0.652**	0.608	**0.638**	0.618

orientation.

- **Average Orientation Error (AOE):** Smallest yaw angle difference between prediction and ground-truth in radians. Orientation error is evaluated at 360 degrees for all classes except barriers, where it is only evaluated at 180 degrees. Orientation errors for cones are ignored.

- **Average Velocity Error (AVE):** Absolute velocity error in m/s. Velocity errors for barriers and cones are ignored.

- **Average Attribute Error (AAE):** Calculated as 1 - acc, where acc is the attribute classification accuracy. Attribute errors for barriers and cones are ignored.

- **nuScenes detection score (NDS):** Consolidation of the above metrics by computing a weighted sum: mAP, mATE, mASE, mAOE, mAVE, and mAAE. As a first step the TP errors are converted to TP scores as TP_score = max(1 - TP_error, 0.0). Then the weight of 5 to mAP and 1 to each of the 5 TP scores are assigned,

Table TC5.3: Class-wise evaluation when using two-way matching for filtering out bounding boxes but without any distance threshold.

Object Class	AP	ATE	ASE	AOE	AVE	AAE
Using CenterPoint-Voxelnet backbone for detection						
car	0.838	0.176	0.153	0.100	0.314	0.187
truck	0.559	0.314	0.175	0.081	0.243	0.233
bus	0.679	0.318	0.177	0.035	0.418	0.280
trailer	0.362	0.516	0.200	0.435	0.225	0.186
construction_vehicle	0.161	0.704	0.429	0.847	0.12	0.297
pedestrian	0.778	0.136	0.274	0.354	0.211	0.086
motorcycle	0.542	0.178	0.235	0.210	0.392	0.278
bicycle	0.353	0.142	0.259	0.297	0.211	0.022
traffic_cone	0.594	0.128	0.321	nan	nan	nan
barrier	0.605	0.186	0.289	0.065	nan	nan
Using CenterPoint-PointPillars backbone for detection						
car	0.828	0.185	0.156	0.145	0.328	0.199
truck	0.528	0.327	0.187	0.141	0.273	0.246
bus	0.636	0.341	0.183	0.050	0.567	0.309
trailer	0.326	0.582	0.217	0.551	0.191	0.131
construction_vehicle	0.140	0.600	0.412	1.122	0.137	0.387
pedestrian	0.756	0.152	0.275	0.389	0.228	0.087
motorcycle	0.455	0.207	0.237	0.318	0.650	0.209
bicycle	0.209	0.179	0.266	0.483	0.234	0.032
traffic_cone	0.528	0.162	0.334	nan	nan	nan
barrier	0.561	0.25	0.289	0.084	nan	nan
Using our proposed adaptive selection model						
car	0.834	0.180	**0.155**	**0.114**	0.321	**0.193**
truck	0.550	0.313	**0.180**	**0.099**	0.253	**0.237**
bus	0.659	0.329	**0.178**	**0.043**	0.451	**0.275**
trailer	0.349	**0.527**	**0.203**	**0.494**	0.223	**0.179**
construction_vehicle	0.153	0.694	**0.430**	**0.996**	0.126	**0.332**
pedestrian	0.770	0.143	**0.275**	**0.363**	0.217	0.085
motorcycle	0.513	0.187	**0.236**	**0.232**	**0.458**	0.264
bicycle	0.301	0.153	**0.263**	**0.308**	0.214	**0.023**
traffic_cone	0.573	0.139	**0.325**	nan	nan	nan
barrier	0.589	0.205	**0.290**	**0.055**	nan	nan
Using Weighted Box Fusion						
car	**0.842**	**0.169**	0.175	0.359	**0.293**	0.193
truck	**0.576**	**0.294**	0.191	0.294	**0.232**	0.241
bus	**0.677**	**0.308**	0.179	0.278	**0.433**	0.303
trailer	**0.378**	0.541	0.278	0.515	**0.185**	0.157
construction_vehicle	**0.187**	**0.648**	0.564	1.090	**0.124**	0.347
pedestrian	**0.776**	**0.134**	0.664	0.528	**0.202**	**0.077**
motorcycle	**0.538**	**0.180**	0.278	0.475	0.467	**0.221**
bicycle	**0.341**	**0.149**	0.324	0.492	**0.199**	0.026
traffic_cone	**0.588**	**0.134**	0.732	nan	nan	nan
barrier	**0.597**	**0.196**	0.848	0.238	nan	nan

and the normalized sum is calculated.

To verify our approach, we evaluate the predictions only for the matched bounding boxes for each model detection separately and compare it against our adaptive model where either of the detection is picked. This is done to cross-verify the results only for the filtered bounding boxes obtained with the existing object detection model, as only those boxes would be considered as detections in our adaptive model approach. We also present the class-wise results in Table TC5.3 and TC5.4 and the AP evaluation metric based on the a bird-eye-view centre distance of 0.5m, 1m, 2m, and 4m for the experiments in Table TC5.5 and TC5.6. The results of WBF are also shown in these tables for comparison.

The overall results of our adaptive model shown in Table TC5.2 show improvement over the CenterPoint-PointPillars model, and as the CenterPoint-VoxelNet performs better than the existing model, the adaptive model should select the detections majorly from the detections of CenterPoint-VoxelNet. The results show that our proposed model shows the expected behavior with close results to those of CenterPoint-VoxelNet. The mAP score is higher with the weighted boxes fusion strategy whereas the mASE and mAOE errors are high as compared to the adaptive selection strategy, and due to high errors, the NDS score is high for the adaptive selection strategy. Also, we can deduce that filtering the matches with a distance threshold of 4.0m results in an improvement of the mAP score by approximately 1%, but the change in the NDS score is minimal.

Similar results are also observed in Table TC5.3 and TC5.4 across different classes. These tables show variations in AP score for both adaptive selection strategy and weighted box fusion strategy as compared to the CenterPoint-VoxelNet across different classes are $<1\%$; however, the ASE error scores for classes pedestrian, traffic-cone, and barrier and the AOE error scores for all the classes except trailer are very high with weighted boxes fusion approach as compared to our proposed adaptive selection approach.

Table TC5.4: Class-wise evaluation when using two-way matching for filtering out bounding boxes with distance threshold of 4.0m.

Object Class	AP	ATE	ASE	AOE	AVE	AAE
Using CenterPoint-Voxelnet backbone for detection						
car	0.845	0.177	0.154	0.102	0.314	0.187
truck	0.576	0.316	0.177	0.086	0.247	0.230
bus	0.694	0.320	0.177	0.036	0.421	0.279
trailer	0.369	0.517	0.200	0.440	0.224	0.183
construction_vehicle	0.165	0.699	0.430	0.853	0.121	0.295
pedestrian	0.795	0.137	0.274	0.358	0.212	0.087
motorcycle	0.552	0.181	0.236	0.214	0.388	0.275
bicycle	0.365	0.147	0.262	0.323	0.204	0.023
traffic_cone	0.613	0.130	0.322	nan	nan	nan
barrier	0.616	0.188	0.289	0.065	nan	nan
Using CenterPoint-PointPillars backbone for detection						
car	0.835	0.186	0.156	0.147	0.329	0.199
truck	0.541	0.330	0.189	0.145	0.276	0.246
bus	0.654	0.344	0.183	0.051	0.570	0.309
trailer	0.336	0.584	0.217	0.559	0.192	0.130
construction_vehicle	0.145	0.608	0.419	1.101	0.135	0.379
pedestrian	0.771	0.154	0.276	0.391	0.228	0.088
motorcycle	0.464	0.212	0.238	0.332	0.648	0.206
bicycle	0.215	0.187	0.269	0.509	0.234	0.030
traffic_cone	0.539	0.166	0.336	nan	nan	nan
barrier	0.571	0.252	0.289	0.085	nan	nan
Using our proposed adaptive selection model						
car	0.840	0.181	**0.155**	**0.116**	0.321	**0.193**
truck	0.566	0.316	**0.181**	**0.104**	0.256	**0.236**
bus	0.677	0.331	**0.178**	**0.043**	0.454	**0.275**
trailer	0.357	**0.530**	**0.204**	**0.504**	0.220	0.174
construction_vehicle	0.157	0.695	**0.433**	**0.984**	**0.124**	**0.324**
pedestrian	0.785	0.144	**0.275**	**0.366**	0.217	0.086
motorcycle	0.526	0.192	**0.237**	**0.244**	**0.455**	0.261
bicycle	0.312	0.160	**0.266**	**0.331**	0.209	0.022
traffic_cone	0.590	0.142	**0.327**	nan	nan	nan
barrier	0.602	0.208	**0.290**	**0.055**	nan	nan
Using Weighted Box Fusion						
car	**0.852**	**0.170**	0.176	0.361	**0.294**	0.193
truck	**0.595**	**0.297**	0.194	0.295	**0.234**	0.239
bus	**0.699**	**0.312**	0.180	0.280	**0.437**	0.302
trailer	**0.388**	0.543	0.279	0.522	**0.185**	**0.154**
construction_vehicle	**0.197**	**0.657**	0.565	1.081	0.124	0.344
pedestrian	**0.796**	**0.136**	0.665	0.531	**0.202**	**0.078**
motorcycle	**0.553**	**0.184**	0.281	0.485	0.467	**0.217**
bicycle	**0.353**	**0.154**	0.332	0.516	**0.196**	**0.026**
traffic_cone	**0.609**	**0.136**	0.733	nan	nan	nan
barrier	**0.610**	**0.197**	0.848	0.238	nan	nan

Table TC5.5: Class-wise average precision (AP) when considering different bird-eye-view centre distance thresholds with respect to the ground truth, and using two-way matching for filtering out bounding boxes without any distance threshold.

Object Class	AP at Distance Threshold of 0.5m	1.0m	2.0m	4.0m	mean AP
Using CenterPoint-VoxelNet backbone for detection					
car	75.30	84.17	87.21	88.37	83.76
truck	38.90	55.47	62.87	66.18	55.85
construction_vehicle	1.82	9.82	22.69	30.13	16.12
bus	45.36	68.79	77.57	79.84	67.89
trailer	9.23	31.89	46.88	56.64	36.16
barrier	52.40	60.32	63.69	65.73	60.54
motorcycle	48.57	54.83	56.15	57.19	54.18
bicycle	34.24	35.44	35.56	35.82	35.27
pedestrian	75.34	76.93	78.45	80.61	77.83
traffic_cone	55.69	57.61	60.29	64.17	59.44
Using CenterPoint-PointPillars backbone for detection					
car	73.10	83.52	86.65	87.86	82.78
truck	35.21	53.24	59.88	62.99	52.83
construction_vehicle	2.40	10.75	18.55	24.16	13.97
bus	40.35	62.81	73.89	77.54	63.65
trailer	7.17	26.33	43.63	53.15	32.57
barrier	44.23	56.32	60.90	63.04	56.12
motorcycle	39.80	46.43	47.43	48.31	45.50
bicycle	20.09	20.90	21.01	21.43	20.86
pedestrian	72.79	74.46	76.80	78.30	75.59
traffic_cone	48.44	51.11	53.77	57.92	52.81
Using our proposed adaptive selection model					
car	74.65	83.94	86.97	88.17	83.43
truck	37.89	55.37	61.69	64.97	54.98
construction_vehicle	1.82	9.99	21.13	28.39	15.33
bus	42.30	66.38	76.04	78.76	65.87
trailer	8.61	30.13	44.96	55.90	34.90
barrier	49.71	58.54	62.67	64.86	58.94
motorcycle	44.96	52.20	53.43	54.57	51.27
bicycle	29.14	30.23	30.37	30.61	30.09
pedestrian	73.97	76.24	77.88	80.07	77.04
traffic_cone	53.05	55.52	58.03	62.51	57.23
Using Weighted Box Fusion					
car	75.87	84.65	87.60	88.70	84.20
truck	41.55	57.62	63.99	67.12	57.57
construction_vehicle	1.95	13.31	26.24	33.43	18.73
bus	45.99	67.83	77.80	79.23	67.72
trailer	10.82	33.09	49.53	57.70	37.78
barrier	50.88	59.33	63.24	65.30	59.69
motorcycle	48.06	54.32	55.73	57.07	53.79
bicycle	32.92	34.14	34.43	34.93	34.10
pedestrian	75.09	76.57	78.76	80.15	77.64
traffic_cone	55.13	57.07	59.62	63.47	58.82

Table TC5.6: Class-wise average precision (AP) when considering different bird-eye-view centre distance thresholds with respect to the ground truth, and using two-way matching for filtering out bounding boxes with distance threshold of 4.0m.

Object Class	AP at Distance Threshold of 0.5m	1.0m	2.0m	4.0m	mean AP
Using CenterPoint-VoxelNet backbone for detection					
car	75.95	84.91	87.98	89.15	84.50
truck	40.10	57.03	64.86	68.31	57.57
construction_vehicle	1.82	10.32	23.02	30.77	16.48
bus	46.17	70.38	79.46	81.78	69.44
trailer	9.30	32.45	47.61	58.04	36.85
barrier	53.47	61.03	64.93	67.02	61.62
motorcycle	49.27	56.04	57.34	58.31	55.24
bicycle	35.46	36.62	36.77	36.97	36.45
pedestrian	76.80	78.43	80.49	82.14	79.47
traffic_cone	57.63	59.57	62.06	66.07	61.33
Using CenterPoint-PointPillars backbone for detection					
car	73.65	84.21	87.40	88.64	83.48
truck	35.92	54.48	61.53	64.48	54.10
construction_vehicle	2.50	10.99	19.17	25.17	14.46
bus	41.15	64.58	75.97	79.74	65.36
trailer	7.34	27.10	45.03	55.11	33.64
barrier	44.72	57.41	62.09	64.33	57.14
motorcycle	40.40	47.36	48.48	49.37	46.40
bicycle	20.65	21.53	21.64	22.01	21.46
pedestrian	73.95	76.14	78.05	80.15	77.07
traffic_cone	49.46	52.16	54.93	59.20	53.94
Using our proposed adaptive selection model					
car	74.72	84.68	87.74	88.96	84.02
truck	38.85	56.89	63.69	67.06	56.62
construction_vehicle	1.84	10.33	21.63	29.06	15.72
bus	43.59	68.19	78.18	80.95	67.73
trailer	8.74	30.64	46.12	57.46	35.74
barrier	50.75	59.79	63.95	66.19	60.17
motorcycle	45.94	53.65	54.90	55.87	52.59
bicycle	30.22	31.36	31.54	31.71	31.21
pedestrian	75.41	77.63	79.34	81.56	78.48
traffic_cone	54.57	57.24	59.91	64.18	58.98
Using Weighted Box Fusion					
car	76.62	86.05	88.46	89.55	85.17
truck	42.64	59.66	66.17	69.34	59.45
construction_vehicle	2.13	14.20	28.04	34.59	19.74
bus	47.24	69.73	80.60	82.05	69.91
trailer	11.11	34.05	50.82	59.35	38.83
barrier	52.05	60.69	64.63	66.71	61.02
motorcycle	49.28	56.24	57.42	58.42	55.34
bicycle	34.30	35.53	35.62	35.91	35.34
pedestrian	76.66	78.70	80.85	82.35	79.64
traffic_cone	56.92	59.17	61.81	65.62	60.88

Table TC5.5 and TC5.6 show that the AP scores based on the bird-eye-view center distance of 0.5m, 1m, 2m, and 4m with the weighted boxes fusion approach are high for all the classes except bicycle as compared to other methods, whereas our adaptive selection method shows the variation of $>2\%$ for classes bus, motorcycle and bicycle and small variations in AP scores as compared to existing CenterPoint-VoxelNet model which performs better than the CenterPoint-PointPillars model implying our proposed adaptive model selected the detections majorly from the detections of CenterPoint-VoxelNet.

However, in the absence of only LiDAR input 3D object detection models, which have better results for some classes while other detection models would be dominant in those classes, we cannot thoroughly verify our approach. But for our selected models, the metrics scores were close and comparable, and even in this scenario, our adaptive model could select the most appropriate detections.

5.5 Summary

We have proposed and implemented an adaptive learning based prediction-selection model, an alternative to pick better detection among the detections of an ensemble of models to boost the 3D object detection without any extra stacked ensemble model which would work for different types of 3D object detection models with a 3D automotive LiDAR point clouds as input. We analyze different distances used for finding the closest neighbor to ground truth based on the GASD shape descriptor. Supervised learning using RFR is used for distance prediction based on shape descriptor. We use the spatial bird-eye-view center distance to find the closest match and propose a two-way matching process. We also analyze the evaluation for filtered bounding boxes with two experiments *i.e.* (i) where we use the distance threshold for match consideration and (ii) no distance threshold considered for match registration. The results show im-

provement over the CenterPoint-PointPillars model, and as the CenterPoint-VoxelNet performs better than the CenterPoint-PointPillars model, the results of the adaptive model are closer to the results of CenterPoint-VoxelNet as expected. Our experiments on nuScenes yielded promising results, which have been quantitatively verified.

CHAPTER 6

CONCLUSIONS

We performed spatio-temporal analysis on three critical perception tasks to aid autonomous driving, and we also proposed various methods for each task that could benefit automated systems. We have suggested a supervised machine learning technique and manually extracted spatial features based on the semantics of the data for the ground and non-ground classification as the foundation of the perception task. The performance of the suggested classification algorithm enhances the distinction between ground and non-ground objects. The outlier elimination phase with the temporal registration process aids in removing many points, reducing the calculation of features for those points, and has shown to be effective for better feature computation, leading to better classification outcomes. We chose the best features at single and multi-scales through tests with various feature subsets and neighborhood analysis. Our method doesn't need a complex deep learning network or a lot of training. However, because features must be extracted at each scale, multi-scale feature generation often takes longer than single-scale feature generation. This can be further improved by using a quicker computation model. We chose the best features at single and multi-scales through tests with various feature subsets and neighborhood analysis. Our method doesn't need a complex deep learning network or a lot of training. However, because features must be extracted at each scale, multi-scale feature generation often takes longer than single-scale feature generation. This can be further improved by using a quicker computation model.

Our road surface extraction method proposed using ground detection, edge detection, and mesh generation, tackles the challenges in extracting the surface for the navigable/drivable area and serves as the state-of-the-art for coarser road surface extraction in an automated system with usage of the spatio-temporal data across a sequence of frames. Our novel frame classification can classify the road structures using the transfer learning process with a high level of accuracy. Meanwhile, edge detection and smoothing work with a low error rate for continuous segments of straight roads. The proposed edge detection and smoothing approach are only suitable for straight roads. The approach fails when there is a large number of contiguous non-straight segments.

The information recorded in the shape descriptors for each identified object is provided to the stacked selection model structure by our adaptive learning prediction model for improving 3D object detection. The two-way matching procedure ensures that the bounding boxes are preserved and have a high degree of confidence because they have been detected in the same spatial domain across both models. To accurately determine the similarity of the histogram distributions, the Bhattacharyya distance performs significantly better than the other distances. The weighted boxes fusion, a substitute for improving detections from an ensemble of models, has superior improvements in terms of accuracy than our adaptive model, while the outcomes for our adaptive selection model have low errors.

6.1 Future Work

Several optimizations, tests, and experiments are in the scope of future work (i.e. the experiments with real-time results and experiments which require data in different conditions and which are time-consuming). Future work will focus on deeper analysis of specific mechanisms, new proposals to test different methods, and optimization of proposed methods.

6.1.1 Ground and Non-Ground Classification

As discussed in Chapter 3, our classification method lacks real-time computation, our road surface extraction has limitations because it can only be applied to straight roads rather than curved or turning roads, and our adaptive learning was unable to outperform the weighted boxes fusion improvements, laying the foundation for further study. The size of the point cloud and the number of features that must be retrieved influence how quickly the desired characteristics may be computed. A deep learning model, on the other hand, would be considerably faster for inference, but it needs a lot of training and computational capacity. Therefore, an intermediate strategy that would speed up the computation without requiring significant training or high-computational power might produce real-time results for the classification of the ground and non-ground.

6.1.2 Road Surface Extraction Using 3D LiDAR Point Cloud Sequences

As discussed in Chapter 4, our road surface extraction has limitations because it can only be applied to straight roads rather than curved or turning roads. The road surface extraction at curves or turns affects the concise road surface extraction at curves or turns, which may lead to bad decisions being made by the automated system proposed based on this perception analysis. The road surface extraction at curves or turns is equally essential as on straight roads. In these situations, a road edge point extraction and filtering method for all road structures will significantly increase the system's robustness.

6.1.3 Boosting 3D Object Detection

The adaptive learning-based prediction selection model proposed in Chapter 5 increases detection relative to the lower benchmark set by one of the models and bases its decisions on the separation between the shape descriptor and real object descriptor. To adequately assess the success of our adaptive-learning model, however, the detections from models that produce superior results for specific classes while other models perform well in the other classes and in different environments need to be further assessed.

6.2 Summary

Spatio-temporal analysis was performed on three critical perception tasks to aid autonomous driving, and the proposed different methods for each task demonstrated improvements and novelty that could benefit automated systems. As the foundation of the perception task, the proposed supervised machine learning technique and manually extracted spatial features based on the semantics of the data produced better results for ground and non-ground classification. Our road surface extraction method, which uses ground and non-ground classification, is quite unique and effective for contiguous straight roads. Furthermore, our proposed adaptive selection model for improving 3D object detection differs from existing approaches and yields promising results for improving the performance of existing methods. To improve perception for more significant assistance in autonomous driving, future work will include developing a holistic system that integrates all of these perceptual tasks with an interactive tool.

Bibliography

[1] Dhvani Katkoria and Jaya Sreevalsan-Nair. RoSELS: Road Surface Extraction for 3D Automotive LiDAR Point Cloud Sequence. In *Proceedings of the 3rd International Conference on Deep Learning Theory and Applications (DeLTA)*, pages 55–67. SciTePress, 2022. doi : https://doi.org/10.5220/0011301700003277.

[2] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9297–9307, 2019. doi : https://doi.org/10.48550/arXiv.1904.01416.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. doi : https://doi.org/10.1109/CVPR.2016.90.

[4] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *CVPR*, 2021. doi : https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.01161.

[5] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the*

IEEE/CVF conference on computer vision and pattern recognition, pages 11621–11631, 2020. doi : https://doi.org/10.1109/cvpr42600.2020.01164.

[6] Roman Solovyev, Weimin Wang, and Tatiana Gabruseva. Weighted boxes fusion: Ensembling boxes from different object detection models. *Image and Vision Computing*, 107:104117, 2021. doi : https://doi.org/10.1016/j.imavis.2021.104117.

[7] Scott Drew Pendleton, Hans Andersen, Xinxin Du, Xiaotong Shen, Malika Meghjani, You Hong Eng, Daniela Rus, and Marcelo H Ang Jr. Perception, planning, control, and coordination for autonomous vehicles. *Machines*, 5(1):6, 2017. doi : https://doi.org/10.3390/machines5010006.

[8] Jens Rieken and Markus Maurer. A lidar-based real-time capable 3d perception system for automated driving in urban domains. *arXiv preprint arXiv:2005.03404*, 2020. doi : https://doi.org/10.48550/arXiv.2005.03404.

[9] Peide Wang. Research on comparison of lidar and camera in autonomous driving. In *Journal of Physics: Conference Series*, volume 2093, page 012032. IOP Publishing, 2021. doi : http://dx.doi.org/10.1088/1742-6596/2093/1/012032.

[10] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006. doi : https://doi.org/10.1007/978-3-540-73429-1_1.

[11] Alireza Asvadi, Paulo Peixoto, and Urbano Nunes. Detection and tracking of moving objects using 2.5 d motion grids. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 788–793. IEEE, 2015. doi : https://doi.org/10.1109/ITSC.2015.133.

[12] Anna Petrovskaya and Sebastian Thrun. Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots*, 26(2):123–139, 2009. doi : https://doi.org/10.1007/s10514-009-9115-1.

[13] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008. doi : https://doi.org/10.1007/978-3-642-03991-1_3.

[14] Michael Himmelsbach, Felix V Hundelshausen, and H-J Wuensche. Fast segmentation of 3d point clouds for ground vehicles. In *2010 IEEE Intelligent Vehicles Symposium*, pages 560–565. IEEE, 2010. doi : https://doi.org/10.1109/IVS.2010.5548059.

[15] John Leonard, Jonathan How, Seth Teller, Mitch Berger, Stefan Campbell, Gaston Fiore, Luke Fletcher, Emilio Frazzoli, Albert Huang, Sertac Karaman, et al. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, 2008. doi : https://doi.org/10.1007/978-3-642-03991-1_5.

[16] E Schnebele, BF Tanyu, G Cervone, and N Waters. Review of remote sensing methodologies for pavement management and assessment. *European Transport Research Review*, 7(2):1–19, 2015. doi : https://doi.org/10.1007/s12544-015-0156-6.

[17] Zhao Liu, Jinling Wang, and Daxue Liu. A new curb detection method for unmanned ground vehicles using 2d sequential laser data. *Sensors*, 13(1):1102–1120, 2013. doi : https://doi.org/10.3390/s130101102.

[18] Sayanan Sivaraman and Mohan Manubhai Trivedi. Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis.

IEEE transactions on intelligent transportation systems, 14(4):1773–1795, 2013. doi : https://doi.org/10.1109/TITS.2013.2266661.

[19] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):743–761, 2011. doi : https://doi.org/10.1109/TPAMI.2011.155.

[20] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013. doi : https://doi.org/10.1007/s11263-013-0620-5.

[21] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European conference on computer vision*, pages 391–405. Springer, 2014. doi : https://doi.org/10.1007/978-3-319-10602-1_26.

[22] Mehul Arora, Louis Wiesmann, Xieyuanli Chen, and Cyrill Stachniss. Mapping the Static Parts of Dynamic Scenes from 3D LiDAR Point Clouds Exploiting Ground Segmentation. In *2021 European Conference on Mobile Robots (ECMR)*, pages 1–6. IEEE, 2021. doi : https://doi.org/10.1109/ECMR50962.2021.9568799.

[23] Zhenchao Ouyang, Xiaoyun Dong, Jiahe Cui, Jianwei Niu, and Mohsen Guizani. PV-EncoNet: Fast Object Detection Based on Colored Point Cloud. *IEEE Transactions on Intelligent Transportation Systems*, Early Access:1–12, 2021. doi : https://doi.org/10.1109/TITS.2021.3114062.

[24] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. doi : https://doi.org/10.1023/A:1010933404324.

[25] Martin Weinmann, Boris Jutzi, Stefan Hinz, and Clément Mallet. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and

efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105:286–304, 2015. doi : https://doi.org/10.1016/j.isprsjprs.2015.01.016.

[26] Jaya Sreevalsan-Nair and Pragyan Mohapatra. Influence of Aleatoric Uncertainty on Semantic Classification of Airborne Lidar Point Clouds: A Case Study with Random Forest Classifier Using Multiscale Features. In *IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium*, pages 1066–1069. IEEE, 2020. doi : https://doi.org/10.1109/igarss39084.2020.9323409.

[27] Jian Li, Jianjun Qian, and Yuhui Zheng. Ensemble r-fcn for object detection. In *Advances in Computer Science and Ubiquitous Computing*, pages 400–406. Springer, 2017. doi : https://doi.org/10.1007/978-981-10-7605-3_66.

[28] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018. doi : http://dx.doi.org/10.1109/CVPR.2018.00644.

[29] Jinsu Lee, Sang-Kwang Lee, and Seong-Il Yang. An ensemble method of cnn models for object detection. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 898–901. IEEE, 2018. doi : https://doi.org/10.1109/ICTC.2018.8539396.

[30] Nguyen D Vo, Khanh Nguyen, Tam V Nguyen, and Khang Nguyen. Ensemble of deep object detectors for page object detection. In *Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication*, pages 1–6, 2018. doi : https://doi.org/10.1145/3164541.3164644.

[31] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. doi : http://dx.doi.org/10.1109/CVPR.2016.91.

[32] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4507–4515, 2017. doi : https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.685.

[33] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms–improving object detection with one line of code. In *Proceedings of the IEEE international conference on computer vision*, pages 5561–5569, 2017. doi : https://doi.org/10.1109/ICCV.2017.593.

[34] Anshul Paigwar, Özgür Erkent, David Sierra-Gonzalez, and Christian Laugier. Gndnet: Fast ground plane estimation and point cloud segmentation for autonomous vehicles. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2150–2156. IEEE, 2020. doi : https://doi.org/10.1109/IROS45743.2020.9340979.

[35] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020. doi : https://doi.org/10.1109/TPAMI.2020.3005434.

[36] Leonardo Gigli, B Ravi Kiran, Thomas Paul, Andrés Serna, Nagarjuna Vemuri, Beatriz Marcotegui, and Santiago Velasco-Forero. Road Segmentation on low resolution LIDAR point clouds for autonomous vehicles. In *XXIV International Society for Photogrammetry and Remote Sensing Congress*, Nice, France, Aug 2020. ISPRS 2020.

[37] Zhihao Shen, Huawei Liang, Linglong Lin, Zhiling Wang, Weixin Huang, and Jie Yu. Fast Ground Segmentation for 3D LiDAR Point Cloud Based on Jump-Convolution-Process. *Remote Sensing*, 13(16):3239, 2021. doi : https://doi.org/10.3390/rs13163239.

[38] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220. IEEE, 2019. doi : https://doi.org/10.1109/IROS40897.2019.8967762.

[39] Tiago Cortinhal, George Tzelepi, and Eren Aksoy. SalsaNext: Fast, Uncertainty-aware Semantic Segmentation of LiDAR Point Clouds for Autonomous Driving. In *15th International Symposium, ISVC 2020, San Diego, CA, USA, October 5–7, 2020*, volume 12510. Springer, 2021. doi : https://doi.org/10.48550/arXiv.2003.03653.

[40] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020. doi : https://doi.org/10.48550/arXiv.1911.11236.

[41] Christoph B Rist, David Schmidt, Markus Enzweiler, and Dariu M Gavrila. SCSSnet: Learning Spatially-Conditioned Scene Segmentation on LiDAR Point Clouds. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1086–1093. IEEE, 2020. doi : https://doi.org/10.1109/IV47402.2020.9304824.

[42] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009. doi : https://doi.org/10.1109/ROBOT.2009.5152473.

[43] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2155–2162. IEEE, 2010. doi : https://doi.org/10.1109/IROS.2010.5651280.

[44] Aitor Aldoma, Markus Vincze, Nico Blodow, David Gossow, Suat Gedikli, Radu Bogdan Rusu, and Gary Bradski. Cad-model recognition and 6dof pose estimation using 3d cues. In *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, pages 585–592. IEEE, 2011. doi : https://doi.org/10.1109/ICCVW.2011.6130296.

[45] Joao Paulo Silva do Monte Lima and Veronica Teichrieb. An efficient global point cloud descriptor for object recognition and pose estimation. In *2016 29th SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*, pages 56–63. IEEE, 2016. doi : https://doi.org/10.1109/SIBGRAPI.2016.017.

[46] Leyang Zhao, Li Yan, and Xiaolin Meng. The Extraction of Street Curbs from Mobile Laser Scanning Data in Urban Areas. *Remote Sensing*, 13(12):2407, 2021. doi : https://doi.org/10.3390/rs13122407.

[47] Lichun Sui, Jianfeng Zhu, Mianqing Zhong, Xue Wang, and Junmei Kang. Extraction of road boundary from MLS data using laser scanner ground trajectory. *Open Geosciences*, 13(1):690–704, 2021. doi : https://doi.org/10.1515/geo-2020-0264.

[48] Inna Stainvas and Yosi Buda. Performance evaluation for curb detection problem. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 25–30. IEEE, 2014. doi : https://doi.org/10.1109/IVS.2014.6856617.

[49] Zhao Liu, Daxue Liu, Tongtong Chen, and Chongyang Wei. Curb detection using 2D range data in a campus environment. In *2013 Seventh International Conference on Image and Graphics*, pages 291–296. IEEE, 2013. doi :https://doi.org/10.1109/ICIG.2013.64.

[50] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Jürgen Gall, and Cyrill Stachniss. Towards 3d lidar-based semantic scene understanding of 3d point cloud sequences: The semantickitti dataset. *The*

International Journal of Robotics Research, 40(8-9):959–967, 2021. doi : https://doi.org/10.1177/02783649211006735.

[51] Zhao Zhou, Yingbin Zheng, Hao Ye, Jian Pu, and Gufei Sun. Satellite image scene classification via ConvNet with context aggregation. In *Pacific Rim Conference on Multimedia*, pages 329–339. Springer, 2018. doi : https://doi.org/10.1007/978-3-030-00767-6_31.

[52] Grant J Scott, Matthew R England, William A Starms, Richard A Marcum, and Curt H Davis. Training deep convolutional neural networks for land–cover classification of high-resolution imagery. *IEEE Geoscience and Remote Sensing Letters*, 14(4):549–553, 2017. doi : https://doi.org/10.1109/LGRS.2017.2657778.

[53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. doi : https://doi.org/10.1145/3065386.

[54] Xieyuanli Chen, Ignacio Vizzo, Thomas Läbe, Jens Behley, and Cyrill Stachniss. Range image-based LiDAR localization for autonomous vehicles. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5802–5808. IEEE, 2021. doi : https://doi.org/10.1109/ICRA48506.2021.9561335.

[55] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286, 2019. doi : https://doi.org/10.1109/ICCV.2019.00937.

[56] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019. doi : https://doi.org/10.48550/arXiv.1812.04244.

[57] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018. doi : https://doi.org/10.48550/arXiv.1711.08488.

[58] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 244–253, 2018. doi : https://doi.org/10.1109/CVPR.2018.00033.

[59] Jin Hyeok Yoo, Yecheol Kim, Jisong Kim, and Jun Won Choi. 3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection. In *European Conference on Computer Vision*, pages 720–736. Springer, 2020. doi : https://doi.org/10.1007/978-3-030-58583-9_43.

[60] Jiarong Wang, Ming Zhu, Bo Wang, Deyao Sun, Hua Wei, Changji Liu, and Haitao Nie. Kda3d: Key-point densification and multi-attention guidance for 3d object detection. *Remote Sensing*, 12(11):1895, 2020. doi : https://doi.org/10.3390/rs12111895.

[61] Burak Uzkent, Matthew J Hoffman, Anthony Vodacek, John P Kerekes, and Bin Chen. Feature matching and adaptive prediction models in an object tracking dddas. *Procedia Computer Science*, 18:1939–1948, 2013. doi : https://doi.org/10.1016/j.procs.2013.05.363.

[62] Satendra Singh and Jaya Sreevalsan-Nair. Adaptive multiscale feature extraction in a distributed system for semantic classification of airborne lidar point clouds. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2021. doi : https://doi.org/10.1109/LGRS.2021.3099935.

[63] Jérôme Demantké, Clément Mallet, Nicolas David, and Bruno Vallet. Dimensionality based scale selection in 3d lidar point clouds. In *Laserscanning*, 2011. doi : https://doi.org/10.5194/isprsarchives-XXXVIII-5-W12-97-2011.

[64] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975. doi : https://doi.org/10.1145/361002.361007.

[65] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948. doi : https://doi.org/10.1002/j.1538-7305.1948.tb01338.x.

[66] Jaya Sreevalsan-Nair and Beena Kumari. Local geometric descriptors for multi-scale probabilistic point classification of airborne lidar point clouds. In *Modeling, Analysis, and Visualization of Anisotropy*, pages 175–200. Springer, 2017. doi : https://doi.org/10.1007/978-3-319-61358-1_8.

[67] J Besl Paul and Neil D McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. doi : https://doi.org/10.1109/34.121791.

[68] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152. IEEE, 2001. doi : https://doi.org/10.1109/IM.2001.924423.

[69] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust Reconstruction of Indoor Scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5556–5565, 2015. doi : https://doi.org/10.1109/CVPR.2015.7299195.

[70] Martin Weinmann, Boris Jutzi, and Clément Mallet. Semantic 3D scene interpretation: A framework combining optimal neighborhood size selection with relevant

features. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2(3):181, 2014. doi : https://doi.org/10.5194/isprsannals-II-3-181-2014.

[71] Beena Kumari and Jaya Sreevalsan-Nair. An interactive visual analytic tool for semantic classification of 3D urban LiDAR point cloud. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 1–4, 2015. doi : https://doi.org/10.1145/2820783.2820863.

[72] D Ziou and S Tabbone. Edge Detection Techniques - an Overview. *Pattern Recognition and Image Analysis (Advances in Mathematical Theory and Applications)*, 8(4):537–559, 1998.

[73] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977. doi : https://doi.org/10.1111/j.2517-6161.1977.tb01600.x.

[74] Martin A Fischler and Robert C Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. doi : https://doi.org/10.1145/358669.358692.

[75] Jonathan Richard Shewchuk. Constrained Delaunay Tetrahedralizations and Provably Good Boundary Recovery. In *Eleventh International Meshing Roundtable (IMR)*, pages 193–204, 2002.

[76] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. doi : https://doi.org/10.48550/arXiv.1801.09847.

[77] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gram-

fort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013. doi : https://doi.org/10.48550/arXiv.1309.0238.

[78] François Chollet et al. Keras. `https://keras.io`, 2015.

[79] C. Bane Sullivan and Alexander Kaszynski. PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). *Journal of Open Source Software*, 4(37):1450, May 2019. doi : https://doi.org/10.21105/joss.01450.

[80] Andrea Tagliasacchi, Thomas Delame, Michela Spagnuolo, Nina Amenta, and Alexandru Telea. 3D Skeletons: A State-of-the-Art Report. In *Computer Graphics Forum*, volume 35, pages 573–597. Wiley Online Library, 2016. doi : https://doi.org/10.1111/cgf.12865.

[81] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Dsgn: Deep stereo geometry network for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12536–12545, 2020. doi : https://doi.org/10.1109/CVPR42600.2020.01255.

[82] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In *Conference on Robot Learning*, pages 923–932. PMLR, 2020. doi : https://dx.doi.org/10.23977/appep.2021.020113.

[83] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from

view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018. doi : https://doi.org/10.1109/IROS.2018.8594049.

[84] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11873–11882, 2020. doi : https://doi.org/10.1109/CVPR42600.2020.01189.

[85] Rui Qian, Xin Lai, and Xirong Li. 3d object detection for autonomous driving: a survey. *Pattern Recognition*, page 108796, 2022. doi : https://doi.org/10.1016/j.patcog.2022.108796.

[86] Wouter Van Gansbeke, Davy Neven, Bert De Brabandere, and Luc Van Gool. Sparse and noisy lidar completion with rgb guidance and uncertainty. In *2019 16th international conference on machine vision applications (MVA)*, pages 1–6. IEEE, 2019. doi : https://doi.org/10.48550/arXiv.1902.05356.

[87] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. doi : https://doi.org/10.5555/3295222.3295263.

[88] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11040–11048, 2020. doi : https://doi.org/10.1109/CVPR42600.2020.01105.

[89] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. doi : https://doi.org/10.3390/s18103337.

[90] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In

Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 12697–12705, 2019. doi : https://doi.org/10.1109/CVPR.2019.01298.

[91] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point r-cnn. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9775–9784, 2019. doi : http://dx.doi.org/10.1109/ICCV.2019.00987.

[92] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE transactions on pattern analysis and machine intelligence*, 43(8):2647–2664, 2020. doi : http://dx.doi.org/10.1109/TPAMI.2020.2977026.

[93] Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. *arXiv preprint arXiv:2102.00463*, 2021. doi : https://doi.org/10.48550/arXiv.2102.00463.

[94] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018. doi : https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00472.

[95] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019. doi : https://doi.org/10.48550/arXiv.1908.09492.

[96] Xian-Feng Hana, Jesse S Jin, Juan Xie, Ming-Jie Wang, and Wei Jiang. A comprehensive review of 3d point cloud descriptors. *arXiv preprint arXiv:1802.02297*, 2, 2018. doi : https://doi.org/10.48550/arXiv.1802.02297v2.

[97] Yulan Guo, Ferdous Sohel, Mohammed Bennamoun, Min Lu, and Jianwei Wan. Rotational projection statistics for 3d local surface description and object recognition. *International journal of computer vision*, 105(1):63–86, 2013. doi : https://doi.org/10.1007/s11263-013-0627-y.

[98] Isma Hadji and Guilherme N DeSouza. Local-to-global signature descriptor for 3d object recognition. In *Asian conference on computer vision*, pages 570–584. Springer, 2014. doi : http://dx.doi.org/10.1007/978-3-319-16628-5_41.