# LiDAR-based 3D Perception from Multi-frame Point Clouds for Autonomous Driving

by

Chengjie Huang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2025

© Chengjie Huang 2025

**Examining Committee Membership**

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner: François Pomerleau
Associate Professor, Dept. of Electrical and Computer Engineering
Université Laval

Supervisor: Krzysztof Czarnecki
Professor, Dept. of Electrical and Computer Engineering,
University of Waterloo

Internal Members: Yuri Boykov
Professor, Cheriton School of Computer Science
University of Waterloo

Hongyang Zhang
Assistant Professor, Cheriton School of Computer Science
University of Waterloo

Internal-External Member: Zhou Wang
Professor, Dept. of Electrical and Computer Engineering
University of Waterloo

## Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

## Statement of Contributions

The following publications are included in parts in this thesis:

[1] **Chengjie Huang**, Vahdat Abdelzad, Sean Sedwards, Krzysztof Czarnecki. VADet: Multi-frame LiDAR 3D Object Detection using Variable Aggregation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2025.

[2] **Chengjie Huang**, Krzysztof Czarnecki. MFSeg: Efficient Multi-frame 3D Semantic Segmentation. To appear in *IEEE International Conference on Robotics and Automation (ICRA)*. 2025.

[3] **Chengjie Huang**, Vahdat Abdelzad, Sean Sedwards, Krzysztof Czarnecki. SOAP: Cross-sensor Domain Adaptation for 3D Object Detection Using Stationary Object Aggregation Pseudo-labelling. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2024.

All three studies were conducted under the supervision of Prof. Krzysztof Czarnecki.

In [1], Vahdat Abdelzad and Sean Sedwards both contributed to discussions and assisted in proofreading the document. Vahdat Abdelzad expanded the limitations and extensions section, while Sean Sedwards helped with organizing the BibTeX entries and made various adjustments to the formatting, including refining the LaTeX code for Algorithm 1.

In [3], Vahdat Abdelzad and Sean Sedwards both provided input at each stage of the project. Sean Sedwards suggested the use of CDF to quantitatively demonstrate the effect of aggregation on scan patterns. Vahdat Abdelzad recommended several improvements to the post-processing pipeline, including the use of weighted box fusion. They both helped proofread the document and improve the writing.

## Abstract

3D perception is a critical component of autonomous driving systems, where accurately detecting objects and understanding the surrounding environment is essential for safety. Recent advances in Light Detection and Ranging (LiDAR) technology and deep neural network architectures have enabled state-of-the-art (SOTA) methods to achieve high performance in 3D object detection and segmentation tasks. Many approaches leverage the sequential nature of LiDAR data by aggregating multiple consecutive scans to generate dense multi-frame point clouds. However, the challenges and applications of multi-frame point clouds have not been fully explored. This thesis makes three key contributions to advance the understanding and application of multi-frame point clouds in 3D perception tasks.

First, we address the limitations of multi-frame point clouds in 3D object detection. Specifically, we observe that increasing the number of aggregated frames has diminishing returns and even performance degradation, due to objects responding differently to the number of aggregated frames. To overcome this performance trade-off, we propose an efficient adaptive method termed Variable Aggregation Detection (VADet). Instead of aggregating the entire scene using a fixed number of frames, VADet performs aggregation per object, with the number of frames determined by an object's observed properties, such as speed and point density. This adaptive approach reduces the inherent trade-offs of fixed aggregation, improving detection accuracy.

Next, we tackle the challenge of applying multi-frame point cloud to 3D semantic segmentation. Point-wise prediction on dense multi-frame point clouds can be computationally expensive, especially for SOTA transformer-based architectures. To address this issue, we propose MFSeg, an efficient multi-frame 3D semantic segmentation framework. MFSeg aggregates point cloud sequences at the feature level and regularizes the feature extraction and aggregation process to reduce computational overhead without compromising accuracy. Additionally, by employing a lightweight MLP-based point decoder, MFSeg eliminates the need to upsample redundant points from past frames, further improving efficiency.

Finally, we explore the use of multi-frame point clouds for cross-sensor domain adaptation. Based on the observation that multi-frame point clouds can weaken the distinct LiDAR scan patterns for stationary objects, we propose Stationary Object Aggregation Pseudo-labelling (SOAP) to generate high quality pseudo-labels for 3D object detection in a target domain. In contrast to the current SOTA in-domain practice of aggregating few input frames, SOAP utilizes entire sequences of point clouds to effectively reduce the sensor domain gap.

**Acknowledgements**

I would like to express my gratitude to everyone who has supported me throughout my PhD journey. Amid a global pandemic, this period has been both challenging and fulfilling.

First and foremost, I would like to thank my family. My mother, Jin Xiao, and my grandmother, Hemei Ouyang, raised me with love and care, and their guidance has shaped who I am today. I am also thankful to my family members, Ruoshui Wang and Qiang Wang, for their constant support and advice over the years, and to my father, Yuming Huang, whose financial assistance made my academic pursuits possible.

I am sincerely grateful to my professor, Krzysztof Cznarnecki, for providing me with the freedom to explore and encouraging me to learn at my own pace. Even if certain explorations did not turn out to be fruitful, I have learned immensely from each endeavor. I also wish to acknowledge Sean Sedwards and Vahdat Abdelzad for their guidance and insightful discussions throughout multiple projects. Their persistent questioning and keen attention to detail have profoundly influenced my approach to research, and thanks to Sean, I will be using CDFs for the rest of my life! My sincere thanks extend to Michal Antkiewicz, who expertly set up and maintained the servers crucial to our experiments, and to Prarthana Bhattacharyya for her invaluable advice and the opportunities she provided to contribute to her projects. I also appreciate the support and camaraderie of other members of the WISE lab I have met along the way, including Rick Salay, Mathiew Pitropov, Benjamin Therien, Luke Rowe, Christopher Gus Mannes, Adrian Chow, Mei Qi Tang, and Mozhgan Nasr Azadani.

Finally, I would like to thank my friends for their unwavering support. I am particularly grateful to Trevor Painter for his warm welcome when I first set foot in the United States and for helping me improve my English, and to my long-time friends Yijie Bai, Yunshu Xu, and Mengshi Zhang for supporting me through college and during the toughest times.

Thank you all for making this journey possible.

## Dedication

To my family, teachers, and friends.

# Table of Contents

**Examining Committee** ii

**Author's Declaration** iii

**Statement of Contributions** iv

**Abstract** v

**Acknowledgements** vi

**Dedication** vii

**List of Figures** xii

**List of Tables** xv

**1 Introduction** 1

**2 Background and Related Work** 6

2.1 LiDAR Sensor . . . . . 6

2.1.1 LiDAR Ranging Mechanisms . . . . . 7

2.1.2 LiDAR Imaging Architectures . . . . . 9

2.1.3 LiDAR Returns . . . . . 10

2.2 LiDAR Perception Datasets for Autonomous Driving . . . 11
2.2.1 KITTI Dataset . . . 11
2.2.2 nuScenes Dataset . . . 13
2.2.3 Waymo Open Dataset . . . 13
2.2.4 Other Datasets . . . 15
2.3 Machine Learning from Point Cloud Data . . . 15
2.3.1 Projection-based . . . 16
2.3.2 Voxel-base . . . 18
2.3.3 Point-based . . . 20
2.4 LiDAR-based 3D Object Detection . . . 23
2.4.1 Architectures . . . 24
2.4.2 Model Training . . . 26
2.4.3 Evaluation Metrics . . . 27
2.5 LiDAR-based 3D Semantic Segmentation . . . 29
2.5.1 Architectures . . . 29
2.5.2 Model Training . . . 30
2.5.3 Evaluation Metrics . . . 31
2.6 Multi-frame/sequential LiDAR perception . . . 31
2.7 Domain Adaptation in LiDAR perception . . . 33

**3 VADet: *Multi-frame LiDAR 3D Object Detection using Variable Aggregation* 34**
3.1 Introduction . . . 34
3.2 Method . . . 36
3.2.1 Random Aggregation Training . . . 36
3.2.2 Variable Aggregation . . . 37
3.3 Experimental Setup . . . 39
3.3.1 Dataset . . . 40
3.3.2 Evaluation Metrics . . . 40

3.3.3 Implementation Details . . . 42
3.4 Results and Analysis . . . 43
3.4.1 Overall Performance . . . 43
3.4.2 Qualitative Results . . . 46
3.4.3 Breakdown Analysis . . . 47
3.5 Limitations and Future Work . . . 50
3.6 Conclusion . . . 52

**4 MFSeg: *Efficient Multi-frame 3D Semantic Segmentation* 53**
4.1 Introduction . . . 53
4.2 Method . . . 55
4.2.1 Local Feature Extraction . . . 55
4.2.2 Feature Aggregation . . . 56
4.2.3 Point Decoder . . . 59
4.3 Experimental Setup . . . 60
4.3.1 Datasets . . . 60
4.3.2 Implementation Details . . . 60
4.4 Results . . . 61
4.4.1 Overall Performance . . . 61
4.4.2 Inference Latency . . . 63
4.4.3 Ablation Study . . . 64
4.5 Limitations and Future Work . . . 65
4.6 Conclusion . . . 66

**5 SOAP: *Cross-sensor Domain Adaptation for 3D Object Detection Using Stationary Object Aggregation Pseudo-labelling* 67**
5.1 Introduction . . . 67
5.2 Method . . . 70

5.2.1 Scene-level Full-sequence Aggregation . . . 71
5.2.2 Quasi-Stationary Training . . . 72
5.2.3 Spatial Consistency Post-processing . . . 74
5.2.4 SOAP Pseudo-labels . . . 74
5.3 Experimental Setup . . . 75
5.3.1 Datasets . . . 75
5.3.2 Point Cloud Input . . . 76
5.3.3 Implementation Details . . . 76
5.3.4 Evaluation . . . 77
5.4 Results . . . 77
5.4.1 Unsupervised Domain Adaptation . . . 77
5.4.2 Semi-supervised Domain Adaptation . . . 80
5.4.3 Speed Analysis . . . 82
5.4.4 Qualitative Results . . . 82
5.4.5 Ablation Study . . . 84
5.5 Limitations and Future Work . . . 84
5.6 Conclusion . . . 87

**6 Discussions and Conclusion** **88**

**References** **91**

# List of Figures


2.1 An overview of different LiDAR ranging mechanisms. . . . . 8

2.2 An overview of different LiDAR imaging architectures. . . . . 9

2.3 An example of a LiDAR point cloud with two returns from the Waymo dataset [218]. The blue points correspond to the first (primary) return, and the red points are from the second return. . . . . 11

2.4 Data collection platforms, sensor configurations, and sample camera and LiDAR data from the KITTI (top), nuScenes (middle), and Waymo (bottom) dataset. . . . 12

2.5 Projection-based methods compared to early multi-view projection methods for 3D shape classification. Renders in (a) are adapted from [217]. . . . . 17

2.6 Voxel-based methods using 3D (submanifold) 3D convolutions. . . . . 19

2.7 Standard convolution activates previously empty pixels (submanifold dilation), while submanifold convolution preserves the sparsity of the data. Visualizations are adapted from [71] . . . . . 19

2.8 Point-based point cloud processing methods. . . . . 21

2.9 VoxelNet/SECOND object detection pipeline. Subsequent works largely follow this modular design with different improvements. . . . . 24

2.10 3D object detection evaluation. . . . . 28

2.11 The U-Net architecture commonly used in 3D semantic segmentation methods. Depending on the input representation, the encoder and decoder blocks can be CNN-based or point-based. . . . . 29

3.1 Performance trade-off between stationary (<0.2 m/s), slow ([0.2,10) m/s), and fast (10 m/s) vehicles. The relative AP in (c) illustrates the improvement/degradation relative to using 3-frame fixed aggregation. (a) and (b) are examples of stationary and fast-moving vehicles after 16-frame fixed aggregation. . . . . 35

3.2 An example of the $\eta$ function computed with two factors: speed and density. . . . 38
3.3 Qualitative results comparing 3-frame and 16-frame fixed aggregation with VADet. Red and green bounding boxes are ground truth and predictions, respectively. Predictions are filtered with 0.5 confidence threshold for visual clarity. . . . . 46
3.4 AP vs. the number of frames for stationary ($<$0.2 m/s), slow ([0.2,10) m/s), and fast-moving ($\geq$10 m/s) vehicles. . . . . 47
3.5 AP vs. the number of frames for sparse ($<$2 pts/m$^2$), medium ((2,100] pts/m$^2$), and dense ($>$100 pts/m$^2$) dynamic vehicles. . . . . 48

4.1 Relative mIoU improvement between various levels of multi-frame point cloud concatenation and single-frame input. Performance are evaluated using a Sparse-UNet [72] on the Waymo dataset [218]. . . . . 54
4.2 Inference time of PTv3 [267] and MFSeg under a range of point cloud sequence length on the nuScenes dataset [20] (left) and Waymo dataset [218] (right). PTv3-ds denotes that the point cloud is downsampled using a voxel-grid with size 0.05 $\mathrm{m}^3$, while PTv3 is evaluated with minimum input downsampling. . . . . 55
4.3 An overview of the proposed MFSeg architecture. . . . . 56
4.4 The feature association process (Section 4.2.2). Previous frame LFE features first undergo ego-motion correction, then are jointly voxelized with the current frame features to establish association. Features that cannot be associated are paired with the zero vector $\mathbf{0}$. . . . . 58
4.5 The proposed training scheme and auxiliary loss for the LFE and the feature aggregator. . . . . 59
4.6 Segmentation results from PTv3 (left) and MFSeg (right). PTv3 misclassifies a partially occluded bicycle along with nearby background points as motorcycle (blue), while MFSeg accurately identifies the points belonging to the bicycle (red). 63
4.7 A breakdown of the latency of each component in MFSeg. . . . . 64

5.1 Scan lines are evident in point clouds when only few input frames are used (a,b), appearing as obvious modes in CDF plots that largely differ because of the modes (c). Aggregating many more frames removes visible scan lines (d,e) and makes CDFs for similar objects in different datasets more alike (f) . . . . . 68
5.2 Overview of Stationary Object Aggregation Pseudo-labelling (SOAP) pipeline. . 70

5.3 Example of a point cloud generated by SFA. Dynamic objects are distorted while stationary objects are densified. . . . . 71

5.4 Example of a quasi-stationary object. This object reached a maximum speed of 1.4 m/s with a total displacement of 3.9 m, and thus would be eliminated by naive filtering. . . . . 72

5.5 Examples of pseudo-labels generated by different methods in nuScenes → Waymo unsupervised domain adaptation setting, and the corresponding ground truth labels. Green represents true positive pseudo-labels, orange represents false positive pseudo-labels, and red represents false negative pseudo-labels. . . . . 85

5.6 Examples of pseudo-labels generated by different methods in nuScenes → Waymo semi-supervised domain adaptation setting. Green represents true positive pseudo-labels, orange represents false positive pseudo-labels, and red represents false negative pseudo-labels. . . . . 86

# List of Tables


3.1 Vehicle AP of a VoxelNeXt [33] detector trained with separate fixed configurations and RAT, evaluated with different input aggregations. . . . . 36

3.2 Dynamic vehicle AP of a CenterPoint model on the Waymo dataset using different subset evaluation metrics. . . . . 41

3.3 Overall vehicle AP on the Waymo validation split. The best performance is in bold and the second best is underlined. . . . . 42

3.4 Overall vehicle performance on the Waymo validation split compared with SOTA multi-frame detectors. The best performance according to APH is highlighted in bold and the second best is underlined. . . . . 44

3.5 Overall vehicle performance on the Waymo test split compared with other methods (without TTA or ensemble). The best performance according to APH is highlighted in bold and the second best is underlined. . . . . 45

3.6 Vehicle AP breakdown by speed. The best performance is highlighted in bold and the second best is underlined. . . . . 47

3.7 Dynamic vehicle AP performance breakdown by point cloud density. The best performance is highlighted in bold and the second best is underlined. . . . . 48

3.8 The percentage of Waymo vehicles in each speed-density subcategory. . . . . 49

3.9 AP performance breakdown by speed and point cloud density. The best performance is highlighted in bold. . . . . 51

4.1 Results on the nuScenes validation split. . . . . 62

4.2 Results on the Waymo validation split. . . . . 62

4.3 Ablation study on the NuScenes validation split without TTA. . . . . 65

5.1 Degradation ($\Delta$) w.r.t. in-domain performance of SOTA detector VoxelNeXt [33] in Waymo $\leftrightarrow$ nuScenes cross-sensor setting. Increasing the number of aggregated frames improves over single-frame input, but there is still a substantial performance gap. . . . . 69

5.2 Unsupervised domain adaptation results for nuScenes $\rightarrow$ Waymo, where Waymo dataset is unlabelled. . . . . 79

5.3 Unsupervised domain adaptation results for Waymo $\rightarrow$ nuScenes, where nuScenes dataset is unlabelled. The percentages represent the amount of the Direct–Oracle domain gap closed. . . . . 79

5.4 Semi-supervised domain adaptation results for nuScenes $\rightarrow$ Waymo, where 1% of Waymo data is labelled. The percentages represent the amount of the Direct–Oracle domain gap closed. . . . . 81

5.5 Semi-supervised domain adaptation results for Waymo $\rightarrow$ nuScenes, where 1% of nuScenes data is labelled. The percentages represent the amount of the Direct–Oracle domain gap closed. . . . . 81

5.6 Unsupervised domain adaptation results for nuScenes $\rightarrow$ Waymo, where Waymo dataset is unlabelled, split based on object speed. The percentages represent the amount of the Direct–Oracle domain gap closed. . . . . 83

5.7 Semi-supervised domain adaptation results for nuScenes $\rightarrow$ Waymo, where 1% of Waymo data is labelled, split based on object speed. The percentages represent the amount of the Direct–Oracle domain gap closed. . . . . 83

5.8 Ablation study results for nuScenes $\rightarrow$ Waymo unsupervised domain adaptation. The percentages represent the amount of domain gap closed relative to the 5-frame baseline detector. . . . . 84

# Chapter 1

# Introduction

Since the early days of automobiles, people have envisioned a future with autonomous vehicles that can safely take us wherever we wish with the press of a button. From Isaac Asimov's 1953 novel *Sally*, featuring an "automatobile" with sentient positronic motors [8], to the vision of a driverless highway in General Motors' 1956 short film *Key to the Future* [171], the dream of a fully autonomous future is driven not only by the desire for convenience but more importantly, by the pursuit of safety. In 2022, an estimated 5.93 million motor vehicle accidents occurred in the United States alone [243]. A recent survey suggests that 94% of the crashes can be traced back to driver-related issues, with the majority cases caused by recognition and decision-making errors [214, 242]. It is easy to imagine that with more advanced perception and decision-making systems, these driver-related accidents could be significantly reduced. As *Sally*'s editor put it, "this leaves us but one answer: make cars more intelligent than their drivers!" [8]

While our vehicles today, or any artificial intelligence for that matter, have yet to gain sentience like Asimov's murderous automatobiles, we have come much closer to the driverless future that previous generations envisioned. Initiatives such as the 1985 DARPA program on autonomous land vehicles, along with pioneering projects like Ernest Dickmann's VaMoRs [49] and CMU's Navlab [234], have driven the rapid development of autonomous driving with growing interest from both industry and academia. Thanks to the recent advances in machine learning [111] and in-vehicle AI hardware like NVIDIA DRIVE [41], the development and deployment of autonomous driving technologies have been accelerated to an unprecedented scale. What began as early testbeds capable of simple road following in the 1980s [234] has evolved into robotaxis driving alongside human drivers in complex real-world environment [149]. It is clear that self-driving is no longer confined to research labs with controlled testing environments but must now adapt to dynamic real-world settings.

To meet the growing demand and complexity of the driving conditions, the software that powers autonomous driving has grown significantly in complexity. It has progressed from the handcrafted classical algorithms [49] and simple neural networks with only a few layers [235], to today's highly intricate systems consisting of dozens of components and hundreds of thousands lines of code [241]. A traditional autonomous driving system (ADS) can be broken down into four components: (i) perception, (ii) prediction, (iii) planning, and (iv) control/actuation. Perception can take many forms, including object detection and semantic segmentation. Its main goal is for the ADS to perceive the environment and identify obstacles or other road users. Prediction involves predicting the future trajectory or occupancy states of other road users. The planning algorithm then determines the ego vehicle's future trajectories based on the current and future states. Finally, the control system is responsible for executing the planned trajectory.

In recent years, as deep learning-based methods started to outperform and replace traditional components, end-to-end ADS has started to gain traction [26]. Unlike the traditional architecture, where components are designed separately and handcrafted data (such as object bounding boxes) is used as input for downstream tasks, the end-to-end paradigm uses a differentiable architecture that takes sensory input and directly outputs planned trajectories or control signals. This has the potential to simplify software design, as evidenced by Tesla's recent shift to an end-to-end architecture, which reportedly eliminates a large portion of the codebase [241]. However, this simplicity comes with a trade-off: ensuring the safety of an end-to-end ADS remains a challenge, as individual components can no longer be tested independently. Currently, both architectural paradigms are in use across the industry. For example, Waymo employs a more traditional architecture, while Tesla's new FSD system has transitioned to an end-to-end design.

Regardless of the paradigm adopted, perception is a critical component for the overall performance of an ADS. Current ADS systems often employ a variety of sensors to perceive and interpret the environment, primarily camera, radar, and LiDAR. These sensors complement each other, covering a wide range of driving conditions.

Cameras provide high-definition and often colored images of the environment. They closely resemble human vision and are excellent at capturing object details. Because of this, companies like Tesla have favored a vision-only approach when designing their ADS, relying solely on camera input. However, due to the 3D-to-2D image projection, depth information is inevitably lost. While a stereoscopic setup or data-driven method can offer estimates [226, 289], accurately recovering 3D information from camera images alone remains a challenging problem. Additionally, as a passive sensor, camera performance can degrade significantly in environments with poor visibility (e.g., fog) or low illumination.

Radar is an active sensor that provides direct measurements of range and relative speed by analyzing the delay and frequency shift between the transmitted and received signal. It typically

has a long operating range and excels in challenging weather conditions, such as fog, rain, and snow, because the millimeter wave is capable of penetrating these elements. However, radar data tends to have lower resolution and is typically integrated with other sensors like cameras and LiDAR to provide a more comprehensive understanding of the environment.

LiDAR, which stands for **Li**ght **D**etection **A**nd **R**anging, is an active sensor that operates on similar principles to radar, but uses optical signals (typically infrared lasers) rather than millimeter radio waves. Compared to camera and radar, LiDAR provides high-resolution, accurate 3D range measurements, making it essential for safety-critical applications like autonomous driving.

*3D object detection* and *semantic segmentation* are two common perception tasks using LiDAR data. The goal of 3D object detection is to localize and classify objects in the scene using LiDAR point clouds. 3D rotated bounding boxes are used to represent the location, size, and orientation of objects. Most publicly available self-driving datasets [69, 218] include the three primary types of road users: vehicles, pedestrians, and cyclists. Some datasets provide more fine-grained categories [20, 25, 261] as well as rare object types such as horse and buggies [181]. In semantic segmentation, each point in the point cloud is classified. In addition to the object types shared with the 3D object detection task, semantic segmentation typically includes different types of surfaces, vegetation, and other structures such as buildings.

In the autonomous driving setting, LiDAR data is typically collected continuously at 10–20 Hz. While the aforementioned perception tasks can be performed on each LiDAR point cloud independently, the sequential nature of the scans offers opportunities to improve perception performance. One common approach to take advantage of this sequential data is by concatenating consecutive LiDAR point clouds after ego-motion correction, which we refer to as point cloud aggregation in this thesis. While many methods have adopted this simple technique, several challenges remain, and the applications of point cloud aggregation have not been fully explored.

In response to the existing challenges and to advance multi-frame perception methods, we consider the following three research problems and make the corresponding contributions:

(i) **Addressing the performance trade-offs in multi-frame 3D object detection**

In 3D object detection, the length of the point cloud sequence introduces inherent performance trade-offs, particularly between static and dynamic objects. Specifically, aggregating longer sequences improves static object detection but degrades dynamic object performance, results in diminishing returns or even degradation in the overall performance of the 3D object detector. This effect has been observed in several existing works [30, 292] but has not been adequately addressed to the best of our knowledge.

In the first part of the thesis (Chapter 3), we investigate this trade-off and present a novel method, "*VADet: multi-frame 3D object detection with variable aggregation.*" We ana-

lyze three representative object detection architectures with different backbone designs and demonstrate that this trade-off is present across all of them. Based on our findings, we propose variable aggregation as an alternative point cloud aggregation method, where the amount of aggregation each object receives is adaptively determined based on the object's properties. We conduct extensive experiments to show that VADet is architecture-agnostic, effectively addresses the performance trade-off, and achieves state-of-the-art results.

VADet is part of the following publication:
Chengjie Huang, Vahdat Abdelzad, Sean Sedwards, Krzysztof Czarnecki. VADet: Multi-frame LiDAR 3D Object Detection using Variable Aggregation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2025.

(ii) **Addressing the computational challenges in multi-frame semantic segmentation**

In 3D semantic segmentation, point-based point cloud processing with transformer backbone has recently demonstrated SOTA performance with the introduction of point transformers [266, 267, 311]. However, these models have not been well-explored in the multi-frame setting. Due to the computational cost of attention operations, they often scale poorly when aggregating multiple frames.

In the second part of the thesis (Chapter 4), we propose a multi-frame method, *MFSeg*, to enhance the SOTA point-based transformer model PTv3 [267]. To improve computational efficiency, we introduce several modification to the original architecture. Additionally, we propose an aggregation technique in the feature space by establishing an equivalence between input point cloud concatenation and feature space aggregation. We develop a novel aggregator design and a training strategy to support this equivalence. Experiments are conducted on multiple datasets to demonstrate the effectiveness of MFSeg.

MFSeg is part of the following publication:
Chengjie Huang, Krzysztof Czarnecki. MFSeg: Efficient Multi-frame 3D Semantic Segmentation. To appear in *IEEE International Conference on Robotics and Automation (ICRA)*. 2025.

(iii) **Exploring the use of long sequences for cross-sensor domain adaptation**

Pioneered by Offboard 3D [187], which established offline detection (also referred to as pseudo-labeling or auto-labelling) as an applications of long sequence aggregation, many subsequent works have refined and built upon this idea [61, 159, 281]. However, these methods typically utilize full sequence information only at the object level for prediction refinement. Moreover, other applications have not been explored outside of pseudo-labeling.

In the final part of the thesis (Chapter 5), we explore a novel application of long sequence for cross-sensor domain adaptation, and propose *"SOAP: Cross-sensor Domain Adapta-*

*tion for 3D Object Detection Using Stationary Object Aggregation Pseudo-labeling."* We observe that point cloud aggregation can reduce the distinct LiDAR scan patterns unique to each sensor, mitigating the domain gap that often leads to performance degradation. To this end, we propose the use of full sequence aggregation at the scene level. A novel training strategy and post-processing technique are developed to address the unique challenges associated with the full sequence aggregation and further improve performance. We perform a comprehensive evaluation to demonstrate that SOAP not only produces high-quality pseudo-labels for the target domain, but also enhances existing unsupervised and semi-supervised domain adaptation methods.

SOAP is part of the following publication:
Chengjie Huang, Vahdat Abdelzad, Sean Sedwards, Krzysztof Czarnecki. SOAP: Cross-sensor Domain Adaptation for 3D Object Detection Using Stationary Object Aggregation Pseudo-labelling. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2024.

In the remainder of the thesis, we will first present background information on relevant topics and review related works in Chapter 2. The three methods are then introduced in Chapters 3 to 5. Finally, the discussions and conclusion follow in Chapter 6.

# Chapter 2

# Background and Related Work

In this chapter, we provide a comprehensive overview of the background and related work in LiDAR and LiDAR-based perception. We begin with an introduction to the basics of LiDAR sensors in section 2.1. In section 2.2, we summarize existing LiDAR-equipped autonomous driving perception datasets, which are essential resources for training and evaluating perception models. Next, in section 2.3, we introduce deep learning methods for point cloud data. They serve as fundamental building blocks for the more complex LiDAR-based perception tasks. Specifically, we review existing works for 3D object detection and semantic segmentation, detailed in section 2.4 and section 2.5, respectively. In section 2.6, we present related works on multi-frame / sequential LiDAR perception, which often extend traditional single-frame methods. Finally, in section 2.7, we introduce the challenges and existing approaches in domain adaptation for LiDAR data.

## 2.1 LiDAR Sensor

LiDAR is an active sensing technology that provides precise and direct range measurement using lasers. It has proven valuable in a wide range of applications including agriculture, forestry, survey and mapping, and more recently, robotics, including autonomous driving. LiDAR sensors can employ different ranging mechanisms, use various imaging architectures, and produce different types of returns. These different aspects will be introduced in this section.

### 2.1.1 LiDAR Ranging Mechanisms

On a high level, a LiDAR system measures the range to a target by emitting a laser beam directed at the target, detecting the reflected beam, then calculating the range based on the signal delay. This process relies on various signal processing techniques, each offering different trade-offs in range, precision, and robustness in real-world applications. Existing LiDAR systems can be categorized into three types: Pulsed/Time-of-Flight (ToF), Amplitude-Modulated Continuous-Wave (AMCW), and Frequency-Modulated Continuous-Wave (FMCW).

#### Pulsed / Time-of-Flight (ToF) LiDAR

ToF is a simple technique for LiDAR ranging. As illustrated in Figure 2.1(a), a pulsed/ToF LiDAR uses a pulsed laser signal, where the laser emitter sends out pulses of light for the receiver to detect. After detection, the ToF of the light pulse can be measured and the range can be determined. Specifically, the distance to target $d$ is a function of the ToF $\Delta t$ and the speed of light $c$:

$$d = \frac{c \cdot \Delta t}{2}. \tag{2.1}$$

A ToF LiDAR often has a long range thanks to the high energy laser pulse, making it suitable for autonomous driving where objects often need to be identified ahead of time. However, since the detection is entirely contained in a single pulse, interference from other LiDAR operating in the vicinity [105] or adversarial systems such as LiDAR spoofers [22, 23, 178, 211] remains a challenge. Noteable developers of these LiDAR systems include Velodyne [247], Ouster [99], and Hesai [229]. Autonomous driving companies like Waymo [148] and Aeye [94] also developed proprietary LiDAR systems based on this method.

#### Amplitude-Modulated Continuous-Wave (AMCW) LiDAR

AMCW LiDAR uses a continuous wave with varying intensities. When a continuous AM signal is received, the phase shift $\Delta\varphi$ between the transmitted signal and the received signal can be determined via signal correlation, as shown in Figure 2.1 (b). Given the known frequency of the amplitude modulation $f$, the distance to target is given by:

$$d = \frac{c \cdot \Delta\varphi}{4\pi f}. \tag{2.2}$$

AMCW LiDAR can be efficiently implemented in compact designs [103, 106] as signal intensity can be easily modulated with change in current. Similar to ToF systems, AMCW LiDARs

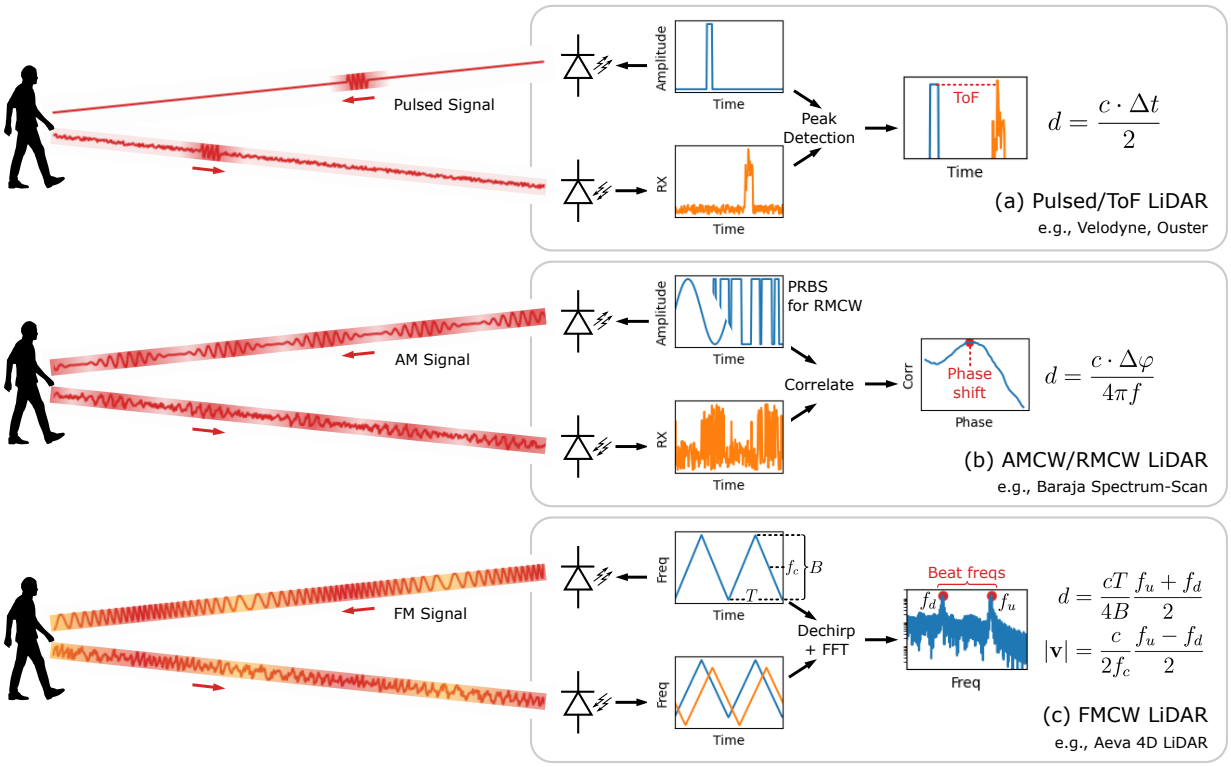



Figure 2.1: An overview of different LiDAR ranging mechanisms.

with simple sinusoidal or sawtooth modulation are also susceptible to mutual interference [90]. Random-Modulated Continuous-Wave (RMCW) [221] is an amplitude modulation technique that can alleviate this problem. In RMCW, the amplitude is modulated using a pseudo-random bit sequence (PRBS), as illustrated in Figure 2.1 (b). As the PRBS code is random and can be changed dynamically, RMCW has been shown to greatly reduce interference from other systems [90]. Baraja's proprietary Spectrum-Scan is a noteable example that capitalizes on this technique [183].

### Frequency-Modulated Continuous-Wave (FMCW) LiDAR

FMCW is a technique commonly used in Radar ranging, where a continuous signal is frequency-modulated. This allows the LiDAR system to not only determine the range, but also measure the Doppler speed of the target. As illustrated in Figure 2.1 (c), the transmitted signal is chirped, and the received signal from a distant moving object will be shifted both in frequency and in time.

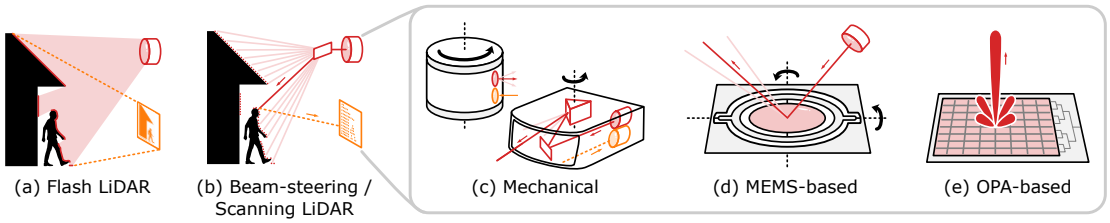



Figure 2.2: An overview of different LiDAR imaging architectures.

After mixing with the original reference signal, the beat frequencies $f_u, f_d$ can be identified to determine the range and relative speed of the target:

$$d = \frac{c \cdot T}{4B} \cdot \frac{f_u + f_d}{2}, \qquad |\mathbf{v}| = \frac{c}{2f_c} \cdot \frac{f_u - f_d}{2}. \tag{2.3}$$

While early FMCW research focuses on short-range, high-precision applications [13], recent advancements in LiDAR technology has enabled long-range FMCW LiDAR systems designed for robotic applications. Several emerging LiDAR manufacturers, such as SiLC [212], Aeva [93], and Baraja [151], as well as autonomous driving companies, like MobileEye [97] [1] and Aurora [95], have all developed FMCW LiDAR systems to bring the speed measurement capability to autonomous driving.

### 2.1.2 LiDAR Imaging Architectures

In addition to ranging techniques, LiDAR sensors vary significantly in how they handle imaging. Different architectures and configurations produce distinct scan patterns, which can influence the performance of downstream perception tasks, including those studied in this thesis. The method introduced in Chapter 5 aims to address the domain gap caused by the point clouds collected using different sensors.

Existing LiDAR sensors for automotive can be broadly categorized into two main types: flash and beam-steering/scanning LiDAR. A flash LiDAR illuminates the entire environment, as illustrated in Figure 2.2 (a), and a photodiode array is responsible for producing fine-grained range measurements within the LiDAR's FOV. However, since the laser power is distributed over the entire scene, they are often limited in range and resolution compared to the beams-steering alternative [13, 130]. Beam-steering/scanning LiDARs cover the FOV by procedurally scanning the environment, using various mechanisms to steer the laser beam (and in some cases

[1] At the time of writing, MobileEye has ended their internal FMCW LiDAR development. [96]

the receiver as well) (Figure 2.2 (b)). This can be achieved mechanically or through solid-state solutions.

Mechanical LiDARs (Figure 2.2 (c)) steer the laser beam by altering the optical paths using moving parts, such as rotating mirrors (e.g., Hesai AT series [230]) or moving the entire LiDAR assembly (e.g., Velodyne HDL series [246]). These systems often offer a large horizontal FOV, up to 360 degrees, making them common in autonomous driving applications. The datasets used in this thesis are collected using mechanical LiDARs mounted on the top of the vehicle, granting a full view of the surrounding environment.

In contrast, solid-state beam-steering LiDARs have no moving components to direct the laser beam. In these systems, beam-steering can be achieved through micro-electromechanical systems (MEMS) or optical phased array (OPA). MEMS-based LiDARs use electromechanical mirrors to direct light (Figure 2.2 (d)), while OPA-based LiDARs rely on phase interference within an optical phased array to steer the laser beams (Figure 2.2 (e)). Solid-state solutions are often more responsive and less prone to mechanical failure. However, they generally have a more limited FOV compared to mechanical systems. Many emerging LiDAR manufacturers have started to adopt solid-state technology, such as Aeva's Atlas [93] and Baraja's Spectrum-Scan [183].

### 2.1.3 LiDAR Returns

A LiDAR sensor provides a 3D coordinate for each laser pulse, determined by the azimuth, elevation, and corresponding range measurement. Commonly, *intensity/reflectivity* is recorded for each point to represent the reflectiveness of the target. For FMCW LiDAR systems, per-point velocity can be directly obtained in the same cycle, while ToF systems may require multiple measurements of the same location or rely on scene flow estimation to compute velocity. For pulsed/ToF LiDAR, certain models can also provide *elongation*, which measures the width of the received laser pulse relative to the normal width of the transmitted signal. This can help provide insights into the texture and transmittance of the target [218].

Each laser beam emitted by the LiDAR diverges over distance, creating a non-negligible cross-section and a small FOV. For example, the Velodyne HDL-64E LiDAR's laser can cover a $15\,\text{cm} \times 2\,\text{cm}$ region at 30 m [246]. As a result, each laser beam can be intercepted by multiple targets, producing multiple return signals. Recent LiDAR systems are often capable of reporting multiple returns per laser, commonly the strongest, first, or last return, which increases the number of points per scan. The Waymo dataset [218], which will be used extensively in this thesis, employs multiple returns to aid in various perception tasks. An example of a point cloud with two returns can be seen in Figure 2.3.

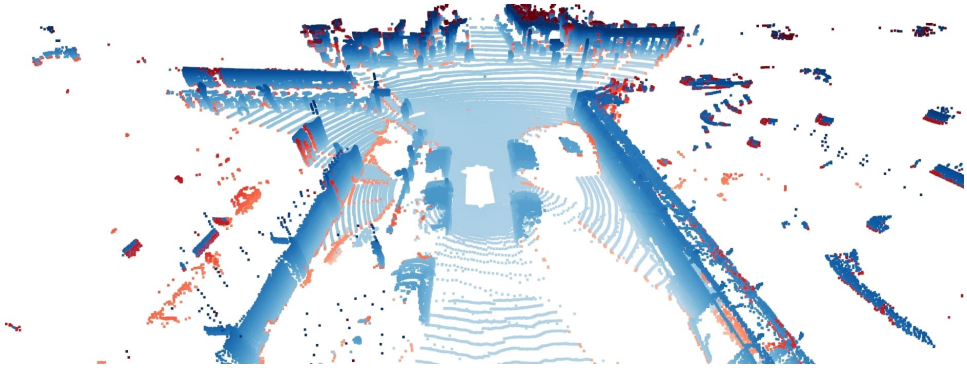


Figure 2.3: An example of a LiDAR point cloud with two returns from the Waymo dataset [218]. The blue points correspond to the first (primary) return, and the red points are from the second return.

## 2.2 LiDAR Perception Datasets for Autonomous Driving

With the growing interests in autonomous driving, several public datasets have been collected with LiDAR-equipped vehicles. In this section, we will take an in-depth look at three datasets: KITTI [69], nuScenes [20], and Waymo [218]. The nuScenes and Waymo datasets are used extensively in this thesis, while the KITTI dataset is included to provide context for the development of other datasets. Other notable datasets will be briefly mentioned, as they are not directly relevant to the work presented in this thesis.

### 2.2.1 KITTI Dataset

The KITTI dataset [69] is one of the first publicly available dataset for autonomous driving applications. Initially released in 2012, the KITTI dataset is collected in Karlsruhe, Germany. The KITTI data collection vehicle features four front-facing cameras in a stereoscopic setup and a top-mounted mechanical spinning LiDAR (Velodyne HDL-64E) operating at 10 Hz. This configuration provides the vehicle with a 90-degree camera field of view (FOV) and 360 degrees of 3D vision. This allows KITTI to establish a wide range of benchmarks for general computer vision tasks such as visual odometry, object/scene flow [166], and depth estimation [240], as well as scene understanding tasks including object detection, tracking, and semantic segmentation [1].

The KITTI 3D object detection benchmark uses a subset of the dataset consisting of 7,481 training samples and 7,518 test samples with held-back ground truth. Car, pedestrian, and cyclists are annotated with 3D rotated bounding boxes, covering three major types of road users within

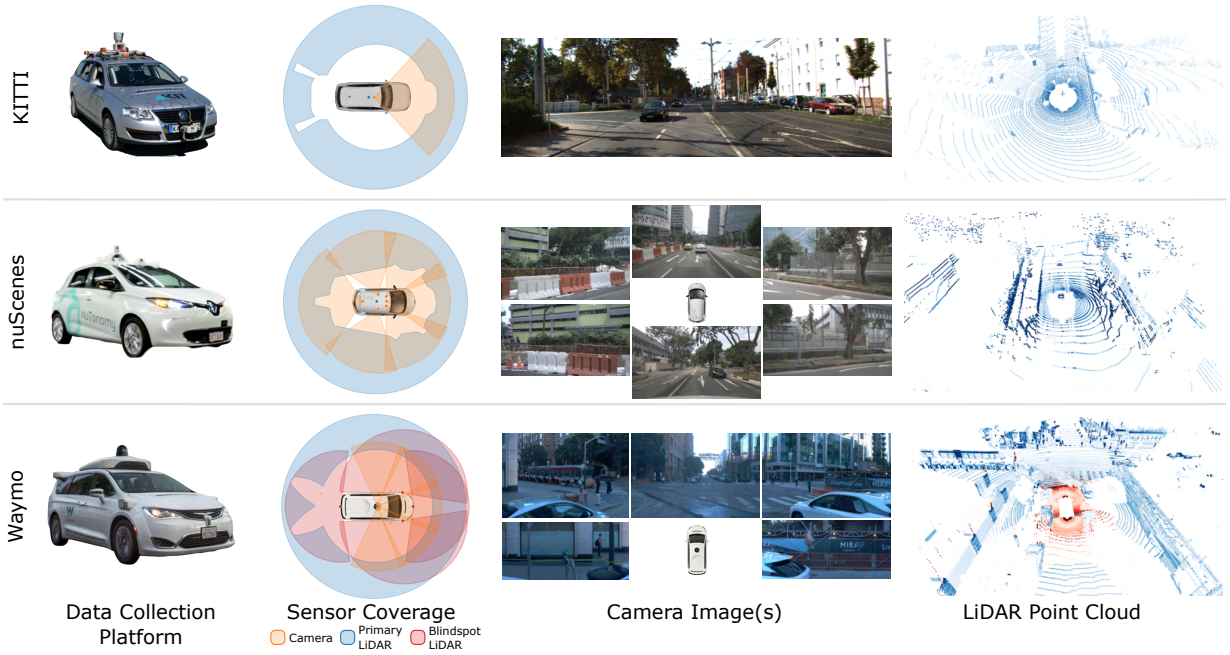



Figure 2.4: Data collection platforms, sensor configurations, and sample camera and LiDAR data from the KITTI (top), nuScenes (middle), and Waymo (bottom) dataset.

the dataset. Since the annotations are based on the front-facing cameras, only objects within the camera's POV are annotated. In total, there are around 200,000 annotations in the KITTI dataset. However, due to its relatively small size and limited FOV, recent works on 3D object detection have increasingly favored modern, large-scale datasets, two of which are featured later in this thesis.

SemanticKITTI [12] is proposed to extend the KITIT dataset for the 3D semantic segmentation task. It provides point cloud semantic annotations for the KITTI odometry dataset and is considerably larger than the 3D object detection subset, with 23,201 training samples and 20,351 test samples. To efficiently and consistently annotate the point clouds, SemanticKITTI utilizes SLAM to obtain the poses and aligns the point clouds in each sequence. Manual labelling is then performed on the aggregated point cloud consisting of multiple scans. This process is similar to our proposed pseudo-labelling method SOAP that will be introduced in Chapter 5.

While not used in the experiments of this thesis, the KITTI dataset is instrumental in advancing perception research for autonomous driving applications and in the development of the other datasets introduced later.

### 2.2.2 nuScenes Dataset

The nuScenes dataset [20] developed by nuTonomy [2] in 2019 is a considerably larger dataset than the KITTI dataset. It was collected in two separate locations, Boston, USA, and Singapore, providing greater variety, including left-hand driving scenarios.

The nuScenes data collection platform includes a surround camera setup consisting of six cameras, five radar sensors, and a top-mounted mechanical spinning LiDAR (Velodyne HDL-32E) operating at a higher 20 Hz. However, the Velodyne HDL-32E has a lower vertical resolution of only 32 laser beams, compared to KITTI's 64-beam LiDAR. The resulting point cloud is consequently much sparser, as can be seen in Figure 2.4. On average each LiDAR point cloud in the nuScenes dataset has 34K points, significantly less than the KITTI dataset (120K) and Waymo dataset (177K). This presents a unique challenge for the LiDAR-based perception tasks considered in this thesis.

The nuScenes dataset contains 1,000 sequences, with 750 sequences designated for training, 150 for validation, and the remaining 150 for testing with no ground truth available. Each sequence lasts approximately 20 seconds and is collected at a data rate of 20 Hz, resulting in roughly 400 samples per sequence. From each sequence, 40 key frames are selected uniformly and annotated, totaling to around 40K annotated LiDAR point clouds for training and testing. The nuScenes dataset provides annotations for 23 fine-grained classes with 1.4 million 3D bounding boxes, making it significantly larger and more comprehensive than the KITTI dataset.

The nuScenes dataset hosts benchmarks for variety of tasks with a stronger focus on autonomous driving applications than the KITTI dataset. These include 3D object detection, tracking, trajectory prediction, 3D semantic and panoptic segmentation. For the 3D object detection task, 10 classes are used for evaluation: car, truck, bus, trailer, construction vehicle, pedestrian, motorcycle, bicycle, traffic cone, and barrier. To enable more comprehensive performance evaluation of 3D object detectors, nuScenes uses a range of evaluation metrics, which will be detailed in Section 2.4.3. For the 3D semantic segmentation task, each method is evaluated on an additional 6 classes to include both objects and background features: terrain, vegetation, driveable surface, sidewalk, manmade, and other flat surfaces.

### 2.2.3 Waymo Open Dataset

The Waymo Open Dataset [218], also known as WOD or simply the Waymo dataset, is developed by Waymo and released in 2019. It is collected in three U.S. cities: Phoenix, Mountain View, and San Francisco, and provides a diverse set of driving conditions.

[2] After a number of acquisitions and merging, nuScenes is now managed by Motional, Inc.

The Waymo dataset is collected using the Waymo Driver, an autonomous driving platform developed in-house by Waymo. The vehicle is equipped with five cameras that cover 230 degrees of the front, although the rear of the vehicle is not covered by any camera. The Waymo Driver uses multiple proprietary LiDAR sensors, including one mechanical spinning LiDAR mounted on the top and four short-range LiDARs mounted around the vehicle to cover the blind spot. The top-mounted LiDAR, similar to the one used in the KITTI dataset, has 64-beam vertical resolution and operates at 10 Hz. Together with the four short-range perimeter LiDARs, this configuration provides more comprehensive coverage than the KITTI and nuScenes datasets, as exemplified in Figure 2.4.

Waymo's proprietary LiDAR has two unique features compared to the KITTI and nuScenes datasets. First, in addition to intensity, each point has an elongation channel that reflects the width of the laser pulse in the received optical signal. It is reported that an elongated return pulse with low intensity often indicates spurious object. Second, Waymo's LIDAR system supports two returns, as explained in the previous section on LiDAR returns Section 2.1.3. As a result, the LiDAR point clouds in the Waymo dataset are denser than those in the other datasets. These features can be favorable in various perception tasks, but at the same time pose computational challenges for real-time systems.

The Waymo dataset is slightly larger than the nuScenes dataset but much more densely annotated. It contains 1,150 sequences, each lasting 20 seconds, divided into 798 sequences for training, 202 for validation, and 150 for testing, with no ground truth available for the test set. The point clouds and images are collected at 10 Hz. Every point clouds ($\approx$200) in each sequence is annotated with 3D bounding boxes, but the semantic segmentation annotations are available at only 2 Hz for every five frames. In total, the Waymo dataset has 230K annotated frames with 12 million annotated 3D bounding boxes. This is more than $5\times$ the number of annotated frames and $8\times$ the number of 3D bounding boxes compared to the nuScenes dataset, making Waymo the largest of the three datasets.

The Waymo dataset is a popular benchmark for autonomous driving perception research thanks to its size and ongoing support. Annual challenges are hosted on various tasks, including 3D object detection and 3D semantic segmentation. For the 3D object detection task, the dataset provides annotations for vehicles, pedestrians, and cyclists. It is worth noting that the vehicle class in the Waymo dataset includes all types of vehicles, from small motorcycles to large trucks and buses, making it more challenging than other datasets. Like the nuScenes dataset, the Waymo benchmark introduces a custom metric to more comprehensively evaluate different methods by taking into account the bounding box headings, which will be discussed later in Section 2.4.3.

In 2024, Waymo introduced a 3D semantic segmentation benchmark, which includes more fine-grained annotations than the 3D object detection task. This benchmark evaluates 23 classes,

including small ground features such as curbs and lane markers. However, it is less popular than the well-established SemanticKITTI and nuScenes benchmarks.

### 2.2.4 Other Datasets

Alongside the release of nuScenes and Waymo, many other datasets of varying sizes have been developed by autonomous driving and automobile companies. Notable examples include the ApolloScape dataset [161] by Baidu, the Honda 3D (H3D) dataset [176] by Honda, the LEVEL5 dataset [104] by Lyft, the A2D2 dataset [70] by Audi, and the ONCE dataset [163] by Huawei. Additionally, several research groups have released datasets aimed at autonomous driving research, including KAIST's multi-spectral dataset [91], the A*3D dataset [179], Argoverse [25] and Argoverse 2 [261]. However, due to the lack of post-launch support (such as scripts for dataset processing and evaluation) or a comprehensive benchmark for 3D object detection and semantic segmentation, they did not gain as much traction for these tasks as the three datasets discussed earlier.

To facilitate the development and evaluation of robust LiDAR-based perception methods, some datasets focus on complex driving scenarios, such as the Indian Driving Dataset (IDD-3D) [52], and challenging weather conditions. The pioneering Seeing Through Fog (STF) dataset [18] presents a large amount of multi-modal with 3D bounding box annotations in multiple weather conditions. The SemanticSTF dataset [270] builds upon STF to support point cloud semantic segmentation. The Canadian Adverse Driving Conditions (CADC) dataset [181] focuses on snowy conditions of various intensities, with a recent extension (CADC+) introducing paired clear weather data to facilitate domain adaptation research [225]. The Waymo dataset [148] also includes a domain adaptation benchmark that consists of a large amount of unlabelled data collected in rainy weather.

## 2.3 Machine Learning from Point Cloud Data

To perform various perception tasks on LiDAR point clouds, deep learning is essential to achieve state-of-the-art (SOTA) performance [185]. Over the years, numerous deep learning approaches have been proposed to learn from point cloud data. Unlike tabular data or images, which are structured and ordered, a point cloud is a unstructured and unordered set of points, making it challenging to apply traditional deep learning methods. Existing works on point cloud learning use different strategies to overcome this challenge, many of which serve as fundamental building blocks for more complex architectures and challenging tasks. In this section, we will detail

different strategies for point cloud learning and introduce seminal works that lay the foundation for point cloud processing.

### 2.3.1 Projection-based

To address the challenges with processing 3D data, Su et al. [217] proposes a multi-view approach to project 3D shapes onto 2D images by rendering them from different viewpoints, as illustrated in Figure 2.5 (a). It became popular for 3D shape recognition and retrieval as it allows the use of standard 2D convolutional neural networks (CNNs) to process 3D data with competitive performance. Many subsequent works build upon this concept with various improvements [27, 57, 65, 77, 102, 184, 251, 260, 275, 291, 300]. While not designed for more complex tasks, these methods pioneered the processing of 3D data using 2D architectures via projection. For outdoor LiDAR scenes, projection-based methods have also been developed to take advantage of the well-established 2D architectures. Two common types of projections are the Birds-Eye View (BEV) and spherical projections.

#### Birds-Eye View (BEV) Projection

BEV-based methods project the 3D LiDAR point cloud onto the ground plane, eliminating the height (often $z$) dimension. This representation exploits the fact that in the outdoor driving setting, objects predominantly lie on the ground with little variation in their vertical locations. To form a $H \times W$ BEV image, the point cloud $[x\ y\ z] \in \mathbb{R}^3$ can be discretized into a 2D grid as follows:

$$r = \left\lfloor \frac{y - y_{\min}}{y_{\max} - y_{\min}} \cdot H \right\rfloor, \qquad c = \left\lfloor \frac{x - x_{\min}}{x_{\max} - x_{\min}} \cdot W \right\rfloor, \tag{2.4}$$

where each cell at $(r, c)$ represents the occupancy of that region. As shown in Figure 2.5 (b), the resulting BEV image is often sparse, especially for faraway and occluded regions, with an unbalanced density distribution. To address this, a polar coordinate alternative is proposed by Zhang et al. [307], which has been adopted by recent work on point cloud segmentation [190, 318].

To preserve the 3D information from the original point cloud, BEV image typically include additional feature channels. These features can be hand-crafted, such as height and intensity [283]. Alternatively, learned feature vectors can be used to encode the points within each cell to form a 2D feature map (also called a pseudo-image) [67, 119, 294]. Such methods are sometimes referred to as pillar-based, since each pixel in the BEV feature map corresponds to a

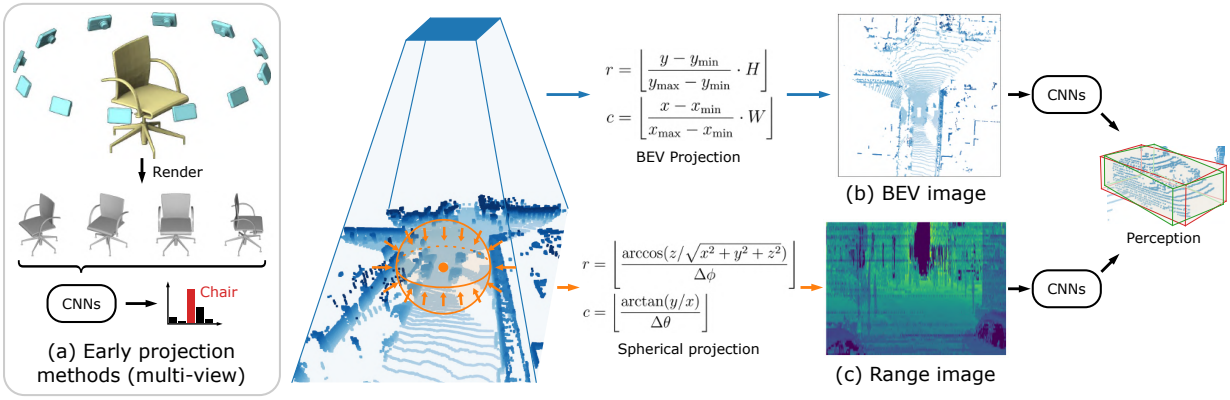



Figure 2.5: Projection-based methods compared to early multi-view projection methods for 3D shape classification. Renders in (a) are adapted from [217].

vertical pillar in the 3D space. In these methods, a small trainable point-based network is often used as the feature extractor, which will be discussed further in Section 2.3.3.

BEV projection offers two main advantages [283]. First, it preserves the metric space, meaning that an object's size relative to the environment remains consistent after the projection. Therefore, a perception model does not need to learn size invariance, allowing it to better exploit object size priors. Second, the BEV image provides an unobstructed view to each object since objects generally do not overlap each other in the BEV view. These properties are in contrast to image-based perception where faraway objects will appear smaller due to the projective transformation, and occlusion between objects is common.

Due to these benefits, the BEV image is widely used in early 3D object detection [6, 10, 14, 170, 213, 282, 283, 308]. Many later methods incorporate camera images for more accurate localization [29, 112, 134]. While recent 3D object detectors have moved away from hand-crafted BEV features, they still collapse the height dimension in the feature space and perform detection on the resulting BEV feature map [33, 280, 317].

### Spherical Projection (Range Image)

Spherical projection is another commonly used technique for outdoor LiDAR point clouds. As the name suggests, this approach quantizes the point clouds' spherical coordinates into a 2D *range image*, where the column represents the azimuth and the row represents the elevation (or beam ID), as illustrated in Figure 2.5 (c). Specifically, for a LiDAR point cloud with horizontal and vertical angular resolution $\Delta\theta$ and $\Delta\phi$, a point $[x\ y\ z]^\top \in \mathbb{R}^3$ is projected onto the range

image at row $r$ and column $c$ by:

$$r = \left\lfloor \frac{\arccos(z/\sqrt{x^2+y^2+z^2})}{\Delta\phi} \right\rfloor, \qquad c = \left\lfloor \frac{\arctan(y/x)}{\Delta\theta} \right\rfloor. \tag{2.5}$$

Like BEV images, range images can encode multiple channels, including coordinates, range, intensity, and potentially other point cloud features such as elongation.

The range image resembles the depth channel of a RGB-D camera image, but is not produced by perspective projection. Consequently, there are often visual distortions as straight lines are not preserved after the projection. Moreover, similar to image-based perception, occlusion and size variation can occur in the range image, making it difficult to utilize size priors [17, 167]. Finally, directly applying 2D CNNs on the range image cannot capture the spatial relationship between the points, as adjacent pixels in the range image can be far apart in the 3D space [9, 58].

Nevertheless, range image offers a few advantages over BEV projection [262]. First, as the range image corresponds to how a top-mounted spinning LiDAR scans the environment, it is more compact and memory-efficient with few empty pixels. The Waymo dataset, for instance, stores its LiDAR point cloud data in the range image format [218]. Additionally, since the range value is encoded as a feature channel rather than a pixel location as in BEV, the size of the range image is independent of the point cloud's maximum range, making range-image-based methods scalable to the emerging long range LiDAR systems.

Range image remains popular in LiDAR-based perception methods thanks to its compactness and image-like appearance. Existing image-based meta-architectures can often be extended to range image with many works on 3D object detection [9, 17, 24, 58, 125, 135, 136, 167, 168, 219, 236, 262] and point cloud segmentation [2, 4, 35, 45, 108, 131, 169, 193, 263, 264, 273, 313]. In addition to CNNs, vision transformers have also been applied to range image [5, 73, 109].

### 2.3.2 Voxel-base

Voxel-based methods transform the unstructured and unordered point cloud into a 3D voxel grid via voxelization (Figure 2.6 (a)). This can be achieved by quantizing the location of the points in the point cloud. Specifically, given a point with coordinate $\mathbf{x} \in \mathbb{R}^3$ and a desired voxel size $\mathbf{d} \in \mathbb{R}^3$, the corresponding voxel grid coordinate $\mathbf{v}$ is given by:

$$\mathbf{v} = \left\lfloor \frac{\mathbf{x} - \mathbf{x}_{\min}}{\mathbf{d}} \right\rfloor, \tag{2.6}$$

where $\mathbf{x}_{\min}$ is the minimum coordinate of the point cloud. The voxel grid is a natural 3D representation that preserves spatial information. Similar to projection-based approach, a voxelized

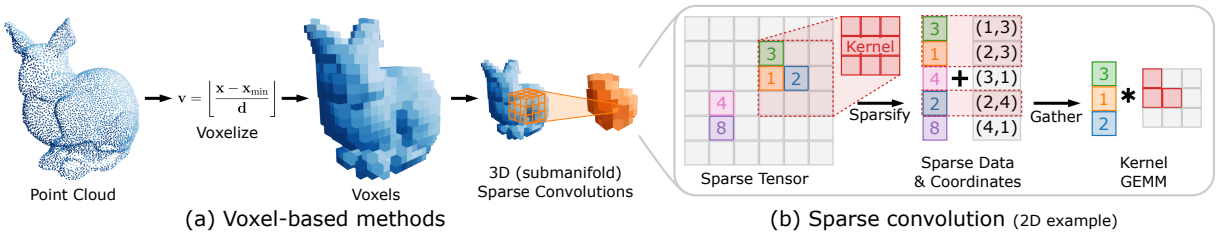



Figure 2.6: Voxel-based methods using 3D (submanifold) 3D convolutions.




Figure 2.7: Standard convolution activates previously empty pixels (submanifold dilation), while submanifold convolution preserves the sparsity of the data. Visualizations are adapted from [71]

point cloud can be processed using meta-architectures from image processing, such as Feature Pyramid Network (FPN) [137] and U-Net [199], but with 3D convolutions.

While typically considered as an image with an extra dimension, the voxel grid introduces two challenges: its size and the computation complexity of 3D convolution. First, voxel grids for outdoor scenes can be much larger than images. Consider the voxelization of a $200\,\mathrm{m} \times 200\,\mathrm{m} \times 10\,\mathrm{m}$ region with a voxel size of $0.05\,\mathrm{m}$, the resulting volume will contain 3.2 billion voxels, comparable to over ten thousands $640 \times 480$ images. Fortunately, each LiDAR scan only contains 100–200 K points [218], resulting in a highly sparse voxel grid. Therefore, outdoor scenes are usually represented as sparse tensors with only the occupied voxels' grid coordinates and feature vectors [280]. The sparse convolution involves gathering the relevant features within the receptive field and applying the appropriate kernel values [71], as illustrated in Figure 2.6 (b). This improves both the memory and the computational efficiency as the empty regions are ignored.

Second, the standard convolution can drastically reduce the sparsity of the data, which worsens the computational complexity in deep models. As shown in Figure 2.7 (a), convolution may activate previously empty neighboring locations, a phenomenon known as submanifold dilation [71]. To counter this, Graham and van der Maaten [71] introduce a *submanifold* convolution for sparse data. Instead of updating all neighboring locations, submanifold convolution leaves previously empty locations unchanged (Figure 2.7 (b)), sacrificing a portion of the receptive field to preserve the sparsity of the data. Subsequent works further optimize the performance of the 3D sparse convolutions on the algorithmic level [31, 122, 141, 175], software level [38, 84, 223, 224, 280, 285], and hardware level [139, 140, 157, 158, 173, 259, 290].

These advancements in sparse data processing and 3D convolution are pivotal for developing voxel-based deep neural networks. While early shape recognition methods [19, 165, 250, 268] can rely on dense representation due to the small data size and simple network designs, sparse representation and algorithms are essential for complex outdoor applications such as 3D object detection [48, 78, 87, 113, 207, 208, 209, 280, 298, 317] and point cloud segmentation [36, 38, 72, 86, 145, 222, 274, 278, 293, 321]. Attention-based voxel processing is another emerging area that also benefits from these advances, with numerous transformer-based backbones proposed to replace 3D CNNs [53, 59, 60, 62, 63, 79, 80, 128, 147, 164, 252, 315, 319].

### 2.3.3 Point-based

Point-based methods directly process the point cloud without any transformation to a structured representation. This approach preserves fine-grained geometric details and avoids quantization errors inherent in projection and voxelization. The are three main approaches for learning directly from point clouds: PointNet (MLP-based), graph neural networks (GNNs), and attention-based point transformers.

#### PointNet / MLP-based

Qi et al. [185] pioneered the direct processing of point clouds with the introduction of PointNet. A key challenge in point cloud learning is that the method needs to be permutation invariant while allowing interactions between points. PointNet achieves this by first processing each point independently using a shared multi-layer perceptron (MLP) and then aggregating the resulting features using a symmetric function, such as max pooling or average pooling. More precisely, given a point cloud with N points $\mathcal{P} = \{\mathbf{x}_i \in \mathbb{R}^3 \mid i = 1, \ldots, N\}$, the network $f(\mathcal{P})$ is defined as follows:

$$f(\mathcal{P}) = g\left(\{h(\mathbf{x}_i)\}_{i=1}^{N}\right) \tag{2.7}$$

where $h$ is a MLP and $g$ is the symmetric function (Figure 2.8 (a)).

However, since the aggregation in PointNet is performed over the entire point cloud, it primarily learns global features. This can limit its performance in large complex scenes where local structures are important. To address this, the follow-up work PointNet++ [188] applies PointNet recursively on local neighborhoods and extracts point features hierarchically, as shown in Figure 2.8 (d), allowing it to capture both local and global structures. Recently, Qian et al. [189] systematically improve PointNet++ with better model scaling and training strategies, achieving

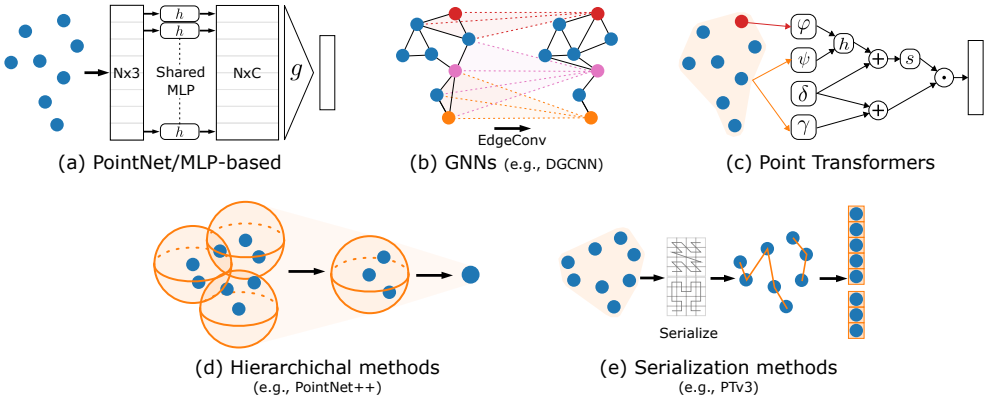



Figure 2.8: Point-based point cloud processing methods.

SOTA performance. Building on this paradigm, subsequent works explore different convolution-like operations on point clouds [232, 233, 265, 276], and PointMLP [160] further introduces residual connections to achieve better performance.

While a few methods directly apply PointNet to the more complex tasks, such as the Frustum-PointNet [186] for 3D object detection, it is more commonly used as an encoder for pillar-based BEV methods [67, 119, 294] and voxel-based methods [252, 280, 317] where the PointNet is applied to points within each voxel partition, typically referred to as the voxel feature extractor (VFE).

### Graph Neural Networks (GNNs)

GNN is another permutation-invariant approach suited for processing unstructured point cloud data. In this paradigm, a point cloud is considered as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertices $\mathcal{V}$ represent points, and edges $\mathcal{E}$ are typically established based on spatial proximity, such as k-nearest neighbors (kNN). In each GNN layer, the feature vector $x_i$ of a vertex $v_i \in \mathcal{V}$ is updated by aggregating information from neighboring vertices:

$$x_i^{'} = g\left(\left\{h\left(x_i, x_j\right)\right\}_{(i,j)\in\mathcal{E}}\right), \tag{2.8}$$

where $h$ is a pair-wise function modeling the interaction between vertices and $g$ is a symmetric aggregator. $h$ and $g$ can take on many forms as exemplified in [258]. Both conventional CNNs and the PointNet as described in Equation (2.7) can be considered special cases of the GNN.

Dynamic Graph CNN (DGCNN) [258] is among the first method to directly process point clouds using GNNs. It introduces EdgeConv to more effectively capture local structures by dy-

namically recomputing the edges after each layer (Figure 2.8 (b)). DeepGCN [126] introduces residual connections and normalization techniques to support deeper GNN networks. More recently, the quasi-dynamic GNN (qDGNN) [174] is proposed to further reduce the computational overhead resulting from the dynamic graph updates by limiting the graph re-computation after a certain depth.

Several works have applied GNNs to complex tasks in outdoor scenes including 3D object detection [81, 210, 272] and point cloud segmentation [118, 231]. However, the adoption of GNN-based backbones in these applications is somewhat limited, potentially due to the relatively high computational cost as well as the complexities in the implementation.

### Point Transformers

The recent popularization of attention [245] has sparked a lot of interests in transformer-based point cloud learning. These approaches utilize the self-attention to model interactions among points while preserving permutation invariance. In the general form, given a set of feature vectors $\mathcal{X} = \{x_i\}$, a self-attention layer updates each feature as follows:

$$x_i^{'} = \sum_{x_j \in \mathcal{X}} s\left(h(x_i, x_j)\right) \cdot \gamma(x_j) \tag{2.9}$$

where $h$ is a relation function, $s$ is a normalization function (e.g., softmax) that converts the output of $h$ into attention weights, and $\gamma$ is a learnable mapping such as a linear layer. In practice, an additional positional encoding $\delta$ is often added to the features. Attention can take many forms [310]. For example, in scaled dot-product attention [245], $h(x_i, x_j) = \varphi(x_i)^\top \psi(x_j)$ where $\varphi$ and $\psi$ are both linear layers, and $s(z) = \mathrm{Softmax}(\frac{z}{\sqrt{d}})$ where $d$ is the feature dimension. In natural language processing (NLP), the projections $\varphi(x_i), \psi(x_j), \gamma(x_j)$ are conventionally referred to as the query $Q$, key $K$, and value $V$, giving the familiar matrix form $\mathrm{Attention}(Q, K, V) = \mathrm{Softmax}\left(\frac{QK^\top}{\sqrt{d_K}}\right) V$.

Point Cloud Transformer (PCT) [74] pioneered the use of scaled dot-product attention and offset attention in point cloud processing. However, similar to PointNet [185], PCT computes attention globally across all points, limiting its scalability to large point clouds. Point Transformer (PT) [311] instead adopts a vector/batch-wise self-attention and restricts the attention computation to local neighborhoods, typically defined via kNN. Furthermore, PT uses a symmetric encoder-decoder design (U-Net) where downscaling is performed by farthest point sampling (FPS) and upscaling by interpolation, which allows it to efficiently process large indoor scenes. The follow-up version PTv2 [266] further improves the computational efficiency with

the grouped vector attention, where attention weights are shared within groups, and partition-based subsampling using voxelization instead of the more expensive FPS.

Neighborhood computation via kNN remains a computational bottleneck in these local attention frameworks [267]. Recent approaches have explored alternative methods based on point cloud serialization including octree [253] and space filling curves [28], as illustrated in Figure 2.8 (c). PTv3 [267] capitalizes on this idea by alternating between different serialization strategies in each layer. This greatly reduces the computational overhead compared to its predecessor PTv2, while achieving SOTA performance across a wide range of tasks. It is worth-noting that these serialization methods inherently violate permutation invariance. However, as demonstrated by PTv3, such requirement might not be critical for complex tasks.

Window-based attention is another alternative that improves the computational efficiency for large point clouds. Following SwinTransformer [144], Lai et al. [116] use the cubic window partition (voxelization) to determine the local attention neighborhood. To facilitate long-range context learning, the partition is performed at different scales with different sampling rate. The subsequent work SphereFormer [117] introduces the radial window partition to better exploit the distribution of the LiDAR point clouds. When used in conjunction with the cubic partition, SphereFormer is able to achieve SOTA performance.

In addition to promising methods like PTv3 and SphereFormer, the popularity of transformer-based backbones stems from several reasons. Algorithmically, transformers can capture complex and long range dependencies without resorting to large graphs or CNN kernels [32]. This naturally gives it a large receptive field [117], making it favorable for complex tasks. From an engineering standpoint, transformers are straightforward to implement since they primarily consist of matrix operations and simple MLPs. It is also well supported by popular deep learning frameworks, often incorporating numerous optimization techniques that improve both computational and memory efficiency [46, 47, 124, 203].

## 2.4 LiDAR-based 3D Object Detection

3D object detection involves identifying and localizing objects within a scene by classifying and predicting a bounding box around the object. For outdoor scenes, a 3D bounding box typically has 7 degrees of freedom parameterized by $(x, y, z, l, w, h, \theta)$, where $(x, y, z)$ represent the 3D location, $(l, w, h)$ are the dimensions of the bounding box, and $\theta$ is the orientation or yaw angle.

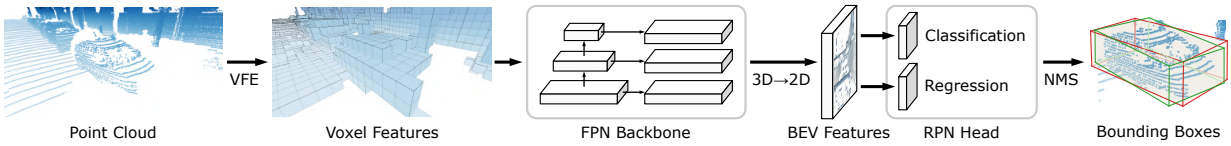



Figure 2.9: VoxelNet/SECOND object detection pipeline. Subsequent works largely follow this modular design with different improvements.

### 2.4.1 Architectures

As mentioned in the previous sections, numerous approaches have been proposed for LiDAR-based 3D object detection. Depending on the point cloud learning method adopted, existing methods can be categorized into projection-based, point-based, and voxel-based. Some approaches [207, 209] use multiple representations, often referred to as hybrid methods. Similar to image-based object detection, existing methods can be single-stage or two-stage, anchor-based or anchor-free. In this section, we will introduce a few relevant and representative voxel-based approaches that form the basis for the methods proposed in this thesis.

**VoxelNet [317] / SECOND [280]** VoxelNet is a seminal work in end-to-end 3D object detection. It establishes a number of architectural designs that serve as blueprints for subsequent works. As illustrated in Figure 2.9, VoxelNet consists of three main components: (i) voxel feature extractor (VFE), (ii) 3D CNN backbone (also referred to as the middle layers), and (iii) region proposal network (RPN, or "head").

- **VFE** After the input point cloud is voxelized into a 3D voxel grid, a PointNet/MLP-based VFE is applied to aggregate local point features within each voxel. The VFE outputs a 4D tensor of dimensions $(\mathrm{C} \times \mathrm{D} \times \mathrm{H} \times \mathrm{W})$, where $\mathrm{C}$ is the number of channels, and $\mathrm{D}, \mathrm{H}, \mathrm{W}$ are the voxel grid dimensions.
- **Backbone** The 3D CNN backbone follows a similar design to the FPN [137]. It progressively aggregates voxel features while reducing the spatial resolution. In VoxelNet, the backbone is implemented using conventional dense 3D convolutions. SECOND improves upon this design by replacing the dense convolutions with a combination of submanifold sparse convolutions and conventional sparse convolutions.
- **RPN Head** In the final stage, the 4D backbone feature tensor is flattened to form a $(\mathrm{C}'\mathrm{D}' \times \mathrm{H}' \times \mathrm{W}')$ dense feature map by concatenating the feature vectors along the height dimension. This resulting feature map is effectively a low resolution BEV pseudo-image

as previously discussed in Section 2.3.1. A 2D CNN-based RPN head is then applied to generate object bounding box proposals. VoxelNet and SECOND both adopt anchor-based RPN. For each pixel in the BEV feature map, a set of pre-defined anchors with different dimensions and orientations are assigned. The RPN then performs classification and regresses offsets to align these anchors with ground truth bounding boxes. Specifically, given an anchor box $(x_a, y_a, z_a, l_a, w_a, h_a, \theta_a)$ and the corresponding ground truth box $(x_g, y_g, z_g, l_g, w_g, h_g, \theta_g)$, the RPN predicts the following:

$$\begin{aligned} &\Delta x = \frac{x_g - x_a}{d_a}, \quad \Delta y = \frac{y_g - y_a}{d_a}, \quad \Delta z = \frac{z_g - z_a}{h_a}, \\ &\Delta l = \log\left(\frac{l_g}{l_a}\right), \quad \Delta w = \log\left(\frac{w_g}{w_a}\right), \quad \Delta h = \log\left(\frac{h_g}{h_a}\right), \\ &\Delta\theta = \theta_g - \theta_a, \qquad \text{where } d_a = \sqrt{l_a^2 + w_a^2}. \end{aligned} \tag{2.10}$$

The predicted boxes are then pruned by confidence score threshold and non-maximum suppression (NMS).

**PointPillars [119]** PointPillars is a pillar-based architecture that performs detection in BEV. It follows the same VFE-Backbone-RPN design as VoxelNet and SECOND, with a two key differences. First, the point cloud is voxelized into pillars instead of a 3D voxel grid, forming a $(C \times H \times W)$ BEV pseudo-image after VFE. Second, the backbone and RPN are both based on 2D CNNs. Consequently, PointPillars is much more efficient than previous works, allowing it to easily achieve real-time performance (over 20 Hz).

**CenterPoint [298]** Inspired by the development of anchor-free 2D object detectors such as CornerNet [121] and CenterNet [55], CenterPoint adapts the anchor-free paradigm to 3D object detection. In contrast to previous works that rely on pre-defined anchor boxes and offset regression, CenterPoint formulates the localization task as keypoint detection on the BEV feature map. Specifically, for object centers, a 2D CNN-based keypoint detector is used to predict a heatmap representing the likelihood of object centers. For multi-class detection, separate heatmaps are generated per object category. Additional box attributes including the box dimensions and orientation are directly regressed to match the corresponding ground truth, similar to the regression of anchor offsets in anchor-based methods. This design simplifies the detection head and eliminates the complexities associated with anchor matching. It can be easily applied to both VoxelNet and PointPillars backbones, and has since been widely adopted by subsequent works.

**VoxelNeXt [33]** Motivated by the fact that the LiDAR BEV feature maps are typically sparse, VoxelNeXt proposes a fully sparse network design, similar to FSD [60, 62, 63]. It uses sparse convolutions in both the backbone and the detection head, and incorporates several optimizations to further reduce computation. Similar to CenterPoint, box attributes are directly regressed. However, since the object center may not correspond to a valid pixel in a sparse feature map, VoxelNeXt instead predicts bounding boxes directly from voxels. In practice, the voxel nearest to the bounding box center is responsible for regressing the objectness and bounding box attributes. During inference, a sparse max-pooling operation is applied to efficiently select these object voxels. VoxelNeXt represents the current SOTA voxel-based detector with both the voxel and pillar variant. This fully sparse design allows it to scale better to large scenes than other dense BEV methods.

**DSVT [252]** DSVT is a SOTA voxel-based 3D object detector with a transformer backbone. It follows the SwinTransformer [144] design where the self-attention is applied locally to sliding windows. To improve the parallelization of self attention, in DSVT, the voxels within each window are further sorted along the $x$ or $y$ coordinate and then divided into subsets of equal size, with the number of subsets dynamically determined by the feature sparsity. Self-attention is then applied to each subset in parallel. To allow intra-window interaction, each DSVT block alternates between partitioning along the $x$-axis and the $y$-axis. DSVT can be implemented in both voxel and pillar variants, with the DSVT-pillar able to achieve better performance than CNN-based backbones while having comparable inference times.

Thanks to the modular nature of the existing architectures, several open-source frameworks have been developed to accelerate research in this domain. Notably, OpenPCDet [228] and MMDetection3D [40] streamline the process of configuring, training, and evaluating 3D detectors. They both support popular dataset benchmarks mentioned in Section 2.2, and implement a variety of VFE, backbone, and RPN architectures, including the ones mentioned above.

### 2.4.2 Model Training

Existing 3D object detection methods largely employ established machine learning techniques, including learning rate scheduling (often the one cycle [215] policy), and optimizers such as Adam [107] and AdamW [150]. In addition to these standard techniques, several widely adopted data augmentation methods and training losses are specific to LiDAR-based 3D object detection and are essential for achieving SOTA performance.

**Data Augmentation** Similar to image-based perception, a variety of transformations are applied to the input point clouds, including random translation, rotation, scaling, and flipping. The

scaling factor is often within the range of $[0.95, 1.05]$ [317] to preserve object size priors. To increase the number of objects per scene and improve model convergence, ground truth sampling (GT sampling) [280] is often used. In GT sampling, several ground truth objects are randomly selected and inserted into the each training scene. Class-balanced grouping and sampling (CGBS) [320] further improves GT sampling for datasets with imbalanced class distribution and is widely used for training on the nuScenes dataset [20].

**Training Losses** For classification, due to the inherent imbalance between object and background in LiDAR data, focal loss [138] is typically used instead of the standard cross-entropy. Specifically, given the prediction $\hat{y}$ and ground truth $y$, the focal loss is given by

$$\mathcal{L}_{\text{FL}} = -y \cdot \alpha \, (1 - \hat{y})^{\gamma} \log(\hat{y}) - (1 - y) \cdot \alpha \, \hat{y}^{\gamma} \log(1 - \hat{y}), \tag{2.11}$$

where $\gamma$ is a parameter that down-weight easy examples while $\alpha$ adjusts the importance of positive and negative classes. For regression, the L1-loss is commonly used. Recently, IoU-based losses have also seen adoption in 3D object detection. Existing formulations for axis-aligned boxes [194, 314] can be directly applied to 3D by ignoring the height and orientation. For instance, the DIoU loss used in VoxelNeXt [33] takes the following form:

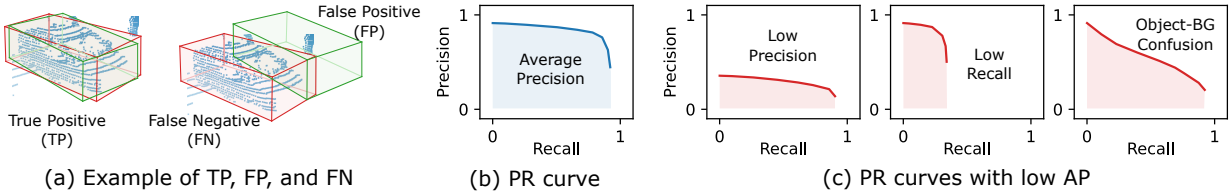
$$\mathcal{L}_{\text{DIoU}} = 1 - \text{IoU}(b, b_{\text{gt}}) + \frac{\rho^2(b, b_{\text{gt}})}{c^2}, \tag{2.12}$$

where IoU is the intersection-over-union, $\rho^2$ is the Euclidean distance between box centers, and $c$ is the diagonal length of the smallest enclosing box. Several IoU-based losses have also been developed for 3D object detection to address the complexities inherent to the IoU calculation of 3D rotated bounding boxes [192, 204].

### 2.4.3 Evaluation Metrics

To evaluate the performance of a 3D object detector, the predictions are first matched with the ground truth (GT) objects. Each prediction is classified as either a *True Positive* (TP) if it matches a ground truth object or a *False Positive* (FP) if it does not. Ground truth objects that do not match any predictions are *False Negatives* (FNs). Examples of TP, FP, and FN are shown in Figure 2.10 (a). After object matching, several evaluation metrics can then be computed based on the TP / FP / FN statistics.

**Matching Criteria** To determine if a prediction matches a ground truth object, intersection-over-union (IoU) is often used as the matching criteria. A prediction is considered a TP if it




(a) Example of TP, FP, and FN (b) PR curve (c) PR curves with low AP

Figure 2.10: 3D object detection evaluation.

overlaps with a GT object with an IoU above a predetermined threshold, typically 0.7 for vehicles and 0.5 for smaller objects such as pedestrians and cyclists [69, 218]. Alternatively, the center distance can be used [20], where a TP prediction only needs to be close in center distance to a GT object, without considering size and orientation. More recently, difference in distance-to-collision ($\Delta$DTC) [304] is proposed as another matching criteria, where the $\Delta$DTC between a prediction and a GT object needs to be small in order to be considered a TP. This criteria can reflect the detector's ability to accurately determine DTC, which is relevant in the autonomous driving setting.

**Average Precision** The primary metric for evaluating an object detector is the *Average Precision* (AP), which is defined as the area under the precision-recall (PR) curve. Given the number of TP, FP, and FN objects, the precision and recall are defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2.13}$$

A PR curve is generated by plotting the precision and recall under different confidence thresholds (Figure 2.10 (b)), which reflects the inherent trade-off between precision and recall. A detector with high AP can achieve high precision and recall simultaneously, while a low AP performance indicates issues including low recall, low precision, and object-background confusion in the 3D object detection model, as illustrated in Figure 2.10 (c). For multi-class object detection, the *mean average precision* (mAP) metric can be computed by averaging the per-class AP scores.

**NuScenes Detection Score** In addition to mAP, the nuScenes benchmark uses a more comprehensive metric that reflects other aspects of the object detection [20]. Specifically, for TP detections, nuscenes measures the average translation error (ATE), average scale error (ASE), average orientation error (AOE), average velocity error (AVE), and average attribute error (AAE). These errors are then consolidated into a single metric called the *nuScenes Detection Score* (NDS):

$$\text{NDS} = \frac{1}{10}[5 \cdot \text{mAP} + \sum_{\text{mTP}}(1 - \min(1, \text{mTP}))] \tag{2.14}$$

where mTP are the five mean errors mentioned above.

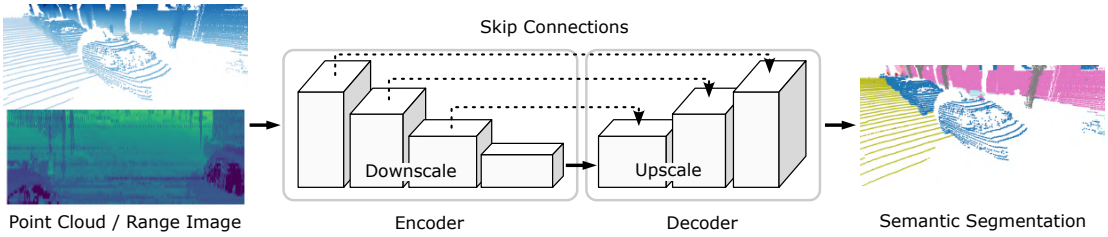



Figure 2.11: The U-Net architecture commonly used in 3D semantic segmentation methods. Depending on the input representation, the encoder and decoder blocks can be CNN-based or point-based.

**Waymo APH** Similar to the AOE from the nuScenes benchmark, Waymo introduces the APH metric [218], an extension to AP that incorporates heading errors. Specifically, when calculating recall, each true positive is weighted by the heading accuracy: $\min(|\hat{\theta}-\theta|, 2\pi-|\hat{\theta}-\theta|)/\pi$. APH is then similarly defined as the area under the weighted PR curve.

## 2.5 LiDAR-based 3D Semantic Segmentation

LiDAR-based 3D semantic segmentation involves classifying each point in the LiDAR point cloud. Unlike object detection, which often requires inferring the size of occluded objects, semantic segmentation focuses solely on the existing points within the point cloud.

### 2.5.1 Architectures

As detailed previously in Section 2.3, numerous methods have been proposed for 3D semantic segmentation using different point cloud learning paradigms. To efficiently process outdoor scenes, many approaches adopt the well-established U-Net [199] architecture. As depicted in Figure 2.11, a U-Net consists of three main components: (i) an encoder that progressively extracting features while increasing the receptive field and reducing the dimensionality of the input data, (ii) a decoder that upscales the encoded feature map back to the original resolution to produce per-point predictions, and (iii) skip connections that connect the corresponding layers in the encoder and decoder at each stage to help preserve fine-grained information.

The U-Net architecture can be easily adapted to range-image-based methods [2, 45, 263, 264, 273] thanks to the similarity between the range image and camera image. For 3D representations, 3D U-Net [39] pioneered the use of U-Net for volumetric segmentation using 3D

convolutions. Graham et al. [72] adapt this design to voxelized point clouds using submanifold sparse convolutions. Similarly, the MinkowskiUNet [38] employs sparse convolutions with non-conventional kernel shapes for the segmentation of point cloud sequences. PolarSeg [307] and Cylinder3D [321] further improve the performance of sparse U-Nets by using polar coordinate partitions and custom convolution operations.

For point-based methods, the traditional CNN backbone can be replaced with various point-based techniques, as detailed in Section 2.3.3. However, downscaling and upscaling operations can be nontrivial due to the unstructured nature of 3D point clouds. In Point Transformer [311], farthest point sampling (FPS) is applied to each stage of the encoder, and kNN is used to establish the local neighborhood for feature pooling. Decoding is done symmetrically by processing the point features with a linear layer and then mapping the result back to the higher resolution point cloud using the previously computed kNN. The subsequent works [116, 117, 266, 267] avoid FPS and kNN-based pooling entirely by using a simpler partition-based neighborhood, often achieved via voxelization and sparse convolutions.

### 2.5.2 Model Training

Due to the complexity of the outdoor scenes, data augmentation and appropriate loss functions are both crucial for training accurate 3D semantic segmentation models, with many techniques proposed and adopted to achieve SOTA performance.

**Data Augmentation** Similar to 3D object detection, standard transformations are applied randomly to the input point cloud during training, including translation, rotation, scaling, flipping, and jittering. Additionally, inspired by mixed sample training approaches in image-based vision tasks [248, 301, 302], several point cloud mixing techniques have been proposed [34, 172, 269, 303] and adopted by SOTA methods [267]. For instance, Mix3D [172] concatenates a pair of randomly transformed point clouds to form a new training sample, exposing objects to a variety of contexts that are not present in the original data. This approach has a similar effect to GT sampling augmentation in 3D object detection [280]. A more recent evolution of this idea, PolarMix [269], performs both scene-level mixing and object-level sampling while respecting the polar nature of LiDAR point clouds.

**Training Losses** Most existing 3D semantic segmentation methods use the standard cross-entropy loss as the training objective. To combat the inherent class imbalance in LiDAR data, the loss is often weighted based on class frequencies [273], and the focal loss is used in some methods [264]. More recently, Berman et al. [15] proposed the Lovász-Softmax loss as a tractable

surrogate for directly optimizing the mean intersection-over-union (mIoU) objective. This loss has been widely adopted by recent 3D semantic segmentation methods, often used in conjunction with the standard cross-entropy loss.

### 2.5.3 Evaluation Metrics

The standard evaluation metric in semantic segmentation is the *mean intersection-over-union* (mIoU), which measures the overlap between predictions and GT across all classes. It is calculated as:

$$\text{mIoU} = \frac{1}{C} \sum_{c=1}^{C} \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c + \text{FN}_c} \tag{2.15}$$

where $C$ is the number of classes, and $\text{TP}_c, \text{FP}_c, \text{FN}_c$ are the true positive, false positive, and false negative for a class $c$, respectively.

A frequency-weighted variant, known as fwIoU, is introduced in nuScenes [20], where each class's IOU is weighted by the point frequency of that class in the dataset. However, fwIoU has not seen much usage in exsting works, as it can be biased by majority classes such as road and terrain, and does not adequately reflect the performance on small objects.

## 2.6 Multi-frame/sequential LiDAR perception

LiDAR point clouds are typically collected continuously in a sequence, and this temporal nature can be used to improve LiDAR perception performance. Many methods have extended existing single-frame architectures to incorporate temporal information.

**Input Concatenation** For both point- and voxel-based methods, one common approach is to concatenate multiple input point clouds after ego-motion correction [20]. Specifically, let $P_\tau \in \mathbb{R}^{N_\tau \times 3}$ denote a point cloud with $N_\tau$ points collected at timestamp $\tau$. We also assume the ego pose $T_\tau \in \mathbb{SE}(3)$ is available, representing a transformation from the ego coordinate system to a common global coordinate system. In practice, the ego pose is typically provided by onboard sensors like GPS and IMU or estimated using SLAM algorithms. The aggregated $n$-frame point cloud $P_\tau^n$ is then given by:

$$P_\tau^n = \bigoplus_{i=0}^{n-1} P_{\tau-i}(T_\tau^{-1} T_{\tau-i})^\top \tag{2.16}$$

where $\oplus$ denotes the concatenation operation. The term $T_{\tau}^{-1}T_{\tau-i}$ represents a transformation that aligns the point cloud $P_{\tau-i}$ with $P_{\tau}$ through the global coordinate system.

Point cloud concatenation does not require any modification to the perception architecture, making it applicable to a wide range of tasks. It is widely adopted in 3D object detection, becoming the default approach for datasets with sparse point clouds like nuScenes [20], but less common in 3D semantic segmentation, as the U-Net architecture does not scale well with increasing input size. Nevertheless, MSeg3D [127] demonstrated that concatenated point clouds can be beneficial to semantic segmentation performance.

This strategy can also be applied to the range image representation commonly used in 3D semantic segmentation. For example, SpSequenceNet [205] proposes a cross-frame global attention and local interpolation to fuse CNN features from the previous frame. Meta-RangeSeg [254] computes a residual range image representation to captures the temporal difference between frames. However, in the multi-frame setting, range images can suffer from misalignment due to ego motion.

**Feature Aggregation** Alternatively, the temporal information can be incorporated at the feature level, which we refer to as feature-based aggregation in this thesis. These methods typically requires modifications to the perception architecture and introduces varying amount of computational overhead. In 3D object detection, early work by Luo et al. [153] concatenates feature maps from multiple point clouds, presenting a trade-off between accuracy and efficiency depending on the feature layer used. However, naive concatenation of feature maps inevitably introduces misalignment at the feature level due to ego motion. Huang et al. [89] instead use an LSTM to encode temporal information as hidden features, and address the alignment issue by transforming the feature map using ego motion. Attention mechanisms have recently gained popularity in feature-level aggregation and have shown promising results. Yin et al. [297] propose a GRU module equipped with spatiotemporal attention for better feature alignment. 3D-MAN [292] and MPPNet [30] both employ attention mechanisms to combine features generated from a single or few-frame region proposal network to produce more refined detections. To better utilize the rich multi-scale features, TransPillars [156] proposes attention-based feature-level aggregation at the voxel level to preserve the instance and contextual information. In 3D semantic segmentation, TemporalLatticeNet [202] extends LatticeNet [154] and fuses temporal features using recurrent neural networks and a novel abstract flow module.

**Pseudo-labelling** Another line of work explores the use of long sequences to generate pseudo-labels for 2D object detection. Unlike conventional object detection, pseudo-labelling methods (also called offline or offboard methods) are not constrained by real-time processing requirements, and thus can afford more complex, multi-stage pipelines. These methods typically em-

ploy a single- or few-frame detector to generate initial predictions, followed by offline tracking and a second stage refinement that utilizes full-sequence point clouds aggregated at object- or track-level [61, 159, 187, 281, 284].

## 2.7 Domain Adaptation in LiDAR perception

It is well-known that the performance of machine learning models can degrade when there is a domain shift between training and testing data. In LiDAR-based perception, the domain gap can come from various factors, including: (i) different noise distributions, such as missing returns and extraneous points often caused by adverse weather conditions, (ii) changes in LiDAR scan patterns due to sensor properties (e.g., beam counts) or external configurations (e.g., mounting positions), and (iii) a shift in object distributions, which may result from a new geolocation. Domain adaptation aims to adapt a model trained on a source domain to perform well on a new target domain.

Several methods have been proposed to address the domain adaptation problem for 3D object detection. One line of work involves improving model robustness via regularization [255, 277]. Other works attempt to close the domain gap via domain mapping [11, 42] or input [238, 239, 256], feature [155, 296, 305], and output [50] alignment. To take advantage of available target domain data, SOTA methods often use pseudo-labelling. Pseudo-labels can be improved via tracking-based refinement [66, 299] or iterative self-training [286, 288]. When a small amount of target labels are available, CutMix and MixUp have also been shown to be effective techniques to incorporate labelled target data [257].

# Chapter 3

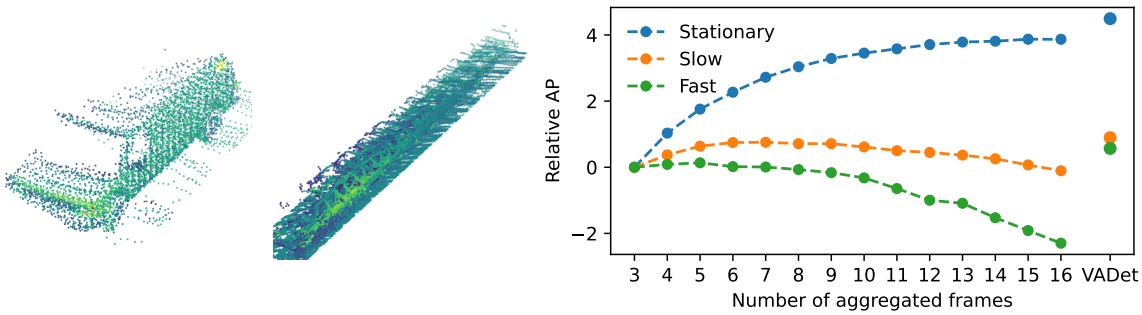
# VADet

## *Multi-frame LiDAR 3D Object Detection using Variable Aggregation*

## 3.1 Introduction

LiDAR-based methods give state-of-the-art (SOTA) 3D object detection performance in autonomous driving. While object detectors can produce accurate detections from a single LiDAR point cloud [33, 252], they have been shown to benefit from aggregated input consisting of multiple consecutive frames. A widely adopted method for incorporating multi-frame sequential point clouds is what we term fixed aggregation, where a fixed number of frames are concatenated after ego motion correction, often including timestamps as an additional feature. This is a simple and effective method to augment the input with more spatial and temporal information, without modifying the architecture [20].

However, it has been observed that the effectiveness of fixed aggregation diminishes as more frames are used, eventually degrading the detection performance [30, 292]. Previous works attribute this degradation to the motion of the objects. While the point clouds of stationary objects are inherently aligned after aggregation [88], producing denser and more complete geometry (fig. 3.1a), dynamic object point clouds become misaligned and distorted from motion (fig. 3.1b). Yang et al. [292] notice that such misalignment makes multi-frame aggregation unhelpful, even degrading performance for fast-moving objects. Chen et al. [30] argue that this effect poses an additional challenge due to different dynamic objects having different distorted point cloud patterns. This introduces a performance trade-off, as illustrated in fig. 3.1c: the detection of objects at different speeds is optimal using different numbers of frames.




(a) Stationary vehicle (b) Fast vehicle (c) Vehicle AP performance

Figure 3.1: Performance trade-off between stationary (<0.2 m/s), slow ([0.2,10) m/s), and fast (10 m/s) vehicles. The relative AP in (c) illustrates the improvement/degradation relative to using 3-frame fixed aggregation. (a) and (b) are examples of stationary and fast-moving vehicles after 16-frame fixed aggregation.

To address this challenge, SOTA multi-frame detectors have made use of attention-based feature-level aggregation to more effectively utilize information from past frames [30, 156, 292]. However, our experiments reveal that there is considerably more performance to be gained from modifying the input, before resorting to specific architectural designs, with the additional complexity and computation costs they entail.

In this chapter, we therefore propose VADet (Variable Aggregation Detection): a simple and effective alternative to fixed aggregation that sets the level of aggregation of an object according to its properties. As VADet operates at input level, it can be integrated into existing architectures to improve detection performance without significant modifications or computational overhead. Moreover, VADet's low latency supports its use in real-time applications.

The core of VADet is a function $\eta$ that maps each detected object to an empirically-optimal number of frames to aggregate. To construct $\eta$, we propose *random aggregation training* (RAT, section 3.2.1) to efficiently study the effects of fixed aggregation on detection performance, over a wide range of configurations.

We use RAT to analyze three representative object detection architectures, establishing that in addition to speed, as illustrated in fig. 3.1, object point density demonstrates another important trade-off (see fig. 3.5). The function $\eta$ is then constructed based on the training data to map an object's estimated speed and point density to a number of frames to aggregate, a process detailed in section 3.2.2.

Thanks to the per-object aggregation based on $\eta$, VADet can achieve good performance for objects with different speed and density in the same scene. Our results (section 3.4) show that

VADet consistently exceeds the performance of fixed aggregation, for a given architecture, and can surpass the performance of much more complex SOTA approaches.

## 3.2 Method

VADet addresses the performance trade-off associated with fixed aggregation by adaptively aggregating different types of objects with a different number of frames. To this end, we first introduce Random Aggregation Training (RAT) to enable a single detector to handle a wide range of input frame counts. We then describe our variable aggregation strategy.

### 3.2.1 Random Aggregation Training

Studying the impact of the number of input frames on different types of objects (e.g., stationary vs. dynamic) is a crucial component of our approach. Existing works tend to demonstrate the performance trade-offs of aggregation by evaluating multiple detectors trained separately with different fixed frame counts [88, 252, 292]. This is computationally expensive and thus is often done only for a few configurations. Moreover, we find that assessing performance differences between different input configurations, which can often be subtle, using multiple separately trained models is prone to high variance.

To efficiently explore the effects of frame counts on detection performance, we introduce random aggregation training (RAT), wherein a single detector is trained with input that has randomly varying numbers of aggregated frames per scene. To compensate for the increased variety of the input, we increase the number of training epochs accordingly. We find that even though the model's capacity remains unchanged, RAT allows the model to achieve equivalent or slightly better performance than detectors trained on different fixed configurations. This is demonstrated in Table 3.1 below.

| | **3-frame** | **4-frame** | **8-frame** | **16-frame** |
|---|---|---|---|---|
| Separate | 72.90 | 73.45 | 74.70 | 75.06 |
| RAT | 73.38 | 74.13 | 75.45 | 75.74 |

Table 3.1: Vehicle AP of a VoxelNeXt [33] detector trained with separate fixed configurations and RAT, evaluated with different input aggregations.

RAT thus offers several advantages. In terms of studying the effect of the input aggregation, it significantly reduces the computational cost because training a separate detector for each input

configuration is no longer required. This enables us to cover a broader range of frame counts than existing work and to more precisely determine the trade-offs for different types of object. Additionally, as the evaluation is done with a single model and varying input configurations, we find that RAT reduces the variance associated with training, providing us with more consistent results.

In this work, we also use RAT as a pre-training strategy. The detector trained with RAT serves as an ideal starting point for our proposed variable aggregation strategy, thanks to its ability to handle multiple input configurations.

### 3.2.2 Variable Aggregation

To address the performance trade-off between different types of objects, we propose per-object variable aggregation, which dynamically aggregates each detected object according to its properties, such as speed and point density.

In VADet, we first perform velocity estimation for each detected object. This serves two purposes: first, it allows us to identify the approximate locations of previously detected objects in the current frame using a constant velocity motion model, enabling us to aggregate each region separately using different aggregation strategies; second, it indicates the motion state of the object and is an important factor for determining the number of frames used in the aggregation. To estimate the velocity of the objects, we follow previous methods [20] and add channels to the regression task representing the $x$ and $y$ components of the velocity vector.

Formally, at timestamp $\tau$, we consider a previously detected bounding box $b_{\tau-1}$ in the coordinate system of the current frame with position $\mathbf{x}_{\tau-1}$, dimensions $(l_{\tau-1}, w_{\tau-1}, h_{\tau-1})$, heading $\theta_{\tau-1}$, estimated velocity $\mathbf{v}_{\tau-1}$, and $n_{\tau-1}$ points from the point cloud at timestamp $\tau - 1$ inside the bounding box. VADet aims to find a function $\eta(b_{\tau-1})$ that gives the empirically best number of frames to aggregate for each object at the current timestamp $\tau$.

The bounding box $b_{\tau-1}$ can be obtained directly from the previous frame's detection results, or initialized from the existing tracklets when a tracker is present. When no prior information about the object is available, which is the case for emerging objects, VADet falls back to using a fixed aggregation strategy for these unknown regions. Alternatively, proposals from a high recall region proposal network (RPN) can be used in place of $b_{\tau-1}$ to initialize VADet.

#### Learning Function $\eta$

To determine the number of aggregated frames that provide the best detection performance for each object, we consider two important factors: speed and point density (number of points per

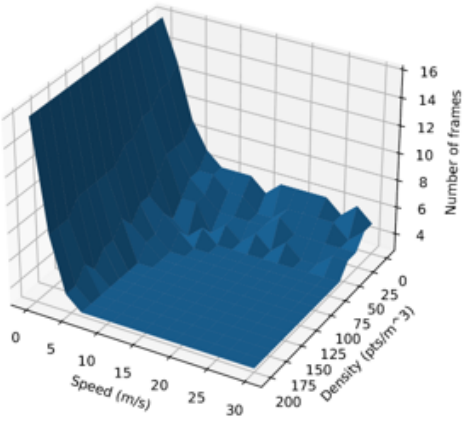



Figure 3.2: An example of the $\eta$ function computed with two factors: speed and density.

unit surface area). Aggregation changes the appearance of the point clouds for objects of different speeds due to motion distortion, as illustrated in fig. 3.1a and fig. 3.1b, and thus the optimal number of frames for aggregation varies with the object's speed. Additionally, as more frames are aggregated, the point density increases proportionally, affecting the number of points representing each object and consequently impacting the detection performance.

In practice, since neither of these factors can be accurately determined for a given object, we use the velocity prediction from the object detector to estimate its speed $\|v_{\tau-1}\|$, and approximate its point density $\rho_{\tau-1}$ using the predicted bounding box dimensions:

$$\rho_{\tau-1} = n_{\tau-1} / \left(l_{\tau-1} \cdot w_{\tau-1} + l_{\tau-1} \cdot h_{\tau-1} + w_{\tau-1} \cdot h_{\tau-1}\right) \tag{3.1}$$

Since the trade-offs for different types of objects can be different for each dataset and architecture, we obtain $\eta$ empirically by evaluating the object detector's performance on different types of objects over a wide range of input configurations on the training split. This is feasible thanks to RAT allowing the use of a single model for evaluation. Specifically, we implement $\eta$ as a piecewise function using a lookup table. The table is constructed by dividing the training set into subcategories of objects with different speeds and densities, and determining the frame count that leads to the highest average precision for each subcategory. An example of our $\eta$ function is shown in Figure 3.2.

While existing works have observed the effects of aggregation on objects with different speeds [30, 88, 292], the way aggregation interacts with objects with different point densities is not well-studied. In VADet, we establish point density as an additional factor that should be considered and evaluated.

**Input construction**

For each object, we first determine the approximate location $\hat{\mathbf{x}}_\tau^*$ of the object at the current timestamp $\tau$ according to the constant velocity model. This is given by

$$\hat{\mathbf{x}}_\tau^* = \mathbf{x}_{\tau-1} + \mathbf{v}_{\tau-1}/f, \tag{3.2}$$

where $f$ is the frame rate of the LiDAR point clouds.

To encompass all the points belonging to the object, including past points that could potentially fall outside of the bounding box, we enlarge the region of aggregation based on the speed of the object and the number of frames used in the aggregation. The final region of aggregation for an object, denoted $\hat{b}_\tau := (\hat{\mathbf{x}}_\tau, \hat{l}_\tau, \hat{w}_\tau, \hat{h}_\tau, \hat{\theta}_\tau)$, is given by

$$\hat{\mathbf{x}}_\tau = \hat{\mathbf{x}}_\tau^* - \frac{\mathbf{v}_{\tau-1} \cdot (\eta(b_{\tau-1}) - 1)}{2f}, \tag{3.3}$$

$$\hat{l}_\tau = \sigma \cdot l_{\tau-1} + \frac{|\mathbf{v}_{\tau-1}| \cdot (\eta(b_{\tau-1}) - 1)}{f}, \tag{3.4}$$

$$\hat{w}_\tau = \sigma \cdot w_{\tau-1}, \tag{3.5}$$

$$\hat{h}_\tau = \sigma \cdot h_{\tau-1}, \tag{3.6}$$

$$\hat{\theta}_\tau = \theta_{\tau-1}, \tag{3.7}$$

where $\sigma \geq 1$ is the enlargement factor that adds a margin to the aggregation region. Note that the second term in eq. (3.4) is used to enlarge the length of the object to include the misaligned "smudges" from object motion. Accordingly, the center of the aggregation region is adjusted in eq. (3.3).

Finally, for each region constructed from the process above, we aggregate the respective number of past frames in that region. In practice, this operation can be efficiently implemented by cropping the regions $\hat{b}_\tau$ for each previous frame $\tau - i$, $i < \eta(b_{\tau-1})$. Point clouds are corrected for ego motion by transformation $T_\tau^{-1} T_{\tau-i}$ before aggregation. This process is detailed in algorithm 1. The aggregated objects are then combined with the remaining background points outside of the aggregation regions.

## 3.3 Experimental Setup

To demonstrate our method is effective and can be easily applied to different architectures, we evaluate VADet using CenterPoint [298], VoxelNeXt [33], and DSVT [252] on the large scale

**Algorithm 1:** Variable Aggregation

**Input:** $\begin{cases} \text{point clouds } P_{\tau-n_{\max}+1}, \dots, P_\tau \\ \text{ego poses } T_{\tau-n_{\max}+1}, \dots, T_\tau \\ \text{previous detections } \mathbf{b}_{\tau-1} \end{cases}$

**Output:** aggregated object points $P_\tau^{\text{obj}}$

Initialize $P_\tau^{\text{obj}}$ to be an empty point cloud
Compute $\hat{\mathbf{b}}_\tau$ from $\mathbf{b}_{\tau-1}$ using eqs. (3.2) to (3.7)
**for** $i \leftarrow 0$ **to** $n_{\max} - 1$ **do**
  $P_{\tau-i}^{\text{corr}} \leftarrow P_{\tau-i}(T_\tau^{-1} T_{\tau-i})^\mathsf{T}$
  $\hat{\mathbf{b}}_\tau^i \leftarrow \left\{ \hat{b}_\tau \mid \hat{b}_\tau \in \hat{\mathbf{b}}_\tau, \eta(b_{\tau-1}) > i \right\}$
  $P_\tau^{\text{obj}} \leftarrow P_\tau^{\text{obj}} \oplus \text{Crop}(P_{\tau-i}^{\text{corr}}, \hat{\mathbf{b}}_\tau^i)$
**return** $P_\tau^{\text{obj}}$

Waymo Open Dataset [218]. CenterPoint, VoxelNeXt, and DSVT respectively represent the SOTA in dense voxel-based, fully sparse, and transformer-based 3D object detectors.

### 3.3.1 Dataset

The Waymo dataset is a large-scale autonomous driving dataset collected under a variety of traffic conditions in San Francisco, Phoenix, and Mountain View. It consists of 798 sequences for the training split, 202 sequences for the validation split, and 150 sequences for the held-back test split. Each sequence is approximately 20 seconds long.

Waymo uses multiple LiDAR sensors operating at 10 Hz, resulting in approximately 200 point clouds per sequence. The main sensor is a top-mounted proprietary 64-beam rotating LiDAR. There are also four close-range LiDARs mounted to the side of the vehicle. In addition to the intensity channel, Waymo's sensors also produce elongation for each point. We use points from all five LiDARs by concatenating them in the ego vehicle coordinate system.

### 3.3.2 Evaluation Metrics

For overall object detection performance, we use the official Waymo evaluation suite. Unless specified otherwise, we report the level 2 average precision (AP) for Vehicle, which includes very sparse objects ($\leq 5$ points). The IoU threshold used for matching true positives is 0.7.

| Metric | Dynamic | Subsets (m/s) | | | | | | | | | | | Weighted |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ≥0.2 m/s | [0.2,1) | [1,3) | [3,5) | [5,7) | [7,9) | [9,11) | [11,13) | [13,15) | [15,17) | [17,20) | ≥20 | average |
| Waymo | 72.9 | 63.0 | 62.4 | 62.9 | 66.5 | 68.9 | 66.8 | 64.1 | 61.8 | 64.0 | 67.0 | 62.9 | 64.7 |
| Ours | 74.4 | 71.6 | 69.8 | 72.2 | 74.6 | 76.6 | 76.3 | 75.7 | 75.3 | 78.3 | 81.6 | 78.3 | 74.4 |

Table 3.2: Dynamic vehicle AP of a CenterPoint model on the Waymo dataset using different subset evaluation metrics.

Evaluating a specific subset of the objects (e.g., dynamic objects) for more detailed analysis requires more careful handling due to false positives that cannot be matched with any ground truth. Waymo evaluation suite provides functionalities to perform such evaluation. Specifically, the subset precision and recall are defined as follows:

$$\mathrm{Prec}_{\mathrm{waymo}} = \frac{\mathrm{TP}_{\mathrm{subset}}}{\mathrm{TP}_{\mathrm{subset}} + \mathrm{FP}_{\mathrm{subset}} + \mathrm{FP}_{\mathrm{unknown}}}, \tag{3.8}$$

$$\mathrm{Rec}_{\mathrm{waymo}} = \frac{\mathrm{TP}_{\mathrm{subset}}}{\mathrm{TP}_{\mathrm{subset}} + \mathrm{FN}_{\mathrm{subset}}}. \tag{3.9}$$

$\mathrm{TP}_{\mathrm{subset}}$ and $\mathrm{FN}_{\mathrm{subset}}$ are the number of true positives and false negatives within the subset of objects. $\mathrm{FP}_{\mathrm{subset}}$ is the number of false positives that overlap with some objects in the subset but do not meet the IoU threshold. $\mathrm{FP}_{\mathrm{unknown}}$ is the number of objects that do not overlap with any ground truth.

However, we argue that this formulation cannot correctly reflect the detection performance of a subset because $\mathrm{FP}_{\mathrm{unknown}}$ is independent of the subset and biases the precision depending on the size of the subset. The consequence is that subsets of different sizes are incomparable and the weighted sum of subset precisions underestimates the precision of the union of subsets.

To address this issue, we introduce a definition of precision that weights $\mathrm{FP}_{\mathrm{unknown}}$ by the proportion of objects in the subset:

$$\mathrm{Prec}_{\mathrm{subset}} = \frac{\mathrm{TP}_{\mathrm{subset}}}{\mathrm{TP}_{\mathrm{subset}} + \mathrm{FP}_{\mathrm{subset}} + \frac{N_{\mathrm{subset}}}{N_{\mathrm{total}}} \cdot \mathrm{FP}_{\mathrm{unknown}}} \tag{3.10}$$

$N_{\mathrm{subset}}$ and $N_{\mathrm{total}}$ are the number of objects in the subset and the total number of objects, respectively. The definition of recall remains unchanged. In contrast to the standard Waymo metric, this formulation allows the comparison and combination of subset APs, as illustrated in Table 3.2. We note, in particular, that when using our formulation the weighted average of subset APs weighted by subset size is closer to the AP of the union of the subsets.

| | VADet | 3f | 4f | 5f | 6f | 7f | 8f | 9f | 10f | 11f | 12f | 13f | 14f | 15f | 16f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CenterPoint | **71.5** | 68.1 | 68.9 | 69.5 | 70.0 | 70.3 | 70.5 | 70.7 | 70.8 | 70.9 | 71.0 | 71.0 | 71.0 | 71.0 | 70.9 |
| VoxelNeXt | **76.5** | 73.4 | 74.1 | 74.6 | 75.0 | 75.3 | 75.4 | 75.6 | 75.7 | 75.7 | 75.8 | 75.8 | 75.8 | 75.8 | 75.7 |
| DSVT-P | **74.5** | 69.5 | 70.3 | 70.8 | 71.4 | 71.7 | 72.0 | 72.2 | 72.4 | 72.6 | 72.7 | 72.8 | 73.0 | 73.0 | 73.2 |

Table 3.3: Overall vehicle AP on the Waymo validation split. The best performance is in bold and the second best is underlined.

### 3.3.3 Implementation Details

#### Architectures

CenterPoint [298] and VoxelNeXt [33] are both single stage CNN-based 3D object detectors. The input point cloud first undergoes voxelization and is then fed to a sparse convolutional backbone. Multiple detection heads are used to separately produce bounding box attributes, including confidence score, location, and box dimensions. DSVT [252] is an emerging transformer-based 3D object detector. In DSVT, the traditional convolution backbone is replaced with multiple transformer blocks consisting of shifted window and partition-based self-attention operations.

Following [20], we add a two-layer regression head to predict an object's velocity vector for all architectures. For DSVT, we use the pillar variant (which we denote DSVT-P). As the dynamic voxelization adopted by the original work cannot be scaled beyond 4-frame aggregation on our hardware, we use a traditional static voxelization where for each voxel, at most 40 points are randomly selected and processed. To make the computation tractable, we further reduce the input channels from 192 to 96, and the hidden channels from 384 to 192.

#### Training

The baseline models are trained on the entire Waymo training split for 20 epochs across 8 NVIDIA A6000 GPUs using the proposed RAT strategy. We use the Adam optimizer and a one-cycle learning rate schedule, with an initial learning rate of 0.0003 and a maximum learning rate of 0.003. We use a total batch size of 32 for CenterPoint, 32 for VoxelNeXt, and 16 for DSVT-P.

We initialize our VADet models with the weights from the baseline models, then fine-tune them for an additional epoch using cosine learning rate decay with an initial learning rate of 0.0001. Since our models rely on information cached from previous frames, including point clouds and predictions, the standard frame-based shuffling cannot be applied during training.

Instead, we divide the dataset into mini-sequences with a maximum of 32 frames and shuffle the mini-sequences. This introduces randomness while ensuring that frames appear in their correct sequence. Furthermore, as the performance of a model can fluctuate during training (due to ongoing optimization), following [62], we load the offline predictions from the baseline models using 3-frame fixed aggregation during training.

#### Specifying $\eta$

We implement $\eta$ as a lookup table: we subdivide the training dataset based on speed and point cloud density and empirically determine the frame count that leads to the highest average precision for each subcategory (section 3.2.2). The speed and density thresholds are set to [0.00, 0.20, 1.55, 3.63, 5.90, 81.6, 11.34, 17.53] m/s and [0.00, 0.68, 1.86, 3.86, 8.02, 18.81, 71.37] pts/m$^2$ respectively. They are chosen based on the training set object statistics to ensure a sufficient number of objects in each speed-density combination for evaluation. For each combination, we then evaluate the baseline model on 3–16-frame input and select the frame count with the highest AP performance.

As certain speed-density combinations can contain very few objects, the performance variance can result in an unsmooth $\eta$ function, as exemplified in Figure 3.2. Additionally, we observe that a small number of combinations do not have any corresponding objects. In this work, we do not apply any smoothing operation, and default to a fixed 3-frame aggregation for the empty cases. However, with sufficient training data and coverage, we expect the $\eta$ function to appear smooth with no gaps. Alternatively, a smooth closed-form approximation can be derived from $\eta$ to express the relationship between different factors and the optimal frame count.

## 3.4 Results and Analysis

We present the overall performance of VADet compared to baseline models and SOTA methods in Section 3.4.1. To demonstrate the better trade-offs achieved by VADet, in Section 3.4.3 we perform a detailed evaluation based on speed and point cloud density.

### 3.4.1 Overall Performance

Table 3.3 shows that, for all three architectures, VADet demonstrates superior performance compared to baseline models using different fixed frame counts. In particular, VADet using CenterPoint achieves 71.5 AP compared to 71.0 AP using 12, 13, 14, or 15-frame aggregation. For

| Method | # frames | L1 AP/APH | L2 AP/APH |
|---|---|---|---|
| SECOND [280] | 1 | 72.3/71.7 | 63.9/63.3 |
| PointPillars [119] | 1 | 72.1/71.5 | 63.6/63.1 |
| CenterPoint-pillar [298] | 1 | 76.1/75.5 | 68.0/67.5 |
| CenterPoint-voxel [298] | 1 | 76.6/76.0 | 68.9/68.4 |
| Part-A2-Net [208] | 1 | 77.1/76.5 | 68.5/68.0 |
| PV-RCNN [207] | 1 | 77.5/76.9 | 69.0/68.4 |
| SST [59] | 1 | 74.2/73.8 | 65.5/65.1 |
| AFDetV2 [87] | 1 | 77.6/77.1 | 69.7/69.2 |
| PV-RCNN++ [209] | 1 | 78.8/78.2 | 70.3/69.7 |
| SWFormer [220] | 1 | 77.8/77.3 | 69.2/68.8 |
| FSD [60] | 1 | 77.8/77.3 | 68.9/68.5 |
| VoxelNeXt [33] | 1 | 78.2/77.7 | 69.9/69.4 |
| OcTr [315] | 1 | 78.1/77.6 | 69.8/69.3 |
| PillarNeXt-B [129] | 1 | 78.4/77.9 | 70.2/69.8 |
| DSVT-P [252] | 1 | 79.3/78.8 | 70.9/70.5 |
| DSVT-V [252] | 1 | 79.7/79.3 | 71.4/71.0 |
| PillarNeXt-B [129] | 3 | 80.6/80.1 | 72.9/72.4 |
| DSVT-pillar [252] | 4 | 81.7/81.2 | 73.8/73.4 |
| DSVT-voxel [252] | 4 | 81.8/81.4 | 74.1/73.6 |
| FSD++ [62] | 7 | 81.4/80.9 | 73.3/72.9 |
| CenterFormer [319] | 8 | 78.8/78.3 | 74.3/73.8 |
| 3D-MAN [292] | 16 | 74.5/74.0 | 67.6/67.1 |
| MPPNet [30] | 16 | <u>82.7/82.3</u> | <u>75.4/75.0</u> |
| VADet-CenterPoint | 3–16 | 79.0/78.5 | 71.7/71.3 |
| VADet-DSVT-P | 3–16 | 82.0/81.6 | 74.5/74.1 |
| VADet-VoxelNeXt | 3–16 | **83.9/83.4** | **76.6/76.1** |

Table 3.4: Overall vehicle performance on the Waymo validation split compared with SOTA multi-frame detectors. The best performance according to APH is highlighted in bold and the second best is underlined.

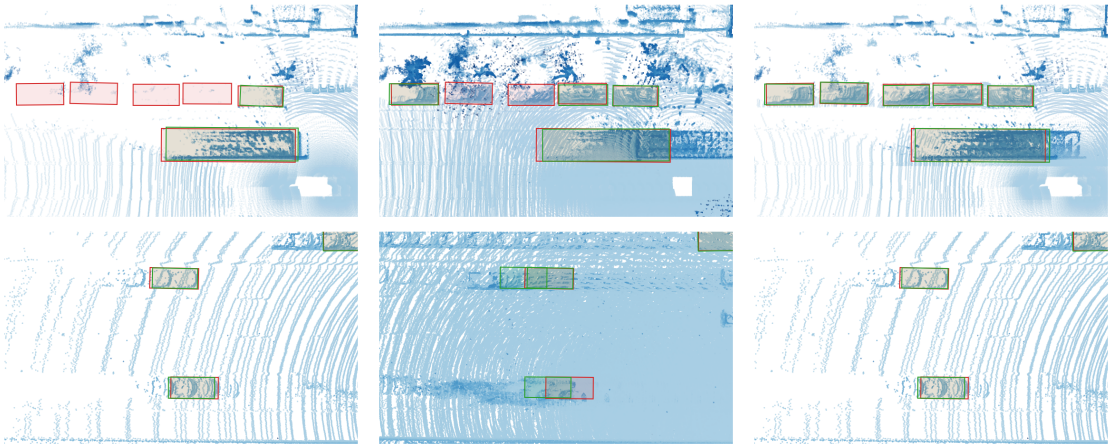
| Method | # frames | Modality | L2 AP/APH |
|---|---|---|---|
| AFDetV2 [87] | 2 | L | 74.3/73.9 |
| PV-RCNN++ [209] | 2 | L | 76.3/75.9 |
| SWFormer [220] | 3 | L | 75.0/74.7 |
| PillarNeXt-B [129] | 3 | L | 76.2/75.8 |
| FSD++ [62] | 7 | L | 77.1/76.7 |
| 3D-MAN [292] | 16 | L | 70.4/70.0 |
| MPPNet [30] | 16 | L | 77.3/76.9 |
| CenterFormer [319] | 16 | L | <u>78.7/78.3</u> |
| BEVFusion [146] | 3 | C+L | 77.9/77.5 |
| DeepFusion [133] | 5 | C+L | 76.1/75.7 |
| HorizonLiDAR3D [51] | 5 | C+L | 78.2/77.8 |
| LoGoNet [132] | 5 | C+L | <u>79.7/79.3</u> |
| VADet-VoxelNeXt | 3–16 | L | **79.8/79.4** |

Table 3.5: Overall vehicle performance on the Waymo test split compared with other methods (without TTA or ensemble). The best performance according to APH is highlighted in bold and the second best is underlined.

VoxelNeXt, VADet achieves 76.5 AP compared to 75.8 AP using 12, 13, 14, or 15-frame aggregation. DSVT-P with VADet achieves 74.5 AP compared to 73.2 AP using 16-frame aggregation.

When compared with SOTA object detection methods in Table 3.4 and Table 3.5, VADet-equipped models also have competitive performance on both the validation and test splits. Most notably, our single-stage VADet-VoxelNeXt achieves 76.1 and 79.4 level 2 APH on the validation and test split respectively, outperforming two-stage multi-frame methods such as MPPNet [30] by a large margin, with lower computation overhead. Specifically, we measure that the second stage proposed by MPPNet introduces an additional 900–2500 ms latency over the base detector, while VADet only requires an additional 50 ms overhead for input aggregation, which we believe can be further optimized with a GPU-accelerated implementation. Furthermore, with a powerful backbone such as VoxelNeXt, our method can match the performance of recent SOTA camera-LiDAR fusion methods such as LoGoNet [132].

Our results demonstrate that VADet can effectively utilize multiple frames to achieve SOTA detection performance, suggesting that by addressing various performance trade-offs with carefully constructed input, a simple single-stage object detection architecture such as VoxelNeXt can outperform much more complex SOTA methods.



(a) 3-frame (b) 16-frame (c) VADet

Figure 3.3: Qualitative results comparing 3-frame and 16-frame fixed aggregation with VADet. Red and green bounding boxes are ground truth and predictions, respectively. Predictions are filtered with 0.5 confidence threshold for visual clarity.

### 3.4.2 Qualitative Results

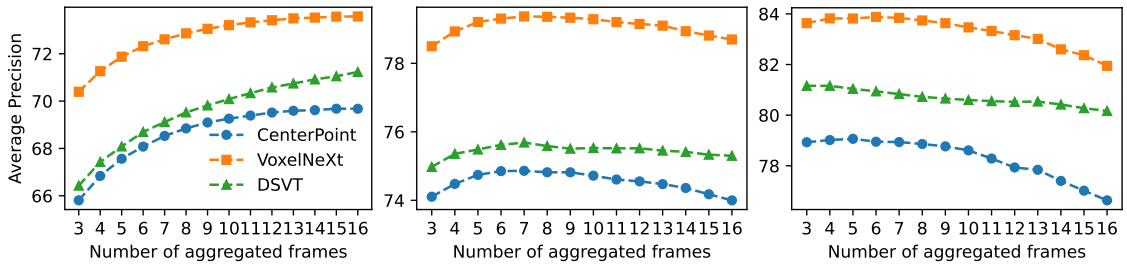
Qualitative results comparing 3-frame and 16-frame fixed aggregation with VADet are shown in fig. 3.3. Overall, we observe that 16-frame input appears much denser than 3-frame and VADet. This is especially noticeable for background points, which may not be beneficial to detecting objects and introduce unnecessary computation.

The 3-frame model is unable to detect the occluded stationary vehicles depicted in the upper images, due to the lack of points. While these objects appear more complete with 16 frames, the model can only partially detect them. We hypothesize this is due to the dense background points leading to confusion. VADet, on the other hand, aggregates only object points while keeping the background sparse, and therefore can accurately detect all vehicles.

Similarly, for the fast-moving vehicles seen in the lower images, 16-frame aggregation results in long "smudges" around each vehicle, leading to inaccurate localization. In these situations, VADet does not over-aggregate these objects and benefits from the more accurate detections achieved by 3-frame input.

| | Speed | VADet | 3f | 4f | 5f | 6f | 7f | 8f | 9f | 10f | 11f | 12f | 13f | 14f | 15f | 16f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CenterPoint | Stationary | **70.3** | 65.8 | 66.8 | 67.6 | 68.1 | 68.5 | 68.8 | 69.1 | 69.3 | 69.4 | 69.5 | 69.6 | 69.6 | 69.7 | 69.7 |
| | Slow | **75.0** | 74.1 | 74.5 | 74.7 | 74.9 | 74.9 | 74.8 | 74.8 | 74.7 | 74.6 | 74.6 | 74.5 | 74.4 | 74.2 | 74.0 |
| | Fast | **79.5** | 78.9 | 79.0 | 79.1 | 79.0 | 78.9 | 78.9 | 78.8 | 78.6 | 78.3 | 77.9 | 77.8 | 77.4 | 77.0 | 76.6 |
| VoxelNeXt | Stationary | **74.3** | 70.4 | 71.3 | 71.9 | 72.3 | 72.6 | 72.9 | 73.1 | 73.2 | 73.3 | 73.4 | 73.5 | 73.5 | 73.6 | 73.6 |
| | Slow | **79.6** | 78.5 | 78.9 | 79.2 | 79.3 | 79.4 | 79.4 | 79.3 | 79.3 | 79.2 | 79.2 | 79.1 | 78.9 | 78.8 | 78.7 |
| | Fast | **84.2** | 83.6 | 83.8 | 83.8 | 83.9 | 83.8 | 83.7 | 83.6 | 83.5 | 83.3 | 83.2 | 83.0 | 82.6 | 82.4 | 81.9 |
| DSVT-P | Stationary | **72.3** | 66.4 | 67.4 | 68.1 | 68.7 | 69.1 | 69.5 | 69.8 | 70.1 | 70.3 | 70.6 | 70.7 | 70.9 | 71.0 | 71.2 |
| | Slow | **77.3** | 75.0 | 75.4 | 75.5 | 75.6 | 75.7 | 75.6 | 75.5 | 75.5 | 75.5 | 75.5 | 75.5 | 75.4 | 75.3 | 75.3 |
| | Fast | **82.6** | 81.2 | 81.2 | 81.0 | 80.9 | 80.8 | 80.7 | 80.7 | 80.6 | 80.6 | 80.5 | 80.5 | 80.4 | 80.3 | 80.2 |

Table 3.6: Vehicle AP breakdown by speed. The best performance is highlighted in bold and the second best is underlined.




(a) Stationary (b) Slow (c) Fast

Figure 3.4: AP vs. the number of frames for stationary (<0.2 m/s), slow ([0.2,10) m/s), and fast-moving (≥10 m/s) vehicles.

### 3.4.3 Breakdown Analysis

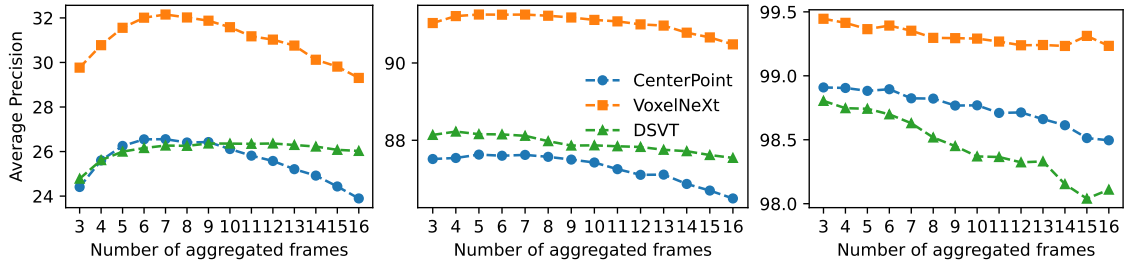
The overall performance degradation at 16-frame fixed aggregation, if any, appears to be minuscule for the baseline models: just 0.1% degradation for CenterPoint and VoxelNeXt. This is due to the extreme imbalance between different types of objects in the dataset. For instance, the overall results are skewed in favour of the approximately 80% stationary vehicles in the dataset, which negatively affects the performance of dynamic objects. As a result, we perform a breakdown analysis to highlight the performance trade-offs and demonstrate how VADet can benefit objects that are negatively impacted by fixed aggregation.

#### Speed

To illustrate the impact of input aggregation on objects with different speeds, we divide the vehicle class into stationary (<0.2 m/s), slow ([0.2,10) m/s), and fast (≥10 m/s) subcategories.

| | Density | VADet | 3f | 4f | 5f | 6f | 7f | 8f | 9f | 10f | 11f | 12f | 13f | 14f | 15f | 16f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CenterPoint | Sparse | **27.6** | 24.4 | 25.6 | 26.3 | 26.5 | 26.6 | 26.4 | 26.4 | 26.1 | 25.8 | 25.6 | 25.2 | 24.9 | 24.4 | 23.9 |
| | Medium | **87.8** | 87.5 | 87.5 | 87.6 | 87.6 | 87.6 | 87.6 | 87.5 | 87.4 | 87.3 | 87.1 | 87.1 | 86.9 | 86.7 | 86.5 |
| | Dense | **98.9** | **98.9** | **98.9** | **98.9** | **98.9** | 98.8 | 98.8 | 98.8 | 98.8 | 98.7 | 98.7 | 98.7 | 98.6 | 98.5 | 98.5 |
| VoxelNeXt | Sparse | **32.8** | 29.8 | 30.8 | 31.6 | 32.0 | 32.2 | 32.0 | 31.9 | 31.6 | 31.2 | 31.0 | 30.8 | 30.1 | 29.8 | 29.3 |
| | Medium | **91.5** | 91.0 | 91.2 | 91.3 | 91.2 | 91.3 | 91.2 | 91.2 | 91.1 | 91.1 | 91.0 | 91.0 | 90.8 | 90.7 | 90.5 |
| | Dense | 99.3 | **99.4** | **99.4** | **99.4** | **99.4** | **99.4** | 99.3 | 99.3 | 99.3 | 99.3 | 99.2 | 99.2 | 99.2 | 99.3 | 99.2 |
| DSVT-P | Sparse | **28.0** | 24.8 | 25.6 | 26.0 | 26.2 | 26.3 | 26.3 | 26.4 | 26.4 | 26.3 | 26.4 | 26.3 | 26.2 | 26.1 | 26.0 |
| | Medium | **89.9** | 88.1 | 88.2 | 88.2 | 88.2 | 88.1 | 88.0 | 87.9 | 87.9 | 87.8 | 87.8 | 87.8 | 87.7 | 87.6 | 87.5 |
| | Dense | **99.0** | 98.8 | 98.7 | 98.7 | 98.7 | 98.6 | 98.5 | 98.5 | 98.4 | 98.4 | 98.3 | 98.3 | 98.2 | 98.0 | 98.1 |

Table 3.7: Dynamic vehicle AP performance breakdown by point cloud density. The best performance is highlighted in bold and the second best is underlined.




(a) Sparse (b) Medium (c) Dense

Figure 3.5: AP vs. the number of frames for sparse ($<$2 pts/m$^2$), medium ((2,100] pts/m$^2$), and dense ($>$100 pts/m$^2$) dynamic vehicles.

Stationary, slow, and fast vehicles respectively amount to 79.7%, 14.2%, and 6.1% of the Waymo validation set. The AP performance for each subcategory is reported in Table 3.6.

Our results for the baseline models are consistent with the trade-off observed in existing work and further suggest that this effect is consistent across different architectures. Moreover, the results show that the optimal number of frames can be different for objects with different speeds. Specifically, the stationary vehicle performance (fig. 3.4a) increases as more frames are used for aggregation: all baseline models achieve the best performance with 16-frame input, suggesting more aggregation is beneficial. Slow vehicles (fig. 3.4b), on the other hand, reach maximum performance at 7 frames, while for fast-moving vehicles (fig. 3.4c), a large degradation can be observed when more frames are used, indicating aggregating fewer frames is more favourable.

VADet, on the other hand, does not exhibit such a performance trade-off. For all three architectures and subcategories, VADet achieves higher AP than the best fixed aggregation con-

| Speed | All | | | Stationary | | | | Slow | | | | Fast | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Density | Sparse | Med. | Dense | All | Sparse | Med. | Dense | All | Sparse | Med. | Dense | All | Sparse | Med. | Dense |
| % | 33.5 | 59.8 | 6.7 | 79.7 | 28.7 | 46.0 | 5.0 | 14.2 | 3.5 | 9.4 | 1.3 | 6.1 | 1.4 | 4.3 | 0.4 |

Table 3.8: The percentage of Waymo vehicles in each speed-density subcategory.

figuration in each respective subcategory (underlined in Table 3.6). This not only highlights the effectiveness of VADet at mitigating the performance trade-off between objects at different speeds but also demonstrates the applicability of VADet to various architectures.

### Point Density

To demonstrate that input aggregation can also lead to a performance trade-off between objects with different point cloud densities, we evaluate and report the performance of vehicles with different point cloud densities. Using eq. (3.1), we divide the vehicle class into sparse ($<2$ pts/m$^2$), medium ([2,100) pts/m$^2$), and dense ($\geq 100$ pts/m$^2$) subcategories.

For stationary objects, we observe the same trend seen in fig. 3.4a, suggesting the performance is not influenced by the point cloud density. The following analysis therefore focuses on dynamic objects ($\geq 0.2$ m/s), with the full results for stationary objects in following section. The dynamic vehicles in the validation set contain 23.9% sparse, 67.9% medium, and 8.2% dense vehicles according to our definition. The AP performance for each subcategory is detailed in Table 3.7.

While we have previously observed that dynamic objects favour fewer input frame counts, we notice in fig. 3.5a that sparse objects can still benefit from more input aggregation: up to 7-frame aggregation for CenterPoint and VoxelNeXt, and up to 12-frame aggregation for DSVT-P. On the other hand, for denser objects (figs. 3.5b and 3.5c), higher frame counts become harmful to the detection performance. All three architectures achieve the best performance for medium density vehicles with 5-frame input, and dense vehicles with 3-frame input. Our results highlight a trade-off that has not been studied in existing work and underscores the necessity of considering point density in our proposed approach.

Compared to fixed aggregation, VADet is comparable to the best performance in each subcategory, surpassing fixed aggregation in many cases. This demonstrates that VADet is also effective at addressing the trade-off between objects with different point densities for different architectures.

**Full Breakdown Results**

For the full breakdown analysis, we consider each speed and point cloud density combination based on the same subdivisions used in the previous sections, and perform evaluation on all subcategories. Table 3.8 shows the composition of vehicles in the Waymo dataset. The results are presented in Table 3.9.

As previously mentioned, while we notice a performance trade-off between dynamic objects with different point cloud densities, we do not observe this effect for stationary objects, which is likely due to the stationary object point clouds being inherently well-aligned and coherent after aggregation. Using more frames therefore does not introduce distortions that could deteriorate the detection performance.

## 3.5 Limitations and Future Work

VADet is a a model-agnostic alternative to fixed aggregation, and thus its performance is dependent on the architecture that it is applied to. While experiments on three current SOTA object detectors demonstrate that VADet can consistently out-perform the baseline methods, its benefits may be less pronounced for architectures that either suffer less from performance trade-offs or do not experience such trade-offs at all. For instance, an architecture capable of effectively attending to the relevant points may mitigate the impact of points from previous frames on dynamic objects. Future work could investigate additional architectures, especially the more recent point-based transformers.

Likewise, classes that do not exhibit significant performance issues related to the amount of aggregation would not benefit as much from VADet. In this thesis, we have chosen to focused on the Waymo vehicle class not only as a common expedient to simplify comparisons, but also due to the speed-related trade-off being most apparent in vehicle class. Pedestrians and cyclists, on the other hand, typically do not vary as much in speed and rarely remain stationary in the same way vehicles do (e.g., due to deformation). Classes like these might not benefit as much from VADet as vehicles do.

Moreover, VADet is designed to prevent "over-aggregation." For objects that do not benefit from more aggregation, VADet will effectively reduce to a few-frame aggregation, in which case the model will not take full advantage of temporal information. In these situations, more advanced methods would be required to incorporate temporal data without causing confusion or degradation in performance.

| Architecture | Speed | Density | VADet | 3f | 4f | 5f | 6f | 7f | 8f | 9f | 10f | 11f | 12f | 13f | 14f | 15f | 16f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CenterPoint | All | Sparse | **35.5** | 25.3 | 27.2 | 28.6 | 29.6 | 30.4 | 31.0 | 31.4 | 31.7 | 31.8 | 32.0 | 32.1 | 32.1 | 32.1 | 32.1 |
| | | Medium | **85.0** | 83.4 | 83.8 | 84.1 | 84.3 | 84.5 | 84.6 | 84.7 | 84.7 | 84.7 | 84.8 | 84.8 | 84.7 | 84.7 | 84.6 |
| | | Dense | **98.7** | 98.5 | 98.5 | 98.5 | 98.6 | 98.6 | 98.6 | 98.6 | 98.6 | 98.6 | 98.6 | 98.6 | 98.6 | 98.6 | 98.5 |
| | Stationary | All | **70.3** | 65.8 | 66.8 | 67.6 | 68.1 | 68.5 | 68.8 | 69.1 | 69.3 | 69.4 | 69.5 | 69.6 | 69.6 | 69.7 | 69.7 |
| | | Sparse | **36.8** | 25.5 | 27.6 | 29.1 | 30.2 | 31.2 | 31.9 | 32.4 | 32.8 | 33.1 | 33.3 | 33.4 | 33.5 | 33.6 | 33.7 |
| | | Medium | **84.2** | 82.2 | 82.7 | 83.1 | 83.3 | 83.6 | 83.7 | 83.9 | 83.9 | 84.0 | 84.1 | 84.1 | 84.1 | 84.1 | 84.0 |
| | | Dense | **98.6** | 98.4 | 98.4 | 98.4 | 98.5 | 98.5 | 98.5 | 98.5 | 98.5 | 98.6 | 98.6 | 98.6 | 98.6 | 98.6 | 98.6 |
| | Slow | All | **75.0** | 74.1 | 74.5 | 74.7 | 74.9 | 74.9 | 74.8 | 74.8 | 74.7 | 74.6 | 74.6 | 74.5 | 74.4 | 74.2 | 74.0 |
| | | Sparse | **25.9** | 22.2 | 23.7 | 24.5 | 25.0 | 25.0 | 25.0 | 25.1 | 24.9 | 24.6 | 24.6 | 24.2 | 24.1 | 23.7 | 23.3 |
| | | Medium | **86.6** | 86.3 | 86.4 | 86.5 | 86.5 | 86.5 | 86.5 | 86.4 | 86.4 | 86.3 | 86.2 | 86.2 | 86.1 | 86.0 | 85.8 |
| | | Dense | **99.2** | 99.2 | 99.2 | 99.2 | 99.2 | 99.2 | 99.1 | 99.1 | 99.0 | 99.0 | 99.0 | 98.9 | 98.8 | 98.7 | 98.7 |
| | Fast | All | **79.5** | 78.9 | 79.0 | 79.1 | 79.0 | 78.9 | 78.9 | 78.8 | 78.6 | 78.3 | 77.9 | 77.8 | 77.4 | 77.0 | 76.6 |
| | | Sparse | **32.1** | 29.8 | 30.4 | 30.8 | 30.6 | 30.4 | 30.0 | 29.8 | 29.3 | 28.8 | 28.2 | 27.8 | 27.0 | 26.4 | 25.4 |
| | | Medium | **90.3** | 90.1 | 90.0 | 90.0 | 89.9 | 89.9 | 89.9 | 89.7 | 89.7 | 89.3 | 89.0 | 89.0 | 88.6 | 88.3 | 88.0 |
| | | Dense | 97.8 | 97.8 | 97.8 | 97.7 | 97.8 | 97.7 | 97.8 | 97.8 | **97.9** | 97.8 | 97.8 | **97.9** | 97.8 | **97.9** | 97.8 |
| VoxelNeXt | All | Sparse | **40.1** | 30.6 | 32.5 | 33.9 | 35.0 | 35.7 | 36.2 | 36.6 | 36.9 | 37.1 | 37.3 | 37.4 | 37.4 | 37.5 | 37.4 |
| | | Medium | **88.6** | 87.2 | 87.6 | 87.8 | 87.9 | 88.0 | 88.1 | 88.1 | 88.2 | 88.2 | 88.2 | 88.2 | 88.2 | 88.1 | 88.1 |
| | | Dense | 98.9 | **99.1** | **99.1** | **99.1** | **99.1** | 99.0 | 99.0 | 99.0 | **99.1** | 99.0 | 99.0 | **99.1** | **99.1** | **99.1** | **99.1** |
| | Stationary | All | **74.3** | 70.4 | 71.3 | 71.9 | 72.3 | 72.6 | 72.9 | 73.1 | 73.2 | 73.3 | 73.4 | 73.5 | 73.5 | 73.6 | 73.6 |
| | | Sparse | **41.2** | 30.8 | 32.9 | 34.4 | 35.5 | 36.3 | 36.9 | 37.4 | 37.8 | 38.1 | 38.4 | 38.5 | 38.6 | 38.8 | 38.8 |
| | | Medium | **87.7** | 86.1 | 86.5 | 86.7 | 86.9 | 87.0 | 87.1 | 87.2 | 87.3 | 87.3 | 87.4 | 87.4 | 87.4 | 87.4 | 87.4 |
| | | Dense | 98.8 | **99.0** | **99.0** | **99.0** | **99.0** | 98.9 | 98.9 | **99.0** | **99.0** | **99.0** | **99.0** | **99.0** | **99.0** | **99.0** | **99.0** |
| | Slow | All | **79.6** | 78.5 | 78.9 | 79.2 | 79.3 | 79.4 | 79.4 | 79.3 | 79.3 | 79.2 | 79.2 | 79.1 | 78.9 | 78.8 | 78.7 |
| | | Sparse | **30.8** | 27.3 | 28.5 | 29.6 | 30.1 | 30.4 | 30.3 | 30.4 | 30.2 | 29.9 | 29.7 | 29.5 | 29.1 | 28.8 | 28.5 |
| | | Medium | **90.4** | 89.9 | 90.1 | 90.2 | 90.2 | 90.2 | 90.2 | 90.1 | 90.1 | 90.1 | 90.0 | 90.0 | 89.9 | 89.8 | 89.7 |
| | | Dense | **99.6** | **99.6** | **99.6** | **99.6** | **99.6** | **99.6** | 99.5 | 99.5 | 99.5 | 99.5 | 99.4 | 99.4 | 99.4 | 99.5 | 99.4 |
| | Fast | All | **84.2** | 83.6 | 83.8 | 83.8 | 83.9 | 83.8 | 83.7 | 83.6 | 83.5 | 83.3 | 83.2 | 83.0 | 82.6 | 82.4 | 81.9 |
| | | Sparse | **37.8** | 35.9 | 36.4 | 36.5 | 36.8 | 36.7 | 36.3 | 35.7 | 35.1 | 34.5 | 34.3 | 33.8 | 32.7 | 32.4 | 31.3 |
| | | Medium | **93.8** | 93.4 | 93.5 | 93.5 | 93.5 | 93.4 | 93.4 | 93.4 | 93.3 | 93.2 | 93.1 | 93.0 | 92.7 | 92.5 | 92.2 |
| | | Dense | 98.3 | **98.8** | 98.7 | 98.6 | 98.7 | 98.6 | 98.6 | 98.6 | 98.6 | 98.7 | **98.8** | 98.7 | 98.7 | **98.8** | 98.6 |
| DSVT-P | All | Sparse | **36.8** | 25.2 | 27.0 | 28.4 | 29.5 | 30.4 | 31.2 | 31.9 | 32.5 | 33.1 | 33.6 | 34.0 | 34.4 | 34.7 | 35.1 |
| | | Medium | **87.2** | 84.1 | 84.6 | 84.8 | 85.1 | 85.2 | 85.3 | 85.4 | 85.5 | 85.6 | 85.6 | 85.6 | 85.7 | 85.7 | 85.8 |
| | | Dense | **98.8** | 98.3 | 98.4 | 98.4 | 98.4 | 98.4 | 98.4 | 98.4 | 98.4 | 98.4 | 98.4 | 98.4 | 98.4 | 98.3 | 98.3 |
| | Stationary | All | **72.3** | 66.4 | 67.4 | 68.1 | 68.7 | 69.1 | 69.5 | 69.8 | 70.1 | 70.3 | 70.6 | 70.7 | 70.9 | 71.0 | 71.2 |
| | | Sparse | **38.1** | 25.2 | 27.3 | 28.8 | 30.1 | 31.1 | 32.0 | 32.8 | 33.5 | 34.1 | 34.7 | 35.2 | 35.7 | 36.1 | 36.5 |
| | | Medium | **86.4** | 82.9 | 83.5 | 83.8 | 84.2 | 84.3 | 84.5 | 84.7 | 84.8 | 84.9 | 85.0 | 85.0 | 85.1 | 85.2 | 85.2 |
| | | Dense | **98.8** | 98.1 | 98.2 | 98.3 | 98.3 | 98.4 | 98.4 | 98.4 | 98.4 | 98.4 | 98.4 | 98.5 | 98.4 | 98.4 | 98.4 |
| | Slow | All | **77.3** | 75.0 | 75.4 | 75.5 | 75.6 | 75.7 | 75.6 | 75.5 | 75.5 | 75.5 | 75.5 | 75.5 | 75.4 | 75.3 | 75.3 |
| | | Sparse | **25.9** | 22.4 | 23.4 | 23.9 | 24.2 | 24.3 | 24.4 | 24.5 | 24.6 | 24.6 | 24.7 | 24.7 | 24.7 | 24.6 | 24.7 |
| | | Medium | **88.5** | 86.5 | 86.7 | 86.7 | 86.7 | 86.7 | 86.6 | 86.5 | 86.5 | 86.5 | 86.4 | 86.3 | 86.3 | 86.3 | 86.2 |
| | | Dense | **99.5** | 99.2 | 99.2 | 99.2 | 99.2 | 99.2 | 98.9 | 98.9 | 98.8 | 98.8 | 98.7 | 98.7 | 98.5 | 98.4 | 98.5 |
| | Fast | All | **82.6** | 81.2 | 81.2 | 81.0 | 80.9 | 80.8 | 80.7 | 80.7 | 80.6 | 80.6 | 80.5 | 80.5 | 80.4 | 80.3 | 80.2 |
| | | Sparse | **33.1** | 30.7 | 31.2 | 31.2 | 31.1 | 31.1 | 30.9 | 30.9 | 30.8 | 30.7 | 30.5 | 30.4 | 30.2 | 29.9 | 29.5 |
| | | Medium | **92.9** | 91.6 | 91.5 | 91.4 | 91.3 | 91.1 | 91.0 | 90.9 | 90.9 | 90.8 | 90.8 | 90.9 | 90.7 | 90.6 | 90.6 |
| | | Dense | 97.5 | **97.6** | 97.2 | 97.2 | 97.1 | 96.8 | 97.0 | 96.9 | 96.8 | 96.7 | 96.9 | 97.0 | 96.8 | 96.8 | 96.7 |

Table 3.9: AP performance breakdown by speed and point cloud density. The best performance is highlighted in bold.

Finally, the current implementation of the $\eta$ function has two major limitations. First, we only explored speed and point cloud density as two primary factors without considering other potential factors related to the object, the environment, or the architecture itself. Second, since $\eta$ is constructed by evaluating performance for each object type across various input configurations, for object types with insufficient training samples (e.g., high-speed cyclists), $\eta$ may suffer from large performance variances, leading to inaccuracies. To address these issues, future work could explore a formulation of $\eta$ that does not require explicitly specifying trade-off factors or subdividing the dataset for evaluation. One promising approach could be to apply reinforcement learning to adaptively learn an aggregation policy.

## 3.6 Conclusion

We have addressed the inherent performance trade-off of fixed aggregation by proposing VADet, a variable aggregation approach that can be easily applied to different architectures with minimal modifications. Our extensive evaluation shows that VADet can effectively combat the trade-off and achieve SOTA performance. While VADet has its limitations in its current form, it underscores the need for more detailed performance assessments in 3D object detection (such as for different object classes, distances, velocities, motion patters, and scan point distributions) and highlights the potential improvements that could be achieved with LiDAR-only methods.

# Chapter 4

# MFSeg

## *Efficient Multi-frame 3D Semantic Segmentation*

## 4.1 Introduction

LiDAR-based 3D semantic segmentation is an essential task for 3D scene understanding in robotic applications such as autonomous driving. Accurate segmentation of LiDAR point clouds allows robotic systems to identify and classify objects in their environment, ensuring safe and reliable navigation.

LiDAR point clouds are typically captured sequentially, and utilizing this temporal nature has been shown to significantly enhance various 3D perception tasks. In the context of 3D semantic segmentation, recent work has explored feature-level aggregation across consecutive frames [202, 205, 254]. While these approaches have shown promise, they rely on structured representations of point clouds such as range image to align intermediate features across frames or are extensions of out-dated architectures. This limits their applicability to the current state-of-the-art (SOTA) methods in 3D perception such as point transformers [117, 266, 267, 311].

An alternative approach involves aggregating multi-frame point clouds at the input level via concatenation. This is widely adopted in 3D object detection as the multi-frame point clouds contain richer geometric information and can improve detection accuracy [20, 30, 88, 292]. In 3D semantic segmentation, we also find this approach to be beneficial. As shown in Figure 4.1, multi-frame point clouds provide a modest overall improvement (2.3%) in mean intersection-over-union (mIoU) performance, and a noticeably higher gain (up to 5.6%) in detecting vulnerable road users such as pedestrians and cyclists. This highlights the benefits of multi-frame point clouds for 3D semantic segmentation task in real-world applications.

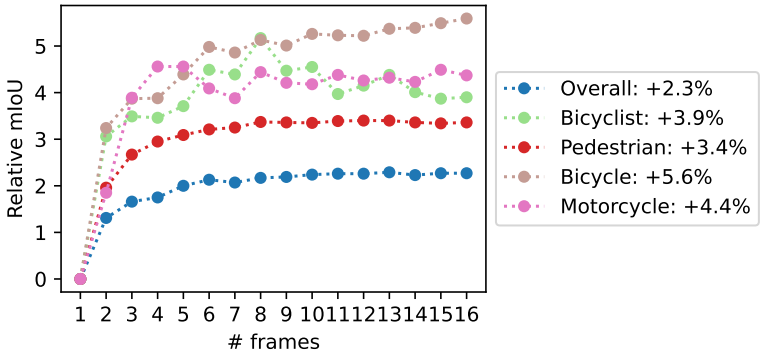



Figure 4.1: Relative mIoU improvement between various levels of multi-frame point cloud concatenation and single-frame input. Performance are evaluated using a SparseUNet [72] on the Waymo dataset [218].

Despite the benefits, multi-frame point clouds pose unique challenges for the SOTA point-based methods. Firstly, the increase in the number of points introduces computational challenges. For instance, a 64-beam LiDAR sensor can produce an average of 177K points per frame [218], and concatenating six frames can result in over one million points. Processing these dense point clouds requires significant computation, as illustrated by the latency of a SOTA method PTv3 [267] in Figure 4.2. Secondly, current methods follow the U-Net design and require upsampling point features in the decoder stage. This introduces unnecessary computational overhead for multi-frame point clouds, as predictions are typically only needed for points in the current frame.

In this chapter, we propose MFSeg, an efficient multi-frame 3D semantic segmentation framework. MFSeg aggregates point cloud sequences at the feature level while retaining the properties of input-level concatenation via a novel aggregation pipeline and regularization strategy. Unlike prior work, our method does not assume a structured feature representation and thus can be applied to SOTA point-based backbones such as point transformers.

To address the computational challenges, we employ a local feature extractor to reduce the input density, allowing MFSeg to efficiently process multi-frame inputs and features. Furthermore, we introduce a simple MLP-based point decoder to replace the traditional symmetrical decoder in the U-Net design, avoiding the unnecessary upsampling of point features.

These components enable MFSeg to achieve SOTA performance with low latency, as illustrated in Figure 4.2. Experiments on two large-scale datasets, nuScenes [20] and Waymo [218], demonstrate that MFSeg can consistently outperform existing methods.

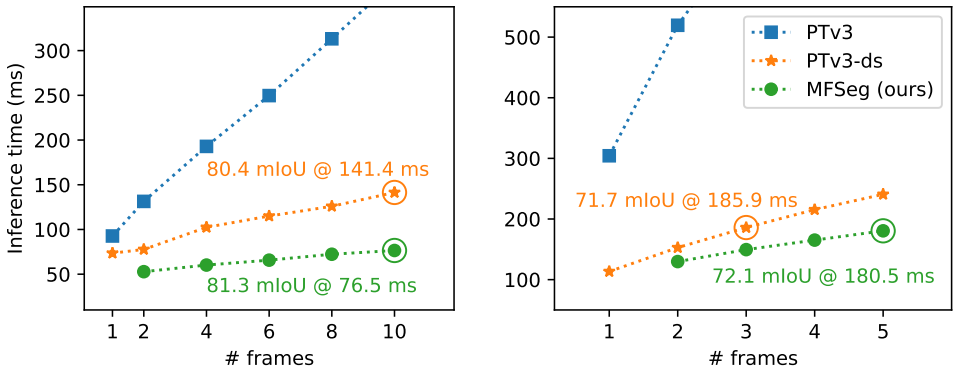



Figure 4.2: Inference time of PTv3 [267] and MFSeg under a range of point cloud sequence length on the nuScenes dataset [20] (left) and Waymo dataset [218] (right). PTv3-ds denotes that the point cloud is downsampled using a voxel-grid with size $0.05\,\mathrm{m}^3$, while PTv3 is evaluated with minimum input downsampling.

## 4.2 Method

MFSeg consists of four key components: (i) a local feature extractor (LFE) for capturing local point cloud features, (ii) a novel feature aggregator that fuses features from multiple frames, (iii) an off-the-shelf backbone encoder for processing the aggregated features, and (iv) an efficient point decoder that generates per-point predictions directly from the encoder output. An overview of the MFSeg architecture is provided in Figure 4.3. In the following sections, we describe each component in detail.

### 4.2.1 Local Feature Extraction

Directly processing dense LiDAR point clouds or aggregated features from multiple frames using a point-based backbone encoder can require substantial computational resources. As illustrated in Figure 4.2, the PTv3's runtime efficiency deteriorates with increasing input size, even when using voxel-grid downsampling.

To address this challenge and facilitate the use of additional frames, we employ a *local feature extractor* (LFE) to capture point cloud features within a local neighbourhood. It reduces redundancies while preserving relevant local geometric information that will be used by the feature aggregator later.

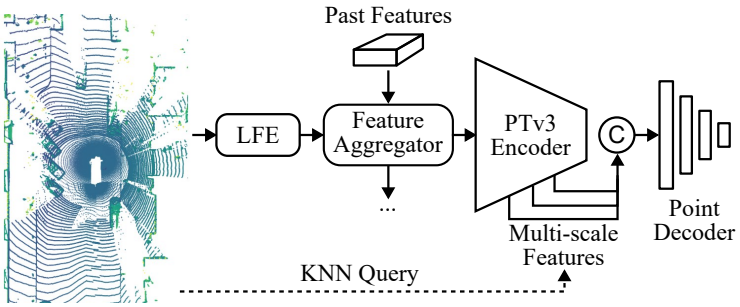



Figure 4.3: An overview of the proposed MFSeg architecture.

To efficiently define the local neighborhood, we first voxelize the point cloud and assign each point to a voxel in the grid. An off-the-shelf voxel feature extractors (VFEs) commonly used in 3D object detection [280] is employed to compute feature vectors for each voxel based on the points contained within it.

Although voxelization is applied, our method does not impose any structural assumptions on the resulting features like voxel grids. Instead, each voxel is treated as a new point, represented by a coordinate and an associated feature vector. Thus the resulting features form a new point cloud corresponding to the set of voxel features.

### 4.2.2 Feature Aggregation

The *feature aggregator* in MFSeg is designed to combine the LFE features from past frames with the current frame features. In order to retain the benefits of input concatenation as previously demonstrated in Figure 4.1, our feature aggregator aims to mimic the behavior of point cloud concatenation but in the feature space, allowing us to aggregate features in a way that maintains the spatial relationships between point clouds. To achieve this, our core idea is to formulate the feature space as a commutative monoid and treat the aggregator as a homomorphism mapping, thereby ensuring that the aggregation operation behaves similarly to point cloud concatenation.

Formally, let $P$ be the set of point clouds and $\oplus : P \times P \to P$ the concatenation operation. Since $\oplus$ is associative and commutative, and the empty point cloud $e \in P$ is an identity element, $(P, \oplus, e)$ thus forms a *commutative monoid*. We further denote the LFE defined in the previous section as a function $f : P \to F$ that maps point clouds to a feature space $F$.

We first formulate $(F, \odot, \mathbf{0})$, the feature space $F$ equipped with the binary aggregator $\odot$ and the zero vector $\mathbf{0}$, to be a commutative monoid. Given features $x, y, z \in F$, the aggregator $\odot$

should satisfy the following properties:

$$x \odot y = y \odot x, \qquad \text{(Commutativity)} \tag{4.1}$$
$$(x \odot y) \odot z = x \odot (y \odot z), \qquad \text{(Associativity)} \tag{4.2}$$
$$x \odot \mathbf{0} = x. \qquad \text{(Identity)} \tag{4.3}$$

Since $(P, \oplus, e)$ and $(F, \odot, 0)$ are now both commutative monoids, we can now treat the LFE $f$ as a *homomorphism* between them. Specifically, for point clouds $u, v \in P$,

$$f(u \oplus v) = f(u) \odot f(v), \tag{4.4}$$
$$f(e) = \mathbf{0}. \tag{4.5}$$

In practical applications, the point cloud sequences are typically aggregated in the order that they are captured by the LiDAR sensor. While this suggests that $\odot$ may not need to be commutative or associative, we find these properties to be beneficial to the model's performance, as will be shown in the ablation study (Section 4.4.3).

With the algebraic structure of the feature space defined, we next focus on how features from different frames are associated and fused during the aggregation process, and how the aforementioned properties can be enforced through architectural design and auxiliary losses.

### Feature Association

Recall from Section 4.2.1, the LFE produces a set of feature vectors for each point cloud, where each feature vector is computed using the points in the corresponding voxel. To combine a set of features from previous frames $X_{\tau-1} \subseteq F$ with the current frame features $X_\tau \subseteq F$, we first need to associate the two set of feature vectors before applying $\odot$ to each associated pair.

As depicted in Figure 4.4, given features $X_{\tau-1}$ and corresponding coordinates $C_{\tau-1} \subseteq \mathbb{R}^3$ from the previous frames, we first map the feature coordinates from the previous coordinate system $T_{\tau-1} \in \mathbb{SE}(3)$ into the current coordinate system $T_\tau$:

$$C^*_{\tau-1} = \left\{ T_\tau^{-1} T_{\tau-1} \cdot c \mid c \in C_{\tau-1} \right\}. \tag{4.6}$$

Afterward, feature vectors from the current frame $X_\tau$ and the past frames $X_{\tau-1}$ are assigned to voxels based on the current coordinates $C_\tau$ and the transformed coordinates $C^*_{\tau-1}$ respectively. We ensure that at most one feature vector from each set is assigned to the same voxel by using the same voxel size as VFE. The feature vectors belonging to the same voxel are associated and will be fused using $\odot$ in the following step.

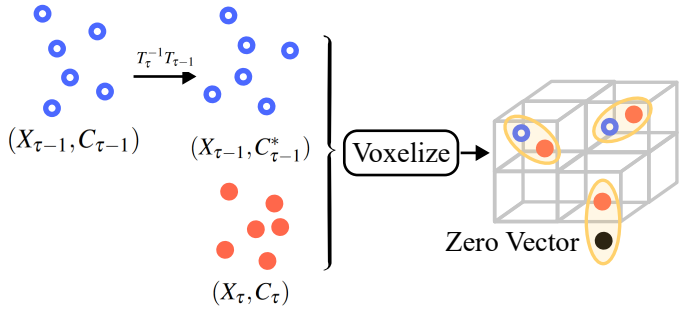



Figure 4.4: The feature association process (Section 4.2.2). Previous frame LFE features first undergo ego-motion correction, then are jointly voxelized with the current frame features to establish association. Features that cannot be associated are paired with the zero vector $\mathbf{0}$.

Since the feature coordinates are generally not aligned with the voxel grid, we further append the offset from the voxel center to each feature vector to preserve fine-grained spatial information. For feature vectors that cannot find an association after voxelization, we set their associated vectors to be the zero vector $\mathbf{0}$, indicating that the point cloud in that region is empty $e$. Note that this trivially satisfies Equation (4.5).

### Feature Aggregator

After the association step, the feature aggregator $\odot$ is applied to each pair of the associated feature vectors. We formulate the aggregator $\odot$ as follows:

$$x \odot y = \begin{cases} x + y, & \text{if } x = \mathbf{0} \text{ or } y = \mathbf{0}, \\ g\left(\frac{h(x,y)+h(y,x)}{2}\right), & \text{otherwise.} \end{cases} \tag{4.7}$$

$h(x, y) : F \times F \to F$ is a binary operator that merges the associated feature vectors $x$ and $y$ via concatenation followed by an FCN; $g : F \to F$ is another FCN.

The first case in Equation (4.7) handles the situation where a feature vector cannot be associated, satisfying the identity property (Equation (4.3)). In the general case, we apply symmetrization to $h$ followed by another FCN $g$ to enforce commutativity (Equation (4.1)).

### Auxiliary Loss

As the associativity (Equation (4.2)) and the homomorphism (Equation (4.4)) properties cannot be strongly enforced via architectural design, we use an auxiliary loss to approximate them.

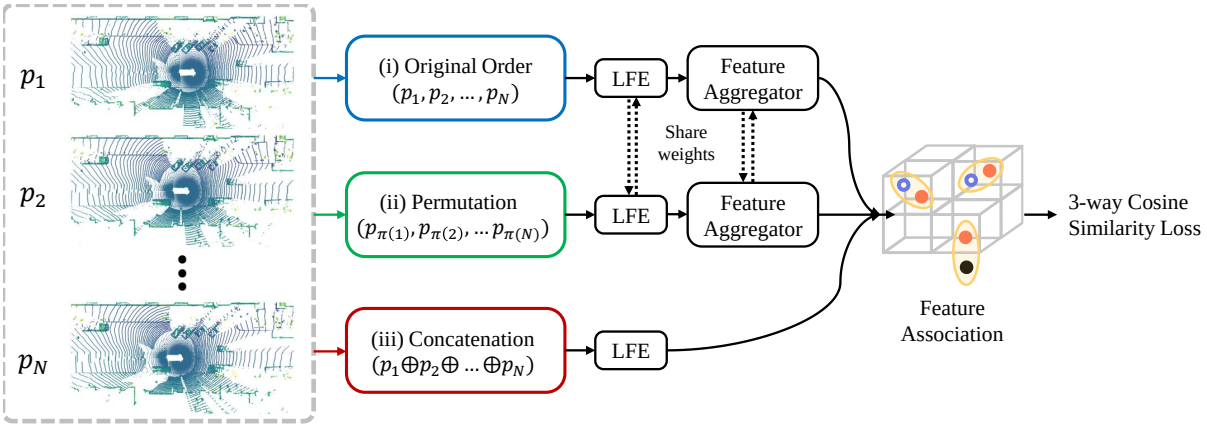



Figure 4.5: The proposed training scheme and auxiliary loss for the LFE and the feature aggregator.

Given point clouds $p_1, p_2, p_3 \in P$, we can rewrite Equations (4.2) and (4.4) as the following:

$$\begin{aligned} f(p_1 \oplus p_2 \oplus p_3) &= \big(f(p_1) \odot f(p_2)\big) \odot f(p_3) \\ &= \big(f(p_{\pi(1)}) \odot f(p_{\pi(2)})\big) \odot f(p_{\pi(3)}), \end{aligned} \tag{4.8}$$

where $\pi$ is an arbitrary permutation. To represent the three terms in Equation (4.8), we process the point cloud sequences with three branches during training, as depicted in Figure 4.5: (i) the sequence is aggregated in its original order, (ii) the sequence is aggregated under some random permutation, and (iii) the sequence is first concatenated at the input level, then processed by the LFE. The three sets of feature vectors are then associated with each other using the same technique as described in Section 4.2.2. For each associated triplet, we compute a three-way cosine similarity loss.

This auxiliary loss ensures that the LFE $f$ and feature aggregator $\odot$ approximate the desired algebraic properties, facilitating the network to learn a more robust and consistent feature aggregation process.

### 4.2.3 Point Decoder

After applying an off-the-shelf backbone to the aggregated features, MFSeg employs an MLP-based *point decoder* to generate per-point predictions. Unlike the U-Net-style decoders used in existing methods, our approach avoids upsampling the encoder features. This design serves two purposes: first, we aim to prevent unnecessary computation of features for points from previous

frames. Second, since the LFE operates on a neighborhood of points and the aggregation occurs at the feature level, symmetrical upsampling back to the original input resolution is not feasible.

Instead, we propose a simple and efficient point decoder that avoids the upsampling process by directly using the encoder's output. As illustrated in Figure 4.3, for each point where a prediction is required, we query the encoder features at multiple scales using K-nearest neighbors (KNN) and append the relative offsets from the feature vector's coordinates. The multi-scale features and offsets are then concatenated and processed by an FCN to generate the final segmentation prediction.

## 4.3 Experimental Setup

### 4.3.1 Datasets

To evaluate the effectiveness and efficiency of MFSeg, we perform experiments on two large scale semantic segmentation benchmarks: nuScenes [20] and Waymo [218].

The nuScenes dataset is collected in the Boston and Singapore areas using a 32-beam LiDAR. It consists of 1000 sequences with 850 in the training and validation splits, and 150 reserved for testing. Due to the relatively sparse nature of the point clouds in this dataset, with an average of 37k points per frame, 3D perception tasks can often benefit from input concatenation across longer sequences.

The Waymo dataset is collected in the San Francisco, Mountain View, and Phoenix regions using a more complex 5-sensor setup, with a primary 64-beam LiDAR. Consequently, the point clouds in Waymo are much denser, averaging 177k points per frame. This density poses a significantly greater computational challenge compared to nuScenes, particularly with multi-frame inputs.

### 4.3.2 Implementation Details

Our implementation is built on the official codebase of PTv3 [1], utilizing the PTv3's encoder as our backbone. The LFE module consists of four linear layers with channel dimensions of $[128, 128, 128, 128]$. The feature aggregator is composed of two FCNs, $h$ and $g$, as described in Section 4.2.2. Both $h$ and $g$ are three-layer FCNs with a hidden channel dimension of 128. Following a common practice in input concatenation, during feature aggregation, we append

[1] https://github.com/Pointcept/Pointcept

timestamps to the feature vectors to help the model distinguish current and past frame features. Finally, the point decoder is an FCN with hidden channel sizes $[1024, 512, 256, 128, 64]$. For nuScenes models, we aggregate a maximum of 10 frames, and for Waymo models, we aggregate a maximum of 5 frames.

We employ a two-stage training strategy. In the first stage, we train the model with input concatenation, bypassing the feature aggregator (branch (iii) in Figure 4.5). This helps initialize the LFE and the other network components. For this stage, we adopt the same training parameters and data augmentations used by PTv3. For both the nuScenes and Waymo datasets, the model is trained for 50 epochs following a one-cycle learning rate schedule with a maximum learning rate of 0.002. In the second stage, we fine-tune the model using the auxiliary loss scheme detailed in Section 4.2.2. The model is trained for an additional 20 epochs with a maximum learning rate of 0.001.

## 4.4 Results

### 4.4.1 Overall Performance

The results on the nuScenes dataset and the Waymo dataset are presented in Table 4.1 and Table 4.2 respectively. To provide a fair comparison with methods that utilize test-time augmentation (TTA), we report results both with and without TTA, although TTA is not practical for real-time applications.

| Method | mIoU | Barrier | Bicycle | Bus | Car | Const. veh. | Motorcycle | Pedestrian | Traffic cone | Trailer | Truck | Drivable | Other flat | Sidewalk | Terrain | Manmade | Vegetation |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cylinder3D [321] | 76.1 | 76.4 | 40.3 | 91.2 | 93.8 | 51.3 | 78.0 | 78.9 | 64.9 | 62.1 | 84.4 | 96.8 | 71.6 | 76.4 | 75.4 | 90.5 | 87.4 |
| PVKD [86] | 76.0 | 76.2 | 40.0 | 90.2 | 94.0 | 50.9 | 77.4 | 78.8 | 64.7 | 62.0 | 84.1 | 96.6 | 71.4 | 76.4 | 76.3 | 90.3 | 86.9 |
| RPVNet [274] | 77.6 | 78.2 | 43.4 | 92.7 | 93.2 | 49.0 | 85.7 | 80.5 | 66.0 | 66.9 | 84.0 | 96.9 | 73.5 | 75.9 | 76.0 | 90.6 | 88.9 |
| SphereFormer* [117] | 79.5 | 78.7 | 46.7 | 95.2 | 93.7 | 54.0 | 88.9 | 81.1 | 68.0 | 74.2 | 86.2 | **97.2** | 74.3 | 76.3 | 75.8 | 91.4 | 89.7 |
| PTv3* [267] | 80.4 | **80.5** | 53.9 | **96.0** | 91.9 | 52.3 | 89.0 | 84.4 | 71.7 | 74.2 | 84.5 | **97.2** | **75.6** | 76.9 | 76.2 | 91.2 | 89.6 |
| PTv3 (10-frame)† | 81.3 | 79.3 | 61.0 | 95.9 | **95.0** | 50.7 | **93.0** | **87.3** | **74.5** | 68.8 | **89.2** | 97.0 | 74.1 | 76.5 | 76.3 | **91.6** | **90.4** |
| MFSeg (ours) | 81.3 | 78.7 | 59.7 | 94.1 | 91.7 | 65.1 | 90.3 | 85.2 | 71.2 | 73.8 | 85.6 | 96.9 | 74.4 | 76.5 | 76.5 | 91.2 | 89.8 |
| w/ TTA* | **82.5** | 79.4 | **62.1** | 94.7 | 92.5 | **67.1** | 92.2 | 86.7 | 73.4 | **75.1** | 86.5 | 97.1 | **75.6** | **77.8** | **77.5** | **91.6** | **90.4** |

* indicates the method uses test-time augmentation (TTA). † results are reproduced by us using the official codebase.

Table 4.1: Results on the nuScenes validation split.



| Method | mIoU | Car | Truck | Bus | Other veh. | Motorcyclist | Bicyclist | Pedestrian | Sign | Traffic light | Pole | Con. cone | Bicycle | Motorcycle | Building | Vegetation | Tree trunk | Curb | Road | Lane marker | Other gro. | Walkable | Sidewalk |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SparseUNet [72] | 66.6 | 94.4 | 59.8 | 85.1 | 37.8 | 2.2 | 69.1 | 89.3 | 73.4 | 40.4 | 74.8 | 57.3 | 66.6 | 75.2 | 95.5 | 91.3 | 67.0 | 68.1 | 92.3 | 41.7 | 30.1 | 79.0 | 75.6 |
| SphereFormer [117] | 69.9 | 94.5 | 61.6 | 87.7 | 40.2 | 0.9 | 69.7 | 90.2 | 73.9 | **41.8** | 77.2 | 65.4 | 71.9 | 83.7 | 95.9 | 91.7 | 68.4 | 69.8 | 93.3 | 53.9 | 47.9 | 80.8 | 77.2 |
| PTv3* [267] | 71.2 | **94.8** | **63.6** | **89.0** | 34.7 | **12.2** | 73.2 | 90.6 | 71.6 | 31.3 | 77.8 | **75.9** | 75.6 | 87.3 | 96.0 | 91.8 | 68.1 | 71.2 | 93.5 | 56.6 | 51.3 | 81.6 | 78.4 |
| MFSeg (ours) | 72.1 | 94.5 | 62.2 | 86.1 | 43.7 | 8.6 | 79.0 | 91.1 | 75.1 | 33.5 | 80.4 | 72.2 | 78.3 | 90.0 | 96.1 | 92.0 | 70.0 | 72.2 | 93.4 | 56.9 | 51.2 | 81.2 | 78.0 |
| w/ TTA* | **72.7** | 94.6 | 63.0 | 86.2 | **44.1** | 4.5 | **81.4** | **91.6** | **76.0** | 34.9 | **81.4** | 73.9 | **79.4** | **92.3** | **96.3** | **92.2** | **70.7** | **73.1** | **93.6** | **57.6** | **52.0** | **81.7** | **78.8** |

* indicates the method uses test-time augmentation (TTA).

Table 4.2: Results on the Waymo validation split.

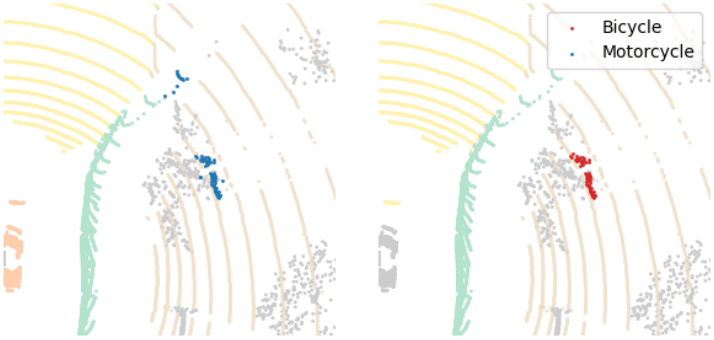



Figure 4.6: Segmentation results from PTv3 (left) and MFSeg (right). PTv3 misclassifies a partially occluded bicycle along with nearby background points as motorcycle (blue), while MFSeg accurately identifies the points belonging to the bicycle (red).

MFSeg outperforms competing single-frame methods across both datasets, even without the use of TTA. On the nuScenes dataset, MFSeg achieves a mIoU of 81.3% without TTA and 82.5% with TTA, surpassing the performance of the SOTA method PTv3 with single-frame input, while matching its performance with 10-frame concatenated point clouds. Noteably, compared to single-frame methods, MFSeg improves the performance of vulnerable road users, including pedestrian, bicycle and motorcycle, which are crucial in practical applications. A qualitative example is shown in Figure 4.6.

On the Waymo dataset, MFSeg also outperforms single-frame SOTA PTv3 in terms of mIoU, achieving 72.1% without TTA and 72.7% with TTA. Similarly, we observe substantial improvements in the performance of bicycle, bicyclist, and motorcycle, with a moderate improvement in pedestrian.

Our results demonstrate that MFSeg can effectively utilize sequential point cloud data, providing benefits similar to point cloud concatenation (Figure 4.1) without the associated computational cost. Moreover, MFSeg's ability to perform well without relying on TTA highlights its potential for real-time deployment.

### 4.4.2 Inference Latency

To demonstrate that MFSeg can process multi-frame point clouds more efficiently than existing point-based methods with concatenated point clouds, we compare its inference time with SOTA method PTv3 across a range of input sequence lengths.

Although PTv3 is a point-based method, in practice, its input point cloud is downsampled

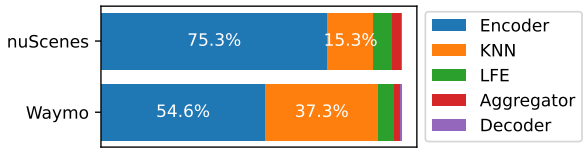



Figure 4.7: A breakdown of the latency of each component in MFSeg.

using a voxel-grid. To highlight the effect of input complexity on the latency of point-based methods without downsampling, we evaluate PTv3 using a small voxel size of 0.01 $\mathrm{m}^3$ to minimize the downsampling effects. This setup demonstrates the computational requirements of processing dense point clouds without simplification. For comparison, we refer to the version of PTv3 that employs voxel-grid downsampling as PTv3-ds.

We measure the latency on the validation split using a single NVIDIA A6000 GPU with minimum additional system load. Data loading time is excluded from the measurement. The results are shown in Figure 4.2. In both datasets, we observe that PTv3, when processing concatenated point clouds with minimal downsampling, scales poorly as the number of frames increases, making it unsuitable for online settings. While PTv3-ds has a much lower latency thanks to input downsampling, it is unable to produce predictions for every point in the point cloud without label interpolation or multiple forward passes. In contrast, MFSeg achieves consistently lower latency and scales more efficiently as the number of frames increases, indicated by the gentler slope, while delivering better performance compared to input concatenation.

Figure 4.7 shows a breakdown of the latency of each component of MFSeg. In both datasets, we observe that the proposed LFE layer and point decoder introduce minimum computational overhead. However, as the input density increases, particularly in the Waymo dataset, the cost of the KNN operation rises significantly. Efficiently extracting multi-scale features from the unstructured features of a point-based backbone remains a computational challenge, and we believe future work should focus on addressing this issue.

### 4.4.3 Ablation Study

We conduct an ablation study on the nuScenes dataset to illustrate the impact of each proposed component. The results are presented in Table 4.3.

First, in row A and B, we investigate the effects of LFE and the point decoder in the single-frame setting. Note that row A represents a baseline PTv3 without our proposed modifications, and the feature aggregator is bypassed in this setup. The results show that both models achieve

| | LFE | Point Decoder | Input Concatenation | Feature Aggregation | Associativity Loss | Homomorphism Loss | mIoU |
|---|---|---|---|---|---|---|---|
| A | - | - | - | - | - | - | 78.6 |
| B | ✓ | ✓ | - | - | - | - | 79.2 |
| C | ✓ | ✓ | ✓ | - | - | - | 81.7 |
| D | ✓ | ✓ | - | ✓ | - | - | 79.8 |
| E | ✓ | ✓ | - | ✓ | ✓ | - | 80.6 |
| F | ✓ | ✓ | - | ✓ | ✓ | ✓ | 81.3 |

Table 4.3: Ablation study on the NuScenes validation split without TTA.

comparable performance, indicating that the LFE and point decoder maintain the SOTA performance of PTv3, despite being simpler in design.

Next, we explore the impact of input concatenation and feature-level aggregation in multi-frame settings. Comparing row C with rows A and B, we observe that multi-frame point clouds can enhance segmentation performance when using a point-based backbone. However, in row D, where feature aggregation is applied without regularization, the model can no longer achieve the full benefits of input concatenation.

By introducing auxiliary losses to enforce the desired algebraic properties in the feature space—-specifically associativity (row E) and homomorphism (row F)—-we observe that our method matches the performance of input concatenation (row C). In row E, we disable the third branch depicted in Figure 4.5, and the feature association and cosine similarity is only computed between the original sequence and permuted sequence, thus only enforcing Equation (4.2). Row F represents our full approach with all the proposed components. These results validate the effectiveness of our proposed method.

## 4.5 Limitations and Future Work

The core idea of MFSeg is to enforce the feature space to mimic the behavior of input-level aggregation. While the motivating example in Figure 4.1 demonstrated the benefits of point cloud aggregation for 3D semantic segmentation, it remains unclear whether this is the optimal approach. In other words, our method might be attempting to replicate the behavior of a sub-optimal strategy for incorporating temporal information, which could fundamentally limit the model's performance. To better understand this, it is necessary to perform a more in-depth analysis of the performance characteristics of 3D semantic segmentation models using different input configurations.

Additionally, this chapter presents experiments with feature aggregation only at the input layer level. It is therefore not clear whether applying feature aggregation to deeper layers, or multiple layers, would be more or less effective. On the one hand, introducing supervision at multiple scales may help the model converge more quickly to the objective defined in Equation (4.8). On the other hand, higher-level features—often residing in much higher-dimensional spaces and encoding contextual information—might not be as easily aggregated in analogy to point cloud concatenation as the early layer features. Further experiments are needed to address these questions.

Related to the previous point, since the aggregator is applied at the lower levels, the aggregated features may lack the contextual information present in the deeper layers of the network. To fully take advantage of the global context from later stages, modifications to the current architecture may be necessary to integrate both local and global features more effectively.

In terms of computational efficiency, MFSeg still has significant room for improvement. During training, one of the proposed branches requires processing dense concatenated point cloud sequences to guide the feature aggregator. While no decoding or prediction occurs in this branch, the initial LFE can still consume a substantial amount of memory due to the large number of points. As a result, while MFSeg scales better at test time, training under this configuration remains challenging. Further optimizations or surrogate losses that approximate Equation (4.8) could eliminate the need for dense processing during training, improving training efficiency.

At test time, although the design of the point decoder is simple and incurs minimal computational cost (Figure 4.7), the kNN operation used to gather multi-scale features from the backbone encoder is computationally expensive and can become a bottleneck in real-time applications. To further improve efficiency, replacing kNN with a voxel-based association technique, such as the one discussed in Section 4.2.2, could improve the scalability of MFSeg.

## 4.6 Conclusion

In this chapter, we introduced MFSeg, an efficient framework for multi-frame 3D semantic segmentation. By aggregating point cloud sequences at the feature level and regularizing the feature extraction and aggregation process using a novel formulation, MFSeg is able to reduce computational overhead while maintaining high accuracy. Experiments on the nuScenes and Waymo datasets demonstrate that MFSeg outperforms existing methods while achieving lower inference latency than the comparable point-based method in the multi-frame setting.

# Chapter 5

# SOAP

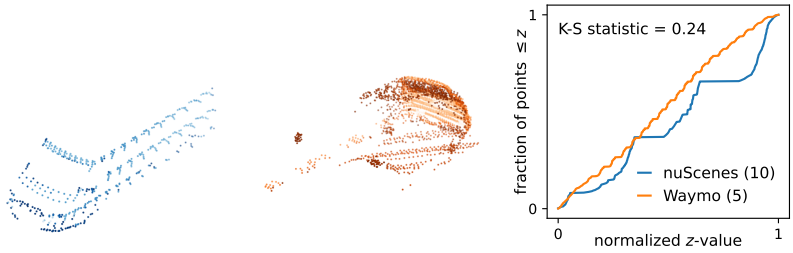
## *Cross-sensor Domain Adaptation for 3D Object Detection Using Stationary Object Aggregation Pseudo-labelling*

## 5.1 Introduction

LiDAR sensors are commonly used in autonomous driving and other safety-critical robotic applications to provide accurate 3D localization of objects. State-of-the-art (SOTA) LiDAR-based object detectors currently use deep neural networks trained via supervised learning, requiring a large amount of realistic data labelled by human annotators. Annotation is expensive, motivating the re-use of existing labelled datasets, but these typically have a limited *domain*, e.g., a specific sensor configuration and relatively few geographic locations and weather conditions. It is well known that detectors trained with a dataset from one domain experience a significant degradation of performance when fed data from a different domain.

In this chapter, we tackle the cross-sensor domain adaptation problem that arises whenever a detector is required to interpret data from a sensor different to that on which it was trained. Specifically, given a detector trained on labelled point clouds collected using one sensor (the *source domain*), we aim to improve the detector's performance on point clouds collected using a different sensor (the *target domain*), either with no labels (unsupervised) or with only a small number of labels from the target domain (semi-supervised). This situation arises commonly when a LiDAR sensor is updated or a fleet of autonomous vehicles uses multiple sensors.

Cross-sensor domain adaptation can present a formidable challenge because similar objects scanned by different LiDAR sensors may have very different scan patterns, even after the widely-




(a) nuScenes sparse (b) Waymo sparse (c) CDFs for sparse




(d) nuScenes dense (e) Waymo dense (f) CDFs for dense

Figure 5.1: Scan lines are evident in point clouds when only few input frames are used (a,b), appearing as obvious modes in CDF plots that largely differ because of the modes (c). Aggregating many more frames removes visible scan lines (d,e) and makes CDFs for similar objects in different datasets more alike (f)

adopted few-frame input aggregation. Figures 5.1a and 5.1b show point clouds of vehicles at similar distances from their respective sensors, created by aggregating 10 frames from nuScenes [20] and 5 frames from Waymo [218] datasets, respectively. There are evident scan lines and obvious visual dissimilarities between the point clouds. There are also significant differences in the corresponding cumulative density function (CDF) plots of the $z$-component of point positions (Figure 5.1c).

Although SOTA 3D object detectors perform well within a given domain, their performance significantly drops when applied to cross-sensor point clouds. This is demonstrated in Table 5.1, where we compare the performance of a VoxelNeXt [33] model trained and evaluated on the nuScenes [20] and Waymo [218] datasets.

We attribute much of this performance degradation to the different scan patterns mentioned above. This view is supported by a recent study based on simulation [64] that suggests the difference in scan patterns alone can have a substantial impact on the performance of existing object detectors. At the same time, we observe that aggregating many more frames tends to

| | # frames | AP | Δ |
|---|---|---|---|
| nuScenes → Waymo | 1 | 7.0 | **–70.5** |
| | 5 | 20.4 | **–57.1** |
| Waymo in-domain | 5 | 77.5 | |
| Waymo → nuScenes | 1 | 42.7 | **–40.8** |
| | 10 | 49.0 | **–34.5** |
| nuScenes in-domain | 10 | 83.5 | |

Table 5.1: Degradation (Δ) w.r.t. in-domain performance of SOTA detector VoxelNeXt [33] in Waymo ↔ nuScenes cross-sensor setting. Increasing the number of aggregated frames improves over single-frame input, but there is still a substantial performance gap.

reduce the scan patterns. This is illustrated in Figures 5.1d and 5.1e, which show dense point clouds created by aggregating 400 nuScenes and 200 Waymo frames, respectively. There are no visible scan lines and the multi-modality evident with few frames has disappeared from the CDF plots in Figure 5.1f.

SOTA methods in cross-sensor domain adaptation often employ pseudo-labelling, where a model trained on labelled data is used to generate labels for unlabelled data. As detailed in Section 2.7, various approaches have been proposed to improve pseudo-label quality and regularize training with pseudo-labels [257, 286, 288], but these approaches do not appear to explicitly address the important difference in scan-patterns between different domains.

To tackle this gap, we propose *Stationary Object Aggregation Pseudo-labelling* (SOAP) to improve cross-sensor pseudo-label accuracy by exploiting scene-level full-sequence aggregation of input point clouds to close the domain gap caused by scan patterns. SOAP is motivated by the facts that (i) aggregation improves the representation of sparsely-scanned objects [255], (ii) sensor-specific scan patterns are reduced by full-sequence aggregation (Figure 5.1), and (iii) stationary objects respond well to full-sequence aggregation and are a statistically important component of object detection: at least two thirds of cars are stationary at some point in sequences in major realistic driving datasets [20, 104, 218, 261].

Unlike previous pseudo-labeling methods, which aggregate point clouds at the object or track level, SOAP directly uses scene-level aggregated point clouds as input. This approach improves pseudo-label quality for stationary objects while preserving dynamic object labels. SOAP pseudo-labels can be employed to update detectors or bootstrapping annotations, making it compatible with many existing domain adaptation methods.

Extensive experiments using nuScenes and Waymo datasets show SOAP pseudo-labels can

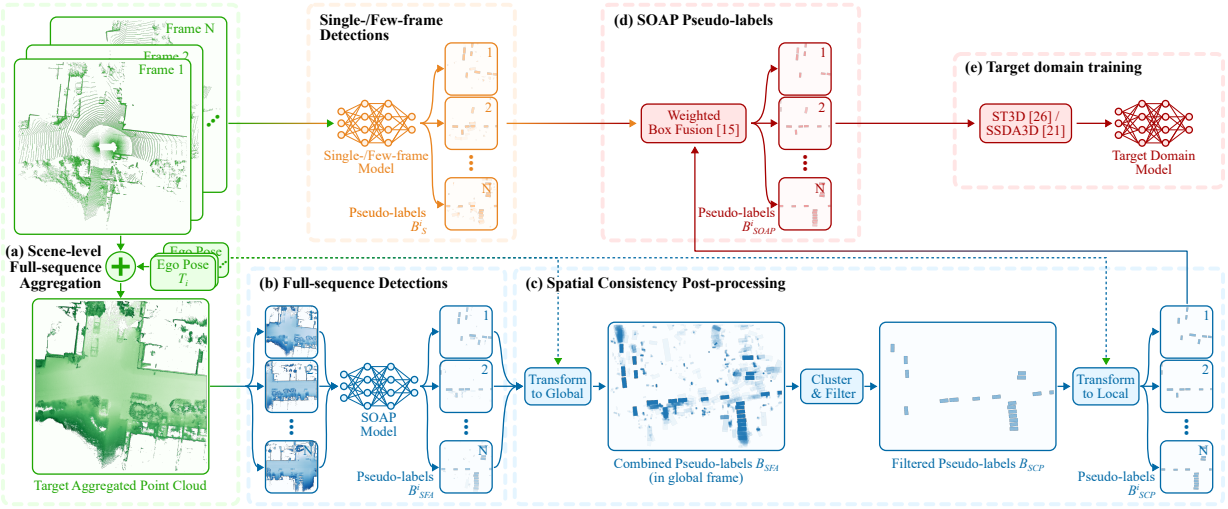



Figure 5.2: Overview of Stationary Object Aggregation Pseudo-labelling (SOAP) pipeline.

close a minimum of 30.3% overall domain gap compared to few-frame detectors without access to any target domain labels. SOAP also complements other SOTA domain adaptation methods, including ST3D [286] and SSDA3D [257], improving their already impressive results in both unsupervised and semi-supervised settings. In nuScenes $\rightarrow$ Waymo setting using CenterPoint [298], for example, using SOAP closes 42.6% domain gap compared to the 9.5% closed by ST3D. With only 1% target domain labels, SOAP closes 86.8% domain gap compared to the 81.4% closed by SSDA3D.

## 5.2 Method

The main components of SOAP are: (i) *Scene-level Full-sequence Aggregation* (SFA), which produces aggregated point clouds from the entire input sequence; (ii) *Quasi-Stationary Training* (QST) that is used to train a pseudo-labelling model to detect stationary objects; and (iii) *Spatial Consistency Post-processing* (SCP) that enhances pseudo-labels by exploiting the stationarity of the predictions. SOAP is used in combination with a pre-trained model to generate high quality pseudo-labels for stationary objects while retaining dynamic object pseudo-labels. An overview of the SOAP pipeline is shown in Figure 5.2.

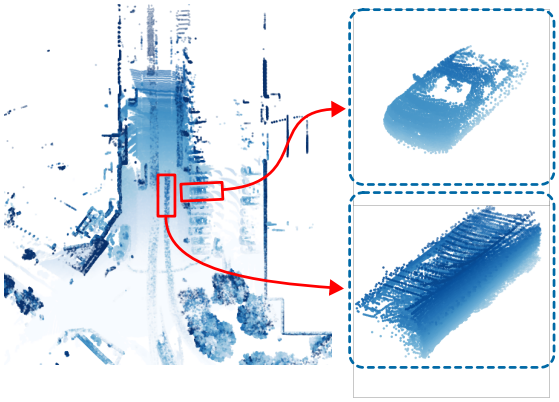


Figure 5.3: Example of a point cloud generated by SFA. Dynamic objects are distorted while stationary objects are densified.

### 5.2.1 Scene-level Full-sequence Aggregation

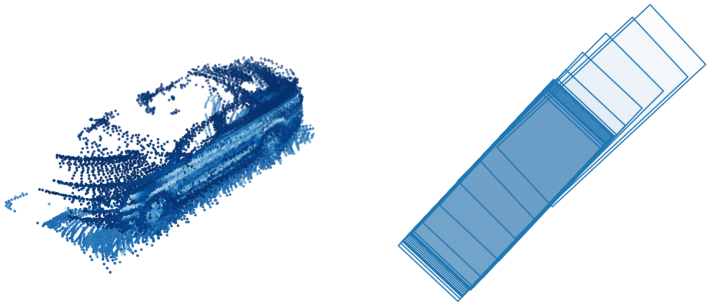
*Scene-level full-sequence aggregation* (SFA) involves projecting a sequence of point clouds to a global coordinate system, where the point clouds are concatenated into a single dense point cloud, as illustrated in Figure 5.2a. Formally, given a sequence of point clouds $P_1, P_2, \ldots, P_N$, where $P_i = \{p_i^1, p_i^2, \ldots, p_i^{M_i}\} \subset \mathbb{R}^3$, and corresponding sequence of $\mathbb{SE}(3)$ pose transformations $T_1, T_2, \ldots, T_N$, which transform the point clouds from the local LiDAR or vehicle coordinate system to a common global coordinate system, the point cloud aggregation process in the global coordinate system is defined by

$$P^* = \bigcup_{i=1,2,\ldots,N} \left\{T_i p_i^j\right\}_{j=1,2,\ldots,M_i}. \tag{5.1}$$

During training, for a given frame $i$, the aggregated point cloud $P^*$ is transformed back to the local coordinate system using the inverse pose transform $T_i^{-1}$.

Scene-level aggregation of a few frames in a short time window has been shown to be effective at producing denser input point clouds and consequently better detection performance [20]. On the other hand, due to the diminishing returns [20] and performance degradation [30, 292] observed for scene-level aggregation with large temporal windows, full-sequence aggregation has only been attempted at object- and track-level [61, 159, 187, 284].

Compared to few-frame aggregation, SFA increases point density and provides richer geometric information for stationary objects. In addition, unlike object- and track-level aggregation used by prior work, SFA does not rely on object annotations or initial predictions produced by



(a) Aggregated point cloud (b) Object trajectory (BEV)

Figure 5.4: Example of a quasi-stationary object. This object reached a maximum speed of 1.4 m/s with a total displacement of 3.9 m, and thus would be eliminated by naive filtering.

single- or few-frame detectors, which we show are unreliable in the cross-sensor setting Table 5.1. SFA is therefore very suitable for its proposed application, since $P^*$ can be obtained for both the source and (unlabelled) target domains.

SFA tends to distort dynamic object point clouds, due to the motion of the objects not being corrected during aggregation. This is depicted in Figure 5.3. By contrast, stationary objects are densified with a more complete and accurate geometry compared to single- or few-frame point clouds. As noted above and in Figure 5.1, the aggregation process also weakens the LiDAR-specific scan patterns.

### 5.2.2 Quasi-Stationary Training

Although annotations are available for the source domain, training a model to detect stationary objects from aggregated point clouds is not straightforward. A naive approach is by filtering the ground truth annotations during training based on speed estimates or the overall displacement of the object. However, since the speed of an object can change over time, especially for sequences that span a large time window, the object can be near stationary during part of the sequence and moving in other parts. If the majority of the observed points come from the part of the sequence where the object is near stationary, the object's aggregated point cloud will have little distortion, as if the object were stationary for the entirety of the sequence.

We refer to such objects as *quasi-stationary* objects. Figure 5.4 depicts an example of such an object, which would be excluded based on a naive speed or displacement criterion. Doing

so would result in undistorted objects remaining in the aggregated point clouds, but without a positive label, causing confusion and reducing model performance.

To avoid excluding these quasi-stationary objects, we formally define the notion of quasi-stationarity using a *quasi-stationary score* (QSS) that takes into account both the movement of the objects and how much each observation contributes to the final aggregated point clouds.

**Definition 1 (QSS)** *Let $\{b_1, b_2, \ldots, b_N\}$ be a set of $N$ bounding boxes of an object annotated in a sequence within a common coordinate system and $\mathrm{C}(b_i)$ be the number of points observed in the bounding box $b_i$. For a given bounding box observation $b_i$, $\mathrm{QSS}$ is defined as the average IoU between $b_i$ and other bounding boxes $b_j$, weighted by the fractions of points contributed by $b_j$:*

$$\mathrm{QSS}(b_i) = \sum_{j=1}^{N} \frac{\mathrm{C}(b_j)}{\sum_{k=1}^{N} \mathrm{C}(b_k)} \mathrm{IoU}(b_i, b_j) \tag{5.2}$$

Intuitively, the $\mathrm{QSS}$ can be interpreted as how likely it is that the point cloud for a given object is undistorted at the location of $b_i$. For example, if another observation $b_j$ has little overlap with $b_i$ (indicating object movement) but contains only a few points, then the final aggregated point cloud is not likely to be distorted by $b_j$. Alternatively, if $b_j$ has a large overlap with $b_i$ and also contains a large fraction of points, then the object is likely to undistorted at location $b_i$.

Finally, the most likely location $b^*$ of the object in the aggregated point clouds and the degree $s^*$ of the point cloud being free from distortion can be estimated as follows:

$$b^* = \arg\max_i \mathrm{QSS}(b_i) \tag{5.3}$$

$$s^* = \max_i \mathrm{QSS}(b_i) \tag{5.4}$$

We refer to $b^*$ as the quasi-stationary bounding box and $s^*$ as the corresponding QSS. Objects with a large QSS $s^* > \epsilon$ for some $\epsilon$ can be considered quasi-stationary. For instance, the object in Figure 5.4 has QSS $s^* = 0.91$.

**Out-of-sight quasi-stationary objects:** In the labelled source domain dataset, we notice objects not visible from the current frame are sometimes not labelled. Since SFA utilizes the entire sequence to create a dense representation of the scene, out-of-sight quasi-stationary objects will also be densified in the corresponding aggregated point cloud. This creates inconsistent labels where in a given frame, some dense objects have annotations while others do not. To address this problem, we construct spatially consistent training labels by projecting the quasi-stationary bounding boxes $b^*$ to *all* frames in the sequence, even if the object is not observed in some frames.

### 5.2.3 Spatial Consistency Post-processing

Since the SOAP model is tasked to detect only quasi-stationary objects from aggregated point clouds, a detected object should have consistent predictions in the global coordinate system across multiple frames. To utilize this stationarity property of the pseudo-labels, we propose *Spatial Consistency Post-processing* (SCP) to eliminate false positive predictions and recover false negative objects, improving pseudo-label quality.

As illustrated in Figure 5.2c, SCP is performed by obtaining per-frame predicted bounding boxes (denoted by $B^i_{SFA}$ for frame $i$) using the SOAP model and then gathering all predictions in the global coordinate system. Gathering all predictions requires transforming per-frame predictions (bounding boxes) using the corresponding ego pose transformation $T_i$:

$$B_{SFA} = \bigcup T_i B^i_{SFA} \tag{5.5}$$

The bounding boxes $B_{SFA}$ are clustered based on an IoU threshold $\mu$. To ensure the pseudo-labels are consistent across multiple frames, we eliminate the clusters with a number of detections fewer than threshold $\eta$ that depends on the frame rate of the dataset.

For the remaining clusters, the boxes in each cluster are combined into a single bounding box per cluster, similar to Weighted Boxes Fusion (WBF) [216]. Specifically, we use the heading $\theta$ of the most confident prediction in each cluster and average other attributes, including position $(x, y, z)$, size $(w, l, h)$ and velocity $(v_x, v_y)$, weighted by the confidence of each box. Finally, non-maximum suppression is applied to remove any overlapping predictions in the global coordinate system. The final bounding boxes in the global coordinate system are denoted by $B_{SCP}$.

To obtain the pseudo-labels $B^i_{SCP}$ for each frame $i$ in the sequence, we transform $B_{SCP}$ back to each frame's local coordinate system using the inverse pose transformation $T_i^{-1}$, defined as follows:

$$B^i_{SCP} = T_i^{-1} B_{SCP} \tag{5.6}$$

Since an object may be occluded in a frame and have very few or no points in the sparse point cloud, we remove any bounding boxes that contain no points, so that the pseudo-labels are reasonable with respect to the frame's sparse point cloud.

### 5.2.4 SOAP Pseudo-labels

In order to recover a complete set of pseudo-labels for both stationary and dynamic objects, SOAP utilizes the predictions from a pre-trained single- or few-frame detector capable of detecting dynamic objects. The SOAP and pre-trained models are calibrated separately with Beta

Calibration [114] using source domain data. Let $B_S^i$ be the bounding boxes predicted by the pre-trained detector for frame $i$, then the SOAP pseudo-labels, denoted by $B_{SOAP}^i$, are obtained by combining $B_S^i$ and $B_{SCP}^i$ using WBF [216].

Our results show that this simple approach can improve existing sparse pseudo-labels by a large margin. We leave it as future work to study more optimal ways of combining pseudo-labels or obtaining dynamic pseudo-labels directly from aggregated point clouds.

## 5.3 Experimental Setup

SOAP is evaluated in both unsupervised and semi-supervised domain adaptation settings. This section presents the experimental setup of our experiments.

### 5.3.1 Datasets

We evaluate SOAP using two large-scale autonomous driving datasets for 3D object detection: nuScenes [20] and Waymo [218]. In what follows, we use the syntax *source domain dataset* $\rightarrow$ *target domain dataset* to denote the training and testing setting, respectively.

The NuScenes dataset [20] contains 1,000 sequences of 20 seconds each, collected in Boston and Singapore. The vehicle is equipped with a single Velodyne HDL-32E 32-beam top-mounted rotating LiDAR operating at 20 Hz, yielding $\approx$400 point cloud scans per sequence, from which 40 keyframes are selected uniformly and annotated. Unless specified otherwise, all models trained on the nuScenes dataset use 10 sweeps (–0.5 s) as input.

The Waymo dataset [218] contains 1,150 sequences of 20 seconds each, collected in San Francisco, Mountain View, and Phoenix. The Waymo dataset uses a 5-sensor setup with a single proprietary 64-beam top-mounted rotating LiDAR operating at 10 Hz, and four side-mounted close-range LiDARs. All point clouds ($\approx$200) in each sequence are annotated with bounding boxes. Due to much higher annotation frequency compared to nuScenes, we use 20% uniformly sampled frames for training. Consistent with the nuScenes models, all models trained on the Waymo dataset use 5 sweeps (–0.5 s) as input.

In addition to the difference in the point cloud and annotation frequency, the nuScenes dataset annotates 23 classes with 8 attributes, 10 of which are used in the object detection task, whereas Waymo contains annotations for only Vehicle, Cyclist, and Pedestrian. Following previous work [155, 238, 239, 256, 286, 288, 296, 299], we select the common vehicle/car class for training and evaluation for all experiments.

### 5.3.2 Point Cloud Input

**Single- and few-frame input:** The nuScenes [20] and Waymo [218] datasets have different sensor ranges and point cloud features. In our experiments, we match the input point cloud format. Specifically, the input point cloud range is $[-75, 75]$ m for both $x$ and $y$ dimensions, and $[-2, 4]$ m for the $z$ dimension. For few-frame models, we use five features $(x, y, z, i, e, t)$ for each point, where $(x, y, z)$ are the point location, $i$ is the intensity normalized to $[0, 1]$, $e$ is the elongation, and $t$ is the timestamp offset in seconds. For the single-frame model, we exclude timestamp offset $t$. Since the nuScenes dataset does not provide elongation information, we set $e = 0$ for all nuScenes point clouds. Moreover, following previous work [257, 286], we apply a $+1.8$ m offset to the $z$ dimension to approximately transform the nuScenes point clouds from sensor frame to ego vehicle frame.

**Full-sequence input:** For full-sequence models, we only use the $(x, y, z)$ channels. To reduce training time and memory consumption, we pre-compute the aggregated point cloud for each sequence and perform a voxel-downsampling step with $3.25\,\mathrm{cm}^3$ voxels. During training, the pre-computed aggregated point clouds are transformed using pose transformations and further uniformly downsampled to at most $1{,}000{,}000$ points. Similar to single and few-frame input, a $+1.8$ m offset is applied to nuScenes aggregated point clouds.

### 5.3.3 Implementation Details

We use CenterPoint [298] and VoxelNeXt [33] implemented in the open-source framework OpenPCDet [228] with minor modifications to make the models compatible to both nuScenes and Waymo datasets. All models are trained with a total batch size of 32, over multiple GPUs. We use the Adam optimizer with a one cycle learning rate schedule.

**Baseline:** All single-frame and few-frame models are trained for 36 epochs. The learning rate is set to $0.001$ for nuScenes models and $0.003$ for Waymo models.

**ST3D [286]** For the nuScenes $\rightarrow$ Waymo direction, the models are fine-tuned for 12 epochs with a learning rate of $0.0001$. The positive and negative confidence thresholds for pseudo-labelling are set to $(0.5, 0.3)$ for Direct pseudo-labels and $(0.1, 0.05)$ for SOAP pseudo-labels. For the Waymo $\rightarrow$ nuScenes direction, the models are fine-tuned for 6 epochs with a learning rate of $0.0003$. The positive and negative confidence thresholds are set to $(0.6, 0.2)$ for Direct pseudo-labels and $(0.3, 0.2)$ for SOAP pseudo-labels. In all experiments, Direct pseudo-labels are updated every 2 epochs using the memory ensemble proposed in ST3D.

**SSDA3D [257]** Both stages in SSDA3D experiments follow the baseline training configurations. The CutMix and MixUp augmentation probabilities are set to $0.5$. Predictions from the first stage models are filtered by a confidence threshold of $0.3$ to construct the corresponding pseudo-labels for second stage training. When SOAP predictions are used, confidence thresholds of $0.15$ and $0.25$ are used to construct Waymo and nuScenes pseudo-labels, respectively.

**SOAP** The SOAP model is initialized with the weights from a corresponding few-frame model and trained for an additional 12 epochs with a learning rate of $0.001$ for nuScenes and $0.003$ for Waymo. As described in the main text, the annotations are constructed using QST. We set the QSS threshold $\epsilon$ to $0.7$ and $0.85$ for nuScenes and Waymo datasets, respectively. The post-processing for each dataset follows the implementation in OpenPCDet. For the proposed SCP step, which clusters and filters predictions in the global coordinate system, the cluster size threshold $\eta$ depends on the frame rate of the dataset. We use $\eta = 10$ for Waymo (10 Hz) and $\eta = 2$ for nuScenes (2 Hz). The cluster threshold $\mu$ for both SCP and WBF are set to $0.5$.

### 5.3.4 Evaluation

For nuScenes evaluation, we consider two primary metrics: mean Average Precision (mAP) and NuScenes Detection Score (NDS). Following the official evaluation, mAP is calculated based on four distance thresholds (0.5, 1.0, 2.0, 4.0) and averaged. As distance-based mAP does not penalize other types of bounding box errors, NDS is used in combination to reflect the average translation, scale, orientation, velocity, and attribute errors for the true positive predictions. All evaluations are performed on the validation split consisting of 150 sequences.

For Waymo evaluation, we use the official evaluation suite and report the Level 1 and Level 2 AP scores. Different from nuScenes mAP, Waymo AP is calculated based on 3D IoU with a threshold of 0.7. Level 1 evaluation includes only objects with more than 5 points within the bounding box, while Level 2 evaluation considers all objects. All evaluations are performed on the validation split consisting of 202 sequences.

## 5.4 Results

### 5.4.1 Unsupervised Domain Adaptation

We first evaluate SOAP pseudo-labels in the unsupervised domain adaptation setting, where annotations from the target domain are unavailable. We compare SOAP with two baseline approaches: "Direct" and ST3D [286]. Direct is where a few-frame detector is trained on the source

domain and directly evaluated on the target domain data. ST3D is a SOTA unsupervised domain adaptation method based on pseudo-labelling and self-training.

In the baseline comparison, ST3D utilizes the Direct model to generate pseudo-labels for self-training. To demonstrate the quality of the SOAP pseudo-labels and the complementary nature of SOAP with other approaches, we further consider ST3D + SOAP, where ST3D uses SOAP pseudo-labels. Both ST3D experiments use the official code release.

SOAP is validated using two object detection architectures: CenterPoint[298] and VoxelNeXt [33]. CenterPoint is a widely-adopted voxel-based dense 3D object detector. VoxelNeXt is a SOTA architecture representing recent advances in fully-sparse 3D object detectors. Both architectures are based on the implementation in the open-source library OpenPCDet. We use the Direct model predictions as few-frame predictions to construct final SOAP pseudo-labels. The main results are shown in Tables 5.2 and 5.3.

**Pseudo-label performance:** Overall, SOAP pseudo-labels improve over the Direct pseudo-labelling baseline by a significant margin. In the nuScenes $\rightarrow$ Waymo setting (Table 5.2), both architectures receive a 25–30 point improvement in mAP, with over 50% domain gap closed. SOAP is as effective in the Waymo $\rightarrow$ nuScenes setting (Table 5.3), with both architectures showing a 10-point improvement in mAP and over 30% domain gap closed.

More importantly, SOAP consistently improves object pseudo-labels at different ranges, with the largest improvements observed for objects at the 30–50 m range in all settings, closing 40–60% of the domain gap. We suppose this is because objects farther away from the sensor have sparse point clouds and are sometimes occluded. The results highlight the benefits of full-sequence aggregation, as far objects are densified, and occlusion can be alleviated by aggregating multiple viewpoints.

| Architecture | Method | Training Data | Overall | | | | 0–30 m | | | | 30–50 m | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Level 1 | | Level 2 | | Level 1 | | Level 2 | | Level 1 | | Level 2 | |
| CenterPoint [298] | Direct | $\{\mathcal{S}\}$ | 23.5 | | 20.2 | | 49.3 | | 48.3 | | 12.0 | | 10.5 | |
| | SOAP (ours) | | **50.9** | **+51.0%** | **45.4** | **+51.2%** | **69.4** | **+47.0%** | **68.4** | **+47.1%** | **47.5** | **+55.5%** | **43.6** | **+55.5%** |
| | ST3D [286] | $\{\mathcal{S}, \mathcal{T}_P\}$ | 28.6 | +9.5% | 24.6 | +8.9% | 56.5 | +16.8% | 55.3 | +16.4% | 18.7 | +10.5% | 16.5 | +10.1% |
| | ST3D + SOAP (ours) | | **46.4** | **+42.6%** | **40.6** | **+41.5%** | **68.6** | **+45.1%** | **67.5** | **+45.0%** | **41.8** | **+46.6%** | **37.6** | **+45.5%** |
| | Oracle | $\{\mathcal{T}\}$ | 77.2 | | 69.4 | | 92.1 | | 91.0 | | 76.0 | | 70.1 | |
| VoxelNeXt [33] | Direct | $\{\mathcal{S}\}$ | 20.4 | | 17.5 | | 44.2 | | 43.3 | | 9.5 | | 8.4 | |
| | SOAP (ours) | | **50.9** | **+53.4%** | **45.6** | **+53.8%** | **67.5** | **+48.5%** | **66.4** | **+48.3%** | **48.6** | **+58.5%** | **44.7** | **+58.6%** |
| | ST3D [286] | $\{\mathcal{S}, \mathcal{T}_P\}$ | 35.0 | +25.6% | 30.2 | +25.3% | 60.7 | +34.4% | 59.5 | +33.9% | 27.9 | +27.5% | 24.8 | +26.5% |
| | ST3D + SOAP (ours) | | **45.6** | **+44.1%** | **40.0** | **+43.1%** | **65.9** | **+45.2%** | **64.8** | **+45.0%** | **41.8** | **+48.4%** | **37.7** | **+47.3%** |
| | Oracle | $\{\mathcal{T}\}$ | 77.5 | | 69.7 | | 92.2 | | 91.1 | | 76.3 | | 70.3 | |

$\mathcal{S}$: labelled source domain; $\mathcal{T}$: labelled target domain; $\mathcal{T}_P$: pseudo-labelled target domain

Table 5.2: Unsupervised domain adaptation results for nuScenes → Waymo, where Waymo dataset is unlabelled.

| Architecture | Method | Training Data | Overall | | | | 0–30 m | | | | 30–50 m | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | mAP | | NDS | | mAP | | NDS | | mAP | | NDS | |
| CenterPoint [298] | Direct | $\{\mathcal{S}\}$ | 51.7 | | 69.6 | | 67.6 | | 78.9 | | 27.6 | | 49.7 | |
| | SOAP (ours) | | **61.4** | **+30.3%** | **76.9** | **+39.2%** | **73.1** | **+21.7%** | **83.9** | **+33.3%** | **41.9** | **+40.1%** | **64.4** | **+60.2%** |
| | ST3D [286] | $\{\mathcal{S}, \mathcal{T}_P\}$ | 59.3 | +23.8% | 72.9 | +17.7% | **74.2** | **+26.1%** | 82.4 | +23.3% | 34.1 | +18.2% | 52.7 | +12.3% |
| | ST3D + SOAP (ours) | | **61.5** | **+30.6%** | **75.4** | **+31.2%** | 73.9 | +24.9% | **83.0** | **+27.3%** | **42.5** | **+41.7%** | **60.4** | **+43.9%** |
| | Oracle | $\{\mathcal{T}\}$ | 83.7 | | 88.2 | | 92.9 | | 93.9 | | 63.3 | | 74.1 | |
| VoxelNeXt [33] | Direct | $\{\mathcal{S}\}$ | 49.0 | | 68.3 | | 62.5 | | 76.5 | | 28.8 | | 50.6 | |
| | SOAP (ours) | | **61.5** | **+36.2%** | **77.0** | **+44.2%** | **72.5** | **+32.5%** | **83.5** | **+40.5%** | **43.9** | **+45.5%** | **65.3** | **+65.3%** |
| | ST3D [286] | $\{\mathcal{S}, \mathcal{T}_P\}$ | 54.2 | +15.1% | 70.8 | +12.7% | **64.6** | **+6.8%** | 77.8 | +7.5% | 38.0 | +27.7% | 55.3 | +20.9% |
| | ST3D + SOAP (ours) | | **56.0** | **+20.3%** | **72.7** | **+22.3%** | **64.6** | **+6.8%** | **78.3** | **+10.4%** | **43.7** | **+44.9%** | **61.2** | **+47.1%** |
| | Oracle | $\{\mathcal{T}\}$ | 83.5 | | 88.0 | | 93.3 | | 93.8 | | 62.0 | | 73.1 | |

$\mathcal{S}$: labelled source domain; $\mathcal{T}$: labelled target domain; $\mathcal{T}_P$: pseudo-labelled target domain

Table 5.3: Unsupervised domain adaptation results for Waymo → nuScenes, where nuScenes dataset is unlabelled. The percentages represent the amount of the Direct–Oracle domain gap closed.

**Adaptation performance:** SOAP complements the SOTA domain adaptation method ST3D. We observe that when equipped with SOAP pseudo-labels, ST3D + SOAP provides better overall performance than ST3D. In the nuScenes $\rightarrow$ Waymo setting (Table 5.2), ST3D + SOAP closes 20–30% more domain gap than ST3D. While the difference is smaller in the Waymo $\rightarrow$ nuScenes setting (Table 5.3), there is still a noticeable improvement, with around 5% more domain gap closed for mAP and over 10% more domain gap closed for NDS. Moreover, the aforementioned improvement over far objects can also be seen after self-training with ST3D. Using SOAP pseudo-labels achieves significantly higher performance compared to ST3D.

### 5.4.2 Semi-supervised Domain Adaptation

In the semi-supervised domain adaptation setting, where a small amount of target domain annotations are available for training, SOAP can also be used to improve pseudo-label quality. To demonstrate this, we compare SOAP with three methods: Direct, Co-training[257], and SSDA3D [257]. As in the unsupervised case, Direct is where the model is trained only on source domain data. Co-training is where the model is trained with a combination of labelled source and labelled target domain data. SSDA3D is a recent SOTA semi-supervised domain adaptation technique. SSDA3D consists of a pseudo-labelling stage with inter-domain CutMix augmentation to improve pseudo-label quality (which we denote CutMix), followed by a target domain training stage with intra-domain MixUp augmentation as regularization. Additionally, we explore the SSDA3D + SOAP configuration, where we replace the SSDA3D pseudo-labels with SOAP pseudo-labels for second-stage target domain training.

Following SSDA3D [257], we use CenterPoint for experiments in this section and consider 1% sequences labelled in the target domain. Note that, unlike experiments in SSDA3D, we uniformly sample entire sequences instead of individual frames. Following how SSDA3D's pseudo-labelling model is trained, the SOAP model is trained on both labelled source and labelled target sequences with CutMix, among other standard augmentation, applied to aggregated point clouds. We use the SSDA3D CutMix predictions as sparse predictions to construct final SOAP pseudo-labels.

The main results are shown in Tables 5.4 and 5.5.

| Method | Training Data | Overall | | | | 0–30 m | | | | 30–50 m | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Level 1 | | Level 2 | | Level 1 | | Level 2 | | Level 1 | | Level 2 | |
| Direct | $\{\mathcal{S}\}$ | 23.5 | | 20.2 | | 49.3 | | 48.3 | | 12.0 | | 10.5 | |
| Co-training | $\{\mathcal{S}, \mathcal{T}_{\subset}\}$ | 58.4 | +65.0% | 51.3 | +63.2% | 79.3 | +70.1% | 78.1 | +69.8% | 55.7 | +68.3% | 50.3 | +66.8% |
| CutMix [257] | | 58.6 | +65.4% | 51.6 | +63.8% | 78.9 | +69.2% | 77.6 | +68.6% | 55.4 | +67.8% | 50.0 | +66.3% |
| SOAP (ours) | | **69.4** | **+85.5%** | **63.1** | **+87.2%** | **84.6** | **+82.5%** | **83.5** | **+82.4%** | **67.8** | **+87.2%** | **63.0** | **+88.1%** |
| SSDA3D [257] | $\{\mathcal{S}, \mathcal{T}_{\subset}, \mathcal{T}_P\}$ | 67.2 | +81.4% | 59.8 | +80.5% | 84.5 | +82.2% | 83.3 | +82.0% | 65.1 | +83.0% | 59.4 | +82.0% |
| SSDA3D + SOAP (ours) | | **70.1** | **+86.8%** | **62.6** | **+86.2%** | **86.1** | **+86.0%** | **84.9** | **+85.7%** | **68.5** | **+88.3%** | **62.8** | **+87.8%** |
| Oracle | $\{\mathcal{T}\}$ | 77.2 | | 69.4 | | 92.1 | | 91.0 | | 76.0 | | 70.1 | |

$\mathcal{S}$: labelled source domain; $\mathcal{T}$: labelled target domain; $\mathcal{T}_{\subset}$: small subset of $\mathcal{T}$; $\mathcal{T}_P$: pseudo-labelled target domain

Table 5.4: Semi-supervised domain adaptation results for nuScenes → Waymo, where 1% of Waymo data is labelled. The percentages represent the amount of the Direct–Oracle domain gap closed.



| Method | Training Data | Overall | | | | 0–30 m | | | | 30–50 m | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | mAP | | NDS | | mAP | | NDS | | mAP | | NDS | |
| Direct | $\{\mathcal{S}\}$ | 51.7 | | 69.6 | | 67.6 | | 78.9 | | 27.6 | | 49.7 | |
| Co-training | $\{\mathcal{S}, \mathcal{T}_{\subset}\}$ | 64.1 | +38.8% | 76.4 | +36.6% | 80.3 | +50.2% | 85.9 | +46.7% | 35.1 | +21.0% | 55.5 | +23.8% |
| CutMix [257] | | 63.6 | +37.2% | 76.7 | +38.2% | 82.3 | +58.1% | 87.4 | +56.7% | 29.9 | +6.4% | 52.7 | +12.3% |
| SOAP (ours) | | **74.0** | **+69.7%** | **83.5** | **+74.7%** | **86.4** | **+74.3%** | **90.6** | **+78.0%** | **49.6** | **+61.6%** | **68.0** | **+75.0%** |
| SSDA3D [257] | $\{\mathcal{S}, \mathcal{T}_{\subset}, \mathcal{T}_P\}$ | 72.7 | +65.6% | 81.3 | +62.9% | 88.1 | +81.0% | **90.4** | **+76.7%** | 42.8 | +42.6% | 60.1 | +42.6% |
| SSDA3D + SOAP (ours) | | **76.8** | **+78.4%** | **83.1** | **+72.6%** | **88.3** | **+81.8%** | **90.4** | **+76.7%** | **52.9** | **+70.9%** | **65.6** | **+65.2%** |
| Oracle | $\{\mathcal{T}\}$ | 83.7 | | 88.2 | | 92.9 | | 93.9 | | 63.3 | | 74.1 | |

$\mathcal{S}$: labelled source domain; $\mathcal{T}$: labelled target domain; $\mathcal{T}_{\subset}$: small subset of $\mathcal{T}$; $\mathcal{T}_P$: pseudo-labelled target domain

Table 5.5: Semi-supervised domain adaptation results for Waymo → nuScenes, where 1% of nuScenes data is labelled. The percentages represent the amount of the Direct–Oracle domain gap closed.

**Pseudo-label performance:** Compared to the pseudo-labels generated by Co-training and CutMix proposed in SSDA3D, SOAP pseudo-labels are much more accurate, closing 85.5% and 69.7% domain gap for nuScenes → Waymo (Table 5.4) and Waymo → nuScenes (Table 5.5), respectively. Similar to the results in the unsupervised settings, the improvement is particularly noticeable for objects at 30–50 m, further illustrating the benefits of full-sequence aggregation.

**Adaptation performance:** SOAP improves the already impressive domain adaptation performance achieved by SSDA3D, closing 5–10% more overall performance gap in both settings. Moreover, we observe the improvements in pseudo-labels for objects at 30–50 m translate to the model after adaptation. In the Waymo → nuScenes setting (Table 5.5), training with SOAP pseudo-labels achieves 10.1% higher mAP and 5.5% higher NDS for 30–50 m objects.

### 5.4.3 Speed Analysis

As mentioned earlier, we observe that stationary objects are a statistically important component of object detection. To demonstrate our previous improvements are not biased by the stationary objects, we perform additional evaluation based on speed for nuScenes → Waymo CenterPoint models in Tables 5.6 and 5.7. First, we notice that in all cases, the stationary performance of SOAP pseudo-labels and models fine-tuned with SOAP pseudo-labels exceeds SOTA by a significant margin, highlighting the effectiveness of our proposed method. Second, interestingly, while the pseudo-label performance for dynamic objects is on par or worse than the few-frame baseline (Direct and Co-training), after fine-tuning with the SOAP pseudo-labels using ST3D or SSDA3D, the dynamic performance is consistently better than SOTA methods. We hypothesize that this is because in the sparse point clouds, there is little distinction between stationary and dynamic objects, improving pseudo-labels for stationary objects thus leads to better overall performance after fine-tuning.

### 5.4.4 Qualitative Results

We present qualitative results of SOAP pseudo-labels for nuScenes → Waymo in figs. 5.5 and 5.6. In both unsupervised and semi-supervised settings, we observe that SOAP pseudo-labels are more accurate compared to Direct, Co-training, and CutMix [257], especially for far objects.

| Method | Training Data | Stationary (<0.2 m/s) | | | | Slow (0.2-1 m/s) | | | | Medium (1-3 m/s) | | | | Fast (3-10 m/s) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Level 1 | | Level 2 | | Level 1 | | Level 2 | | Level 1 | | Level 2 | | Level 1 | | Level 2 | |
| Direct | $\{\mathcal{S}\}$ | 21.5 | | 18.0 | | **24.3** | | **22.3** | | **27.5** | | **25.5** | | **27.1** | | **25.4** | |
| SOAP (ours) | | **56.2** | **+63.3%** | **49.5** | **+63.5%** | 23.9 | -0.9% | 22.0 | -0.7% | 24.4 | -7.4% | 22.6 | -7.4% | 25.7 | -2.8% | 24.1 | -2.7% |
| ST3D [286] | $\{\mathcal{S}, \mathcal{T}_P\}$ | 27.1 | +10.2% | 22.8 | +9.7% | 26.1 | +3.9% | 24.0 | +4.0% | 23.9 | -8.6% | 22.1 | -8.6% | 28.8 | +3.4% | 27.0 | +3.4% |
| ST3D + SOAP (ours) | | **48.6** | **+49.5%** | **41.7** | **+47.8%** | **32.6** | **+17.9%** | **30.0** | **+17.9%** | **32.2** | **+11.2%** | **29.9** | **+11.2%** | **34.4** | **+14.7%** | **32.3** | **+14.5%** |
| Oracle | $\{\mathcal{T}\}$ | 76.3 | | 67.6 | | 70.6 | | 65.3 | | 69.6 | | 64.9 | | 76.7 | | 73.1 | |

$\mathcal{S}$: labelled source domain; $\mathcal{T}$: labelled target domain; $\mathcal{T}_{\subset}$: small subset of $\mathcal{T}$; $\mathcal{T}_P$: pseudo-labelled target domain

Table 5.6: Unsupervised domain adaptation results for nuScenes → Waymo, where Waymo dataset is unlabelled, split based on object speed. The percentages represent the amount of the Direct–Oracle domain gap closed.



| Method | Training Data | Stationary (<0.2 m/s) | | | | Slow (0.2-1 m/s) | | | | Medium (1-3 m/s) | | | | Fast (3-10 m/s) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Level 1 | | Level 2 | | Level 1 | | Level 2 | | Level 1 | | Level 2 | | Level 1 | | Level 2 | |
| Direct | $\{\mathcal{S}\}$ | 21.5 | | 18.0 | | 24.3 | | 22.3 | | 27.5 | | 25.5 | | 27.1 | | 25.4 | |
| Co-training | $\{\mathcal{S}, \mathcal{T}_{\subset}\}$ | 58.1 | +66.2% | 50.0 | +64.4% | 50.1 | +52.3% | 46.2 | +52.1% | 44.5 | +37.7% | 41.3 | +37.6% | 51.5 | +47.9% | 48.7 | +47.5% |
| CutMix [257] | | 57.4 | +65.1% | 49.6 | +63.4% | **51.1** | **+54.3%** | **47.1** | **+54.1%** | **48.0** | **+45.4%** | **44.5** | **+45.2%** | **57.2** | **+59.2%** | **54.2** | **+58.7%** |
| SOAP (ours) | | **71.9** | **+91.3%** | **64.6** | **+93.6%** | 39.4 | +30.6% | 36.4 | +30.8% | 38.6 | +24.6% | 35.8 | +24.5% | 51.8 | +48.5% | 49.0 | +48.2% |
| SSDA3D [257] | $\{\mathcal{S}, \mathcal{T}_{\subset}, \mathcal{T}_P\}$ | 66.5 | +81.6% | 58.2 | +80.7% | 56.3 | +64.8% | 52.0 | +64.6% | 53.8 | +58.2% | 50.0 | +58.2% | 62.6 | +69.7% | 59.4 | +69.2% |
| SSDA3D + SOAP (ours) | | **70.1** | **+88.0%** | **61.6** | **+87.5%** | **58.7** | **+69.7%** | **54.3** | **+69.6%** | **54.6** | **+60.0%** | **50.7** | **+59.9%** | **63.6** | **71.8%** | **60.3** | **+71.2%** |
| Oracle | $\{\mathcal{T}\}$ | 76.7 | | 67.8 | | 73.7 | | 68.2 | | 72.6 | | 67.6 | | 78.0 | | 74.4 | |

$\mathcal{S}$: labelled source domain; $\mathcal{T}$: labelled target domain; $\mathcal{T}_{\subset}$: small subset of $\mathcal{T}$; $\mathcal{T}_P$: pseudo-labelled target domain

Table 5.7: Semi-supervised domain adaptation results for nuScenes → Waymo, where 1% of Waymo data is labelled, split based on object speed. The percentages represent the amount of the Direct–Oracle domain gap closed.

| | 1-frame | 5-frame | SFA | QST | SCP | Overall | | | | Stationary | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Level 1 | | Level 2 | | Level 1 | | Level 2 | |
| **(a)** | ✓ | | | | | 7.0 | -23.5% | 6.0 | -22.0% | 5.9 | -21.9% | 4.9 | -20.5% |
| **(b)** | | ✓ | | | | 20.4 | | 17.5 | | 18.6 | | 15.6 | |
| **(c)** | | ✓ | ✓ | | | 35.7 | +36.6% | 31.2 | +35.6% | 38.0 | +50.1% | 32.6 | +48.3% |
| **(d)** | | ✓ | ✓ | ✓ | | 46.7 | +46.1% | 41.6 | +46.2% | 52.1 | +57.7% | 45.8 | +57.9% |
| **(e)** | | ✓ | ✓ | ✓ | ✓ | **50.9** | **+53.4%** | **45.6** | **+53.8%** | **57.2** | **+66.4%** | **50.6** | **+67.0%** |
| **Oracle** | | | | | | 77.5 | | 69.7 | | 76.7 | | 67.8 | |

Table 5.8: Ablation study results for nuScenes → Waymo unsupervised domain adaptation. The percentages represent the amount of domain gap closed relative to the 5-frame baseline detector.

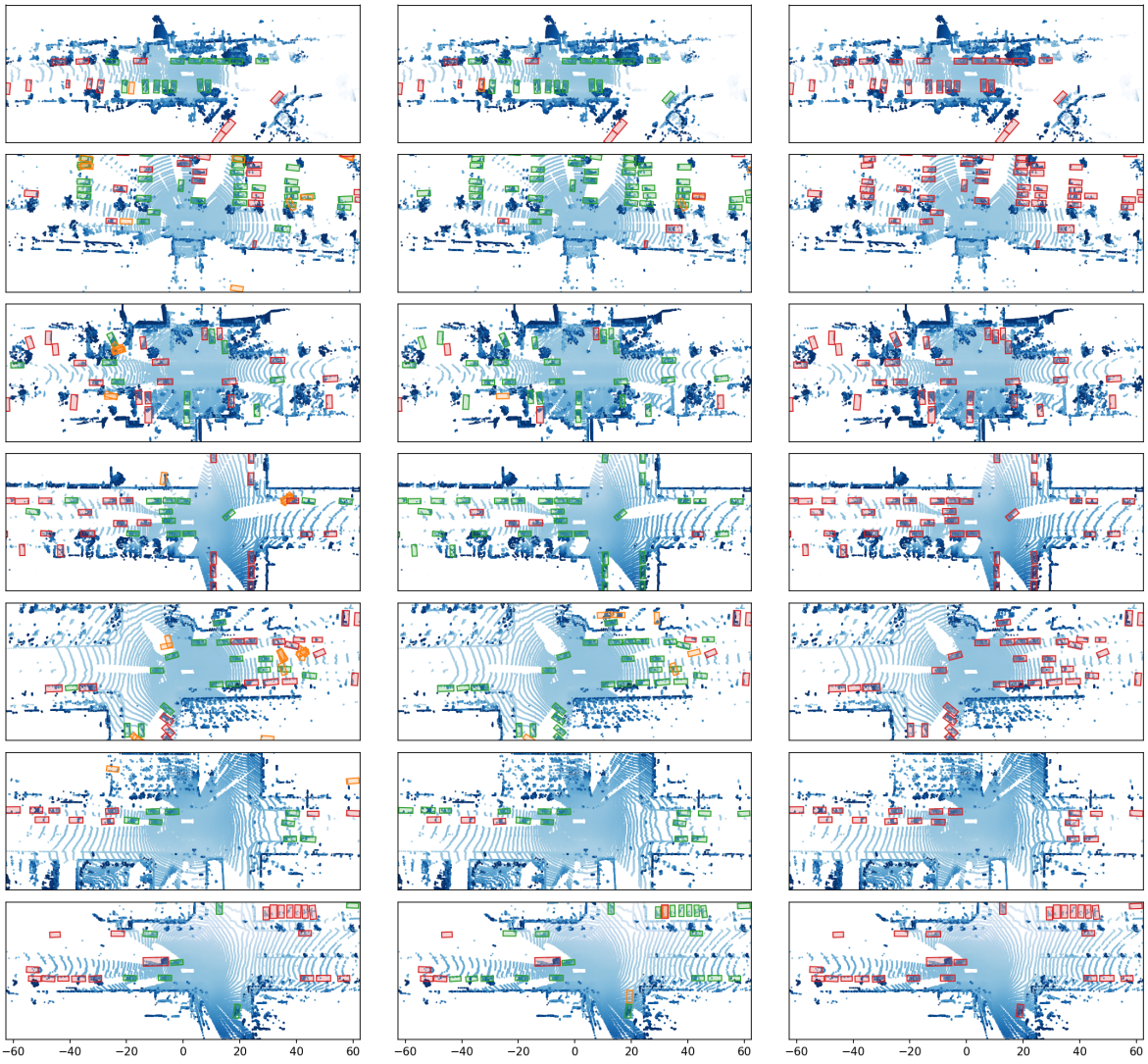
### 5.4.5 Ablation Study

In this section, we investigate the benefits of QST and SCP in the nuScenes → Waymo unsupervised domain adaptation setting, using the VoxelNeXt architecture. The results are presented in Table 5.8.

We use the 1-frame and 5-frame detectors as baselines in lines (a) and (b), respectively, and progressively introduce each component of SOAP. In line (c), SFA augments the 5-frame detector with stationary object predictions using aggregated point clouds. The model is trained by naively filtering object speed based on a threshold of 0.2 m/s. Line (d) enables QST, replacing naive filtering. While the SFA pseudo-labels improve over both the 1-frame and 5-frame baselines, especially for stationary objects, it is still significantly outperformed by QST. This highlights both the effectiveness of full-sequence aggregation in cross-sensor settings and the importance of constructing robust training labels for stationary objects using QST. Moreover, incorporating SCP in line (e) further improves the AP by over 4%, demonstrating the benefit of exploiting the stationarity of the detected objects.

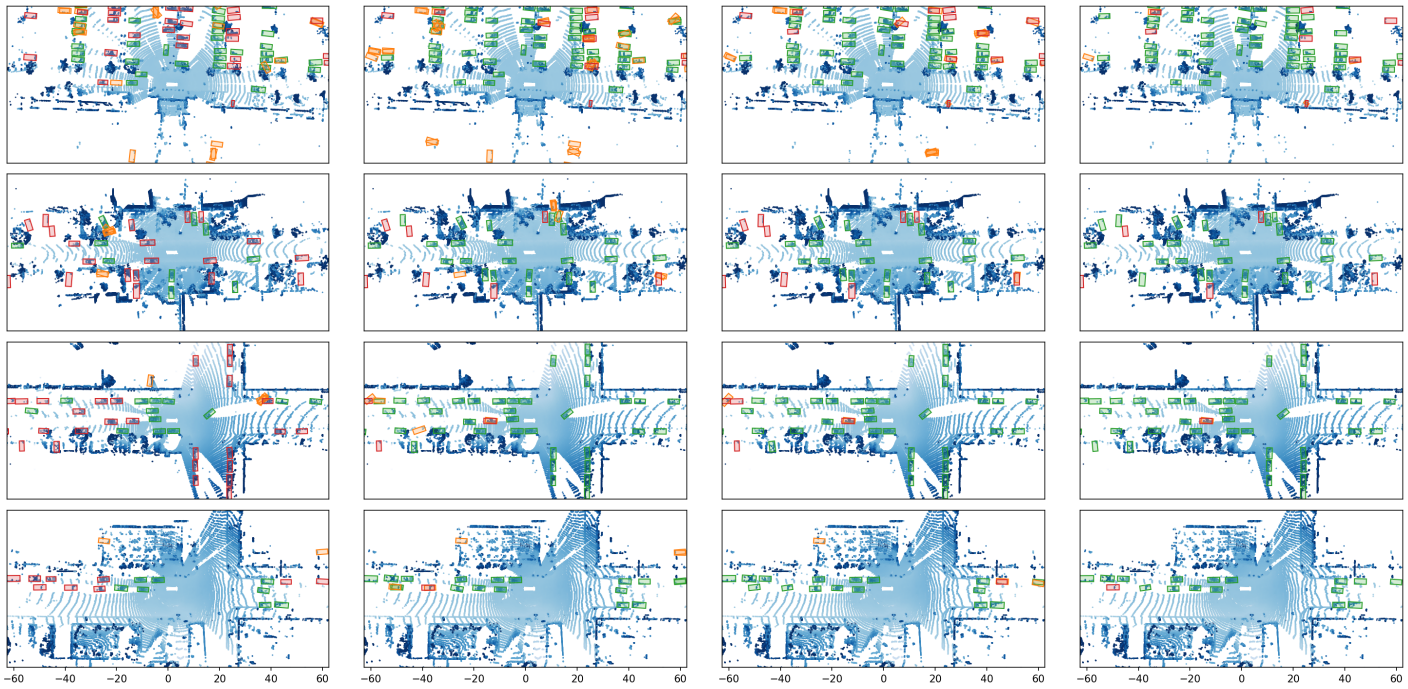
## 5.5 Limitations and Future Work

SOAP is designed to address the domain gap caused by distinct LiDAR scan patterns. However, its effectiveness may be limited in cases where these scan patterns cannot be mitigated. This can happen when an object does not have enough observations due to distance or occlusion, or when the ego-vehicle does not have sufficient movement relative to the object, failing to provide scans from multiple view points. In these scenarios, alternative domain adaptation methods, such as




(a) Direct (b) SOAP (c) Ground Truth

Figure 5.5: Examples of pseudo-labels generated by different methods in nuScenes → Waymo unsupervised domain adaptation setting, and the corresponding ground truth labels. Green represents true positive pseudo-labels, orange represents false positive pseudo-labels, and red represents false negative pseudo-labels.




(a) Direct (b) Co-training (c) CutMix (SSDA3D) (d) SOAP

Figure 5.6: Examples of pseudo-labels generated by different methods in nuScenes → Waymo semi-supervised domain adaptation setting. Green represents true positive pseudo-labels, orange represents false positive pseudo-labels, and red represents false negative pseudo-labels.

self-training [286, 288], domain mapping [11], or data augmentation approaches like MixUp [34] and CutMix [303], might be more effective. Future works could further investigate the synergy between SOAP and these complementary domain adaptation methods.

Additionally, since SOAP is specifically designed to enhance performance on stationary objects, its benefits diminish for objects that are rarely stationary, such as pedestrians and cyclists, or in scenes dominated by dynamic objects. For instance, unstructured traffic scenes with many moving objects, such as the ones in the Indian Driving Dataset (IDD-3D) [52], can pose a significant challenge to SOAP, and further studies are needed to evaluate its performance in such dynamic environments. To fundamentally address this limitation, future research could explore scene flow estimation and point cloud registration to better align dynamic object point clouds and improve dynamic pseudo-label quality.

Algorithmically, SOAP consists of a combination of deep neural networks and classical algorithms, resulting in a disjointed pipeline that is both challenging to use and prone to errors. In general, classical algorithms have been consistently outperformed by machine learning methods, making them the potential bottleneck in the current design. A promising direction is to develop an end-to-end approach for pseudo-labelling and label refinement, streamlining the entire training and deployment process.

In terms of computational efficiency, the base detector in SOAP must process extremely dense point clouds resulting from full-sequence aggregation. While the voxel-based architectures used in this thesis can handle such data, SOTA point-based or transformer-based object detectors might struggle with this increased density. Therefore, it is crucial to refine the aggregation process and adopt point cloud pruning strategies, such as ground removal, to improve the computational efficiency of large SOTA models.

## 5.6 Conclusion

We have presented Stationary Object Aggregation Pseudo-labelling (SOAP), a novel method that utilizes full-sequence scene-level aggregation to generate high-quality pseudo-labels for the cross-sensor domain adaptation setting. We have provided extensive evaluation that demonstrates SOAP can provide high-quality pseudo-labels and improves the already impressive results achieved by SOTA methods such as ST3D and SSDA3D.

# Chapter 6

# Discussions and Conclusion

In this thesis, we have presented three methods to advance LiDAR-based perception from multi-frame point clouds. To summarize, we have made the following three contributions to address the challenges and limitations of existing works:

- In Chapter 3, we proposed VADet that uses variable aggregation as a simple yet effective alternative to the widely adopted fix aggregation. By adapting the number of aggregated frames on a per-object basis, VADet effectively mitigates the trade-offs inherent in multi-frame 3D object detection, achieving state-of-the-art performance across various architectures.

- In Chapter 4, we proposed MFSeg, an efficient multi-frame 3D semantic segmentation framework built upon the SOTA point-based transformer PTv3 [267]. We introduced several architectural modifications to allow more efficient point cloud processing, and proposed a novel training pipeline to retain the benefit of multi-frame point cloud aggregation.

- In Chapter 5, we proposed SOAP, a novel application of long sequence aggregation to cross-sensor domain adaptation. By aggregating entire point cloud sequences at the scene level and employing a post-processing pipeline to preserve spatial consistency, SOAP generates high-quality pseudo-labels that enhance the performance of existing domain adaptation methods such as ST3D and SSDA3D.

The three proposed methods have potentially wide applicability in autonomous driving, especially those using LiDAR based object detectors and segmenters as dedicated modules. Both VADet and MFSeg can be seamlessly integrated into the traditional autonomous driving system (ADS) to improve perception performance: VADet can replace the fixed-aggregation preprocessing in any LiDAR detector, and MFSeg can be added on to any point-based segmentation backbone. Likewise, SOAP can generate reliable pseudo-labels for unlabelled/partially labelled target-domain data when the perception system needs to adapt. Because all three methods extend existing backbone architectures, they can be adopted in future architectures to maintain state-of-the-art performance as the backbones evolve.

However, there are two practical considerations when applying our methods:

1. The need for accurately aligned multi-frame point clouds: All three approaches assume proper alignment of LiDAR point clouds via ego motion. In practice, ego motion estimated from onboard sensors—such as GPS/IMU or SLAM-based localization—often introduces drift and bias. In this thesis, we demonstrate strong results on carefully curated datasets with minimal localization error. However, a suboptimal setup could result in misaligned and distorted multi-frame point clouds. VADet and MFSeg are designed for online operation using only a handful of frames, and thus they are likely to tolerate moderate localization errors. SOAP, by contrast, requires full-sequence aggregation and a reliable global coordinate frame to optimize stationary pseudo-labels, making it potentially more sensitive to localization errors. To fully understand the effect on each method, future work should therefore quantify each method's sensitivity to varying levels of localization error.

2. The disconnect between perception metrics and real-world driving performance: While we report improvements in average precision (AP) for object detection and intersection-over-union (IoU) for semantic segmentation, they do not necessarily translate into safer or smoother driving. As we demonstrate in VADet, higher metric values can conceal systematic failures in potentially critical scenarios, such as the detection of dynamic objects. Conversely, improvements in certain aspects of perception, such as better detection of distant objects, may offer minimal benefit for short-term planning. A more comprehensive evaluation in the context of a target ADS and its operational design domain is therefore essential to measure how these perception improvements translate into reduced crash rates and more reliable autonomous driving.

Recent research has shifted toward end-to-end autonomous driving, where the system is optimized and evaluated as a whole, to avoid the aforementioned issue. Most current end-to-end frameworks still rely on traditional sensory backbones for feature extraction, varying in how they process those features into planning outputs [26]. We believe that multi-frame point clouds can

benefit such systems and that the proposed methods can be adapted with minor modifications. However, using multi-frame point clouds in end-to-end ADS is underexplored and may pose unique challenges that warrant future investigation.

Building on the end-to-end paradigm, another emerging line of work applies foundation models such as vision-language models (VLMs) to autonomous driving [316]. Compared to models trained solely on driving data, a VLM-powered ADS takes advantage of large-scale diverse datasets to cover rare and edge cases that can be absent from driving datasets. However, our methods may not be compatible with these methods due to the large architectural differences between VLMs and conventional backbones. Moreover, because VLM-based ADS are inherently multi-modal and temporal, it remains unclear whether multi-frame point clouds would have additional benefit. Investigating these questions presents another promising direction for future research.

In conclusion, this thesis addresses the challenges and limitations of applying multi-frame point clouds to the perception component in a traditional ADS. We introduce VADet and MF-Seg to improve online detection and segmentation performance, and develop SOAP to generate robust pseudo-labels for domain adaptation. While these methods are designed for existing traditional modular ADS and may require modification to be applied to the evolving end-to-end paradigm, we believe they provide a solid foundation for future research. We hope this thesis will inspire investigation into the efficient use of multi-frame point clouds in emerging architectures, ultimately advancing the safety of future ADS.

# References


[1] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *International Journal of Computer Vision*, 126:961–972, 2018.

[2] Eren Erdal Aksoy, Saimir Baci, and Selcuk Cavdar. Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 926–932, 2020.

[3] Inigo Alonso, Luis Riazuelo, Luis Montesano, and Ana C. Murillo. Domain adaptation in lidar semantic segmentation by aligning class distributions, 2021. URL https://arxiv.org/abs/2010.12239.

[4] Iñigo Alonso, Luis Riazuelo, Luis Montesano, and Ana C. Murillo. 3d-mininet: Learning a 2d representation from point clouds for fast and efficient 3d lidar semantic segmentation. *IEEE Robotics and Automation Letters*, 5(4):5432–5439, 2020.

[5] Angelika Ando, Spyros Gidaris, Andrei Bursuc, Gilles Puy, Alexandre Boulch, and Renaud Marlet. Rangevit: Towards vision transformers for 3d semantic segmentation in autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5240–5250, June 2023.

[6] Nerea Aranjuelo, Guus Engels, Luis Unzueta, Ignacio Arganda-Carreras, Marcos Nieto, and Oihana Otaegui. Robust 3d object detection from lidar point cloud data with spatial information aggregation. In Álvaro Herrero, Carlos Cambra, Daniel Urda, Javier Sedano, Héctor Quintián, and Emilio Corchado, editors, *15th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2020)*, pages 813–823, Cham, 2021. Springer International Publishing.

[7] Mehul Arora, Louis Wiesmann, Xieyuanli Chen, and Cyrill Stachniss. Static map generation from 3d lidar point clouds exploiting ground segmentation. *Robotics and Autonomous*

*Systems*, 159:104287, 2023. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2022.104287. URL https://www.sciencedirect.com/science/article/pii/S0921889022001762.

[8] Isaac Asimov. Sally. *Fantastic*, 2(3):35–50, May–June 1953.

[9] Yeqi BAI, Ben Fei, Youquan Liu, Tao MA, Yuenan Hou, Botian Shi, and Yikang LI. Rangeperception: Taming lidar range view for efficient and accurate 3d object detection. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 79557–79569. Curran Associates, Inc., 2023.

[10] Alejandro Barrera, Carlos Guindel, Jorge Beltrán, and Fernando García. Birdnet+: End-to-end 3d object detection in lidar bird's eye view. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2020.

[11] Alejandro Barrera, Jorge Beltrán, Carlos Guindel, Jose Antonio Iglesias, and Fernando García. Cycle and semantic consistent adversarial domain adaptation for reducing simulation-to-real domain shift in lidar bird's eye view. In *IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3081–3086, 2021. doi: 10.1109/ITSC48978.2021.9564553.

[12] 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 (ICCV)*, October 2019.

[13] Behnam Behroozpour, Phillip A. M. Sandborn, Ming C. Wu, and Bernhard E. Boser. Lidar system architectures and circuits. *IEEE Communications Magazine*, 55(10):135–142, 2017.

[14] Jorge Beltrán, Carlos Guindel, Francisco Miguel Moreno, Daniel Cruzado, Fernando García, and Arturo De La Escalera. Birdnet: A 3d object detection framework from lidar information. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3517–3523, 2018.

[15] Maxim Berman, Amal Rannen Triki, and Matthew B. Blaschko. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[16] Borna Bešić, Nikhil Gosala, Daniele Cattaneo, and Abhinav Valada. Unsupervised domain adaptation for LiDAR panoptic segmentation. *IEEE Robotics and Automation Letters*, 7 (2):3404–3411, 2022.

[17] Alex Bewley, Pei Sun, Thomas Mensink, Dragomir Anguelov, and Cristian Sminchisescu. Range conditioned dilated convolutions for scale invariant 3d object detection. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 627–641. PMLR, 16–18 Nov 2021.

[18] Mario Bijelic, Tobias Gruber, Fahim Mannan, Florian Kraus, Werner Ritter, Klaus Dietmayer, and Felix Heide. Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[19] Andrew Brock, Theodore Lim, J. M. Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks, 2016. URL https://arxiv.org/abs/1608.04236.

[20] 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 (CVPR)*, June 2020.

[21] Benjamin Caine, Rebecca Roelofs, Vijay Vasudevan, Jiquan Ngiam, Yuning Chai, Zhifeng Chen, and Jonathon Shlens. Pseudo-labeling for scalable 3d object detection, 2021. URL https://arxiv.org/abs/2103.02093.

[22] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 2267–2281, 2019.

[23] Yulong Cao, S Hrushikesh Bhupathiraju, Pirouz Naghavi, Takeshi Sugawara, Z Morley Mao, and Sara Rampazzi. You can't see me: Physical removal attacks on {lidar-based} autonomous vehicles driving frameworks. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 2993–3010, 2023.

[24] Yuning Chai, Pei Sun, Jiquan Ngiam, Weiyue Wang, Benjamin Caine, Vijay Vasudevan, Xiao Zhang, and Dragomir Anguelov. To the point: Efficient 3d object detection in the

range image with graph convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2021.

[25] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[26] Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[27] Songle Chen, Lintao Zheng, Yan Zhang, Zhixin Sun, and Kai Xu. VERAM: View-Enhanced Recurrent Attention Model for 3D Shape Classification . *IEEE Transactions on Visualization & Computer Graphics*, 25(12):3244–3257, December 2019. ISSN 1941-0506.

[28] Wanli Chen, Xinge Zhu, Guojin Chen, and Bei Yu. Efficient point cloud analysis using hilbert curve. In *European Conference on Computer Vision*, pages 730–747. Springer, 2022.

[29] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[30] Xuesong Chen, Shaoshuai Shi, Benjin Zhu, Ka Chun Cheung, Hang Xu, and Hongsheng Li. Mppnet: Multi-frame feature intertwining with proxy points for 3d temporal object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 680–697, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-20074-8.

[31] Yukang Chen, Yanwei Li, Xiangyu Zhang, Jian Sun, and Jiaya Jia. Focal sparse convolutional networks for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5428–5437, June 2022.

[32] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia. Largekernel3d: Scaling up kernels in 3d sparse cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13488–13498, June 2023.

[33] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia. Voxelnext: Fully sparse voxelnet for 3d object detection and tracking. In *Proceedings of the IEEE/CVF*

*Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21674–21683, June 2023.

[34] Yunlu Chen, Vincent Tao Hu, Efstratios Gavves, Thomas Mensink, Pascal Mettes, Pengwan Yang, and Cees G. M. Snoek. Pointmixup: Augmentation for point clouds. In *Computer Vision – ECCV 2020*, pages 330–345, 2020.

[35] Hui–Xian Cheng, Xian–Feng Han, and Guo–Qiang Xiao. Cenet: Toward concise and efficient lidar semantic segmentation for autonomous driving. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 01–06, 2022.

[36] Ran Cheng, Ryan Razani, Ehsan Taghavi, Enxu Li, and Bingbing Liu. (af)2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12547–12556, June 2021.

[37] Sumit Chopra, Suhrid Balakrishnan, and Raghuraman Gopalan. Dlid: Deep learning for domain adaptation by interpolating between domains. In *ICML workshop on challenges in representation learning*, 2013.

[38] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[39] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, pages 424–432, 2016.

[40] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020.

[41] NVIDIA Corporation. Self-driving cars technology & solutions — nvidia automotive, 2024. URL https://www.nvidia.com/en-us/self-driving-cars/. Accessed: 2024-10-09.

[42] Eduardo R. Corral-Soto, Amir Nabatchian, Martin Gerdzhev, and Liu Bingbing. Lidar few-shot domain adaptation via integrated cyclegan and 3d object detector with joint learning delay. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 13099–13105, 2021. doi: 10.1109/ICRA48506.2021.9561466.

[43] Eduardo R. Corral-Soto, Mrigank Rochan, Yannis Y. He, Shubhra Aich, Yang Liu, and Liu Bingbing. Domain adaptation in lidar semantic segmentation via alternating skip connections and hybrid learning, 2022. URL https://arxiv.org/abs/2201.05585.

[44] Eduardo R. Corral-Soto, Mrigank Rochan, Yannis Y. He, Shubhra Aich, Yang Liu, and Liu Bingbing. Domain adaptation in LiDAR semantic segmentation via alternating skip connections and hybrid learning, 2022. URL https://arxiv.org/abs/2201.05585. arXiv: 2201.05585.

[45] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds. In *Advances in Visual Computing: 15th International Symposium, ISVC 2020, San Diego, CA, USA, October 5–7, 2020, Proceedings, Part II*, page 207–222, Berlin, Heidelberg, 2020. Springer-Verlag.

[46] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023. URL https://arxiv.org/abs/2307.08691.

[47] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pages 16344–16359, 2022.

[48] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *Proceedings of the AAAI conference on artificial intelligence*, pages 1201–1209, 2021.

[49] E.D. Dickmanns and A. Zapp. Autonomous high speed road vehicle guidance by computer vision1. *IFAC Proceedings Volumes*, 20(5, Part 4):221–226, 1987.

[50] Guangyao Ding, Meiying Zhang, E Li, and Qi Hao. Jst: Joint self-training for unsupervised domain adaptation on 2d&3d object detection. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 477–483, 2022. doi: 10.1109/ICRA46639.2022.9811975.

[51] Zhuangzhuang Ding, Yihan Hu, Runzhou Ge, Li Huang, Sijia Chen, Yu Wang, and Jie Liao. 1st place solution for waymo open dataset challenge – 3d detection and domain adaptation, 2020. URL https://arxiv.org/abs/2006.15505.

[52] Shubham Dokania, A. H. Abdul Hafez, Anbumani Subramanian, Manmohan Chandraker, and C. V. Jawahar. Idd-3d: Indian driving dataset for 3d unstructured road scenes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4482–4491, January 2023.

[53] Shaocong Dong, Lihe Ding, Haiyang Wang, Tingfa Xu, Xinli Xu, Jie Wang, Ziyang Bian, Ying Wang, and Jianan Li. Mssvt: Mixed-scale sparse voxel transformer for 3d object detection on point clouds. In *Advances in Neural Information Processing Systems*, volume 35, pages 11615–11628, 2022.

[54] Liang Du, Xiaoqing Ye, Xiao Tan, Jianfeng Feng, Zhenbo Xu, Errui Ding, and Shilei Wen. Associate-3ddet: Perceptual-to-conceptual association for 3d point cloud object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[55] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[56] João Espadinha, Ivan Lebedev, Luka Lukic, and Alexandre Bernardino. Lidar data noise models and methodology for sim-to-real domain generalization and adaptation in autonomous driving perception. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 797–803, 2021. doi: 10.1109/IV48863.2021.9576034.

[57] Carlos Esteves, Yinshuang Xu, Christine Allen-Blanchette, and Kostas Daniilidis. Equivariant multi-view networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[58] Lue Fan, Xuan Xiong, Feng Wang, Naiyan Wang, and ZhaoXiang Zhang. Rangedet: In defense of range view for lidar-based 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2918–2927, October 2021.

[59] Lue Fan, Ziqi Pang, Tianyuan Zhang, Yu-Xiong Wang, Hang Zhao, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Embracing single stride 3d object detector with sparse transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8458–8468, June 2022.

[60] Lue Fan, Feng Wang, Naiyan Wang, and ZHAO-XIANG ZHANG. Fully sparse 3d object detection. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 351–363. Curran Associates, Inc., 2022.

[61] Lue Fan, Yuxue Yang, Yiming Mao, Feng Wang, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Once detected, never lost: Surpassing human performance in offline lidar based 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19820–19829, October 2023.

[62] Lue Fan, Yuxue Yang, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Super sparse 3d object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (10):12490–12505, 2023.

[63] Lue Fan, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Fsd v2: Improving fully sparse 3d object detection with virtual voxels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(2):1279–1292, 2025.

[64] Jin Fang, Dingfu Zhou, Jingjing Zhao, Chenming Wu, Chulin Tang, Cheng-Zhong Xu, and Liangjun Zhang. Lidar-cs dataset: Lidar point cloud dataset with cross-sensors for 3d object detection, 2024. URL https://arxiv.org/abs/2301.12515.

[65] Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[66] Christian Fruhwirth-Reisinger, Michael Opitz, Horst Possegger, and Horst Bischof. FAST3D: Flow-aware self-training for 3D object detectors. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2021.

[67] Runzhou Ge, Zhuangzhuang Ding, Yihan Hu, Yu Wang, Sijia Chen, Li Huang, and Yuan Li. Afdet: Anchor free one stage 3d object detection, 2020. URL https://arxiv.org/abs/2006.12671.

[68] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[69] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[70] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S. Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian Mühlegg, Sebastian Dorn, Tiffany Fernandez, Martin Jänicke, Sudesh Mirashi, Chiragkumar Savani, Martin Sturm, Oleksandr Vorobiov, Martin Oelker, Sebastian Garreis, and Peter Schuberth. A2d2: Audi autonomous driving dataset, 2020. URL https://arxiv.org/abs/2004.06320.

[71] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks, 2017. URL https://arxiv.org/abs/1706.01307.

[72] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[73] Yi Gu, Yuming Huang, Chengzhong Xu, and Hui Kong. Maskrange: A mask-classification model for range-view based lidar segmentation, 2022. URL https://arxiv.org/abs/2206.12073.

[74] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7:187–199, 2021.

[75] Martin Hahner, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Fog simulation on real lidar point clouds for 3d object detection in adverse weather. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15283–15292, October 2021.

[76] Martin Hahner, Christos Sakaridis, Mario Bijelic, Felix Heide, Fisher Yu, Dengxin Dai, and Luc Van Gool. Lidar snowfall simulation for robust 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16364–16374, June 2022.

[77] Abdullah Hamdi, Silvio Giancola, and Bernard Ghanem. Mvtn: Multi-view transformation network for 3d shape recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1–11, October 2021.

[78] 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 (CVPR)*, June 2020.

[79] Chenhang He, Ruihuang Li, Shuai Li, and Lei Zhang. Voxel set transformer: A set-to-set approach to 3d object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8417–8427, June 2022.

[80] Chenhang He, Ruihuang Li, Guowen Zhang, and Lei Zhang. Scatterformer: Efficient voxel transformer with scattered linear attention. In *Computer Vision – ECCV 2024*, pages 74–92, Cham, 2025.

[81] Qingdong He, Zhengning Wang, Hao Zeng, Yi Zeng, and Yijun Liu. Svga-net: Sparse voxel-graph attention network for 3d object detection from point clouds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 870–878, 2022.

[82] Deepti Hegde and Vishal M. Patel. Attentive prototypes for source-free unsupervised domain adaptive 3d object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3066–3076, January 2024.

[83] Deepti Hegde, Vishwanath Sindagi, Velat Kilic, A. Brinton Cooper, Mark Foster, and Vishal Patel. Uncertainty-aware mean teacher for source-free unsupervised domain adaptive 3d object detection, 2021. URL https://arxiv.org/abs/2109.14651.

[84] Ke Hong, Zhongming Yu, Guohao Dai, Xinhao Yang, Yaoxiu Lian, Zehao Liu, Ningyi Xu, and Yu Wang. Exploiting Hardware Utilization and Adaptive Dataflow for Efficient Sparse Convolution in 3D Point Clouds. In *Sixth Conference on Machine Learning and Systems (MLSys)*, 2023.

[85] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: an efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, April 2013.

[86] Yuenan Hou, Xinge Zhu, Yuexin Ma, Chen Change Loy, and Yikang Li. Point-to-voxel knowledge distillation for lidar semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8479–8488, June 2022.

[87] Yihan Hu, Zhuangzhuang Ding, Runzhou Ge, Wenxin Shao, Li Huang, Kun Li, and Qiang Liu. Afdetv2: Rethinking the necessity of the second stage for object detection from point clouds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 969–979, 2022.

[88] Chengjie Huang, Vahdat Abdelzad, Sean Sedwards, and Krzysztof Czarnecki. Soap: Cross-sensor domain adaptation for 3d object detection using stationary object aggregation pseudo-labelling. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3352–3361, January 2024.

[89] Rui Huang, Wanyue Zhang, Abhijit Kundu, Caroline Pantofaru, David A. Ross, Thomas Funkhouser, and Alireza Fathi. An lstm approach to temporal 3d object detection in lidar point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 266–282, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58523-5.

[90] Il-Pyeong Hwang and Chang-Hee Lee. Mutual interferences of a true-random lidar with other lidar signals. *IEEE Access*, 8:124123–124133, 2020. doi: 10.1109/ACCESS.2020.3004891.

[91] Soonmin Hwang, Jaesik Park, Namil Kim, Yukyung Choi, and In So Kweon. Multispectral pedestrian detection: Benchmark dataset and baseline. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[92] Sunwook Hwang, Youngseok Kim, Seongwon Kim, Saewoong Bahk, and Hyung-Sin Kim. Upcycling: Semi-supervised 3d object detection without sharing raw-level unlabeled scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 23351–23361, October 2023.

[93] Aeva Inc. Atlas, 2024. URL https://www.aeva.com/atlas/. Accessed: 2024-10-09.

[94] AEye Inc. Lidar technology - aeye 4sight. https://www.aeye.ai/4sight-lidar/, 2024. Accessed: 2024-10-09.

[95] Aurora Innovation Inc. The aurora driver, 2024. URL https://aurora.tech/aurora-driver. Accessed: 2024-10-09.

[96] Mobileye Global Inc. Mobileye to end internal lidar development, 2024. URL https://www.mobileye.com/news/mobileye-to-end-internal-lidar-development/. Accessed: 2024-10-09.

[97] Mobileye Global Inc. Mobileye's fmcw lidar with gil tidhar, phd, 2024. URL https://www.youtube.com/watch?v=Vvg9heQObyQ. Accessed: 2024-10-09.

[98] Ouster Inc. *Ouster OS2 Long-Range High-Resolution Imaging Lidar*. Ouster Inc., June 2023. URL https://www.ouster.com.

[99] Ouster Inc. Digital lidar sensors for automation, drones & robotics — ouster. https://www.ouster.com/, 2024. Accessed: 2024-10-09.

[100] Maximilian Jaritz, Tuan-Hung Vu, Raoul de Charette, Emilie Wirbel, and Patrick Perez. xmuda: Cross-modal unsupervised domain adaptation for 3d semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[101] Peng Jiang and Srikanth Saripalli. Lidarnet: A boundary-aware domain adaptation model for point cloud semantic segmentation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2457–2464, 2021. doi: 10.1109/ICRA48506.2021.9561255.

[102] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[103] Shoji Kawahito, Izhal Abdul Halin, Takeo Ushinaga, Tomonari Sawada, Mitsuru Homma, and Yasunari Maeda. A cmos time-of-flight range image sensor with gates-on-field-oxide structure. *IEEE Sensors Journal*, 7(12):1578–1586, 2007.

[104] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Lyft level 5 perception dataset 2020. https://level5.lyft.com/dataset/, 2019.

[105] Gunzung Kim, Jeongsook Eom, and Yongwan Park. Investigation on the occurrence of mutual interference between pulsed terrestrial lidar scanners. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 437–442. IEEE, 2015.

[106] Wonjoo Kim, Wang Yibing, Ilia Ovsiannikov, SeungHoon Lee, Yoondong Park, Chilhee Chung, and Eric Fossum. A 1.5mpixel rgbz cmos image sensor for simultaneous color and range image capture. In *2012 IEEE International Solid-State Circuits Conference*, pages 392–394, 2012.

[107] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[108] Deyvid Kochanov, Fatemeh Karimi Nejadasl, and Olaf Booij. Kprnet: Improving projection-based lidar semantic segmentation, 2020. URL https://arxiv.org/abs/2007.12668.

[109] Lingdong Kong, Youquan Liu, Runnan Chen, Yuexin Ma, Xinge Zhu, Yikang Li, Yuenan Hou, Yu Qiao, and Ziwei Liu. Rethinking range view representation for lidar segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 228–240, October 2023.

[110] Lingdong Kong, Niamul Quader, and Venice Erin Liong. Conda: Unsupervised domain adaptation for lidar segmentation via regularized domain concatenation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 9338–9345, 2023. doi: 10.1109/ICRA48891.2023.10160410.

[111] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

[112] 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, 2018.

[113] Hongwu Kuang, Bei Wang, Jianping An, Ming Zhang, and Zehan Zhang. Voxel-fpn: Multi-scale voxel feature aggregation for 3d object detection from lidar point clouds. *Sensors*, 20(3), 2020.

[114] Meelis Kull, Telmo Silva Filho, and Peter Flach. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 623–631. PMLR, April 2017. URL https://proceedings.mlr.press/v54/kull17a.html.

[115] A Kurup and J Bos. The winter adverse driving dataset (wads). https://digitalcommons.mtu.edu/wads/, 2021. URL https://digitalcommons.mtu.edu/wads/.

[116] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8500–8509, June 2022.

[117] Xin Lai, Yukang Chen, Fanbin Lu, Jianhui Liu, and Jiaya Jia. Spherical transformer for lidar-based 3d recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17545–17555, June 2023.

[118] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[119] 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 (CVPR)*, June 2019.

[120] Ferdinand Langer, Andres Milioto, Alexandre Haag, Jens Behley, and Cyrill Stachniss. Domain transfer for semantic segmentation of lidar data using deep neural networks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8263–8270, 2020. doi: 10.1109/IROS45743.2020.9341508.

[121] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[122] Minjae Lee, Seongmin Park, Hyungmin Kim, Minyong Yoon, Janghwan Lee, Jun Won Choi, Nam Sung Kim, Mingu Kang, and Jungwook Choi. Spade: Sparse pillar-based 3d object detection accelerator for autonomous driving. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 454–467, 2024.

[123] Seungjae Lee, Hyungtae Lim, and Hyun Myung. Patchwork++: Fast and robust ground segmentation solving partial under-segmentation using 3d point cloud. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13276–13283, 2022. doi: 10.1109/IROS47612.2022.9981561.

[124] Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. xformers: A modular and hackable transformer modelling library. https://github.com/facebookresearch/xformers, 2022.

[125] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. In *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016.

[126] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[127] Jiale Li, Hang Dai, Hao Han, and Yong Ding. Mseg3d: Multi-modal 3d semantic segmentation for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21694–21704, 2023.

[128] Jianan Li, Shaocong Dong, Lihe Ding, and Tingfa Xu. Mssvt++: Mixed-scale sparse voxel transformer with center voting for 3d object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(5):3736–3752, 2024.

[129] Jinyu Li, Chenxu Luo, and Xiaodong Yang. PillarNeXt: Rethinking network designs for 3D object detection in LiDAR point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17567–17576, 2023.

[130] Nanxi Li, Chong Pei Ho, Jin Xue, Leh Woon Lim, Guanyu Chen, Yuan Hsing Fu, and Lennon Yao Ting Lee. A progress review on solid-state lidar and nanophotonics-based lidar sensors. *Laser & Photonics Reviews*, 16(11):2100511, 2022.

[131] Rong Li, Anh-Quan CAO, and Raoul de Charette. Class-prototypes for contrastive learning in weakly-supervised 3d point cloud segmentation. In *33rd British Machine Vision Conference 2022, BMVC 2022, London, UK, November 21-24, 2022*. BMVA Press, 2022.

[132] Xin Li, Tao Ma, Yuenan Hou, Botian Shi, Yuchen Yang, Youquan Liu, Xingjiao Wu, Qin Chen, Yikang Li, Yu Qiao, et al. Logonet: Towards accurate 3d object detection with local-to-global cross-modal fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17524–17534, 2023.

[133] Yingwei Li, Adams Wei Yu, Tianjian Meng, Ben Caine, Jiquan Ngiam, Daiyi Peng, Junyang Shen, Yifeng Lu, Denny Zhou, Quoc V Le, et al. Deepfusion: Lidar-camera deep fusion for multi-modal 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17182–17191, 2022.

[134] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[135] Zhidong Liang, Ming Zhang, Zehan Zhang, Xian Zhao, and Shiliang Pu. Rangercnn: Towards fast and accurate 3d object detection with range image representation, 2021. URL https://arxiv.org/abs/2009.00206.

[136] Zhidong Liang, Zehan Zhang, Ming Zhang, Xian Zhao, and Shiliang Pu. Rangeioudet: Range image based real-time 3d object detector optimized by intersection over union. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7140–7149, June 2021.

[137] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[138] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[139] Xipeng Lin, Shanshi Huang, and Hongwu Jiang. Voxel-cim: An efficient compute-in-memory accelerator for voxel-based point cloud neural networks, 2024. URL https://arxiv.org/abs/2409.19077.

[140] Yujun Lin, Zhekai Zhang, Haotian Tang, Hanrui Wang, and Song Han. Pointacc: Efficient point cloud accelerator. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, page 449–461, 2021.

[141] Jianhui Liu, Yukang Chen, Xiaoqing Ye, Zhuotao Tian, Xiao Tan, and Xiaojuan Qi. Spatial pruned sparse convolution for efficient 3d object detection. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 6735–6748. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/2ce10f144bb93449767f355c01f24cc1-Paper-Conference.pdf.

[142] Wei Liu, Zhiming Luo, Yuanzheng Cai, Ying Yu, Yang Ke, José Marcato Junior, Wesley Nunes Gonçalves, and Jonathan Li. Adversarial unsupervised domain adaptation for 3D semantic segmentation with multi-modal learning. *ISPRS Journal of Photogrammetry and Remote Sensing*, 176:211–221, 2021.

[143] Youquan Liu, Runnan Chen, Xin Li, Lingdong Kong, Yuchen Yang, Zhaoyang Xia, Yeqi Bai, Xinge Zhu, Yuexin Ma, Yikang Li, et al. Uniseg: A unified multi-modal lidar segmentation network and the openpcseg codebase. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 21662–21673, 2023.

[144] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, October 2021.

[145] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[146] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela L. Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2774–2781, 2023. doi: 10.1109/ICRA48891.2023.10160968.

[147] Zhijian Liu, Xinyu Yang, Haotian Tang, Shang Yang, and Song Han. Flatformer: Flattened window attention for efficient point cloud transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1200–1211, June 2023.

[148] Waymo LLC. Lidar - laser bear honeycomb – waymo. https://waymo.com/lidar, 2019. Accessed: 2019-07-06.

[149] Waymo LLC. Waymo one: The world's first autonomous ride-hailing service, 2024. URL https://waymo.com/waymo-one/. Accessed: 2024-10-09.

[150] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[151] Baraja Pty Ltd. *Spectrum HD25 Datasheet*. Baraja Pty Ltd., 2022. URL https://www.baraja.com/en/spectrum-hd25.

[152] Haifeng Luo, Kourosh Khoshelham, Lina Fang, and Chongcheng Chen. Unsupervised scene adaptation for semantic segmentation of urban mobile laser scanning point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 169:253–267, 2020.

[153] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[154] Xiaotong Luo, Yuan Xie, Yulun Zhang, Yanyun Qu, Cuihua Li, and Yun Fu. Latticenet: Towards lightweight image super-resolution with lattice block. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 272–289, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58542-6.

[155] Zhipeng Luo, Zhongang Cai, Changqing Zhou, Gongjie Zhang, Haiyu Zhao, Shuai Yi, Shijian Lu, Hongsheng Li, Shanghang Zhang, and Ziwei Liu. Unsupervised domain adaptive 3d detection with multi-level consistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8866–8875, October 2021.

[156] Zhipeng Luo, Gongjie Zhang, Changqing Zhou, Tianrui Liu, Shijian Lu, and Liang Pan. TransPillars: Coarse-to-fine aggregation for multi-frame 3D object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4230–4239, January 2023.

[157] Dongxu Lyu, Zhenyu Lil, Yuzhou Chen, Jinming Zhang, Ningyi Xu, and Guanghui He. Spocta: A 3d sparse convolution accelerator with octree-encoding-based map search and inherent sparsity-aware processing. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–9, 2023.

[158] Dongxu Lyu, Zhenyu Li, Yuzhou Chen, Gang Wang, Weifeng He, Ningyi Xu, and Guanghui He. Flna: Flexibly accelerating feature learning networks for large-scale point clouds with efficient dataflow decoupling. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 32(4):739–751, 2024.

[159] Tao Ma, Xuemeng Yang, Hongbin Zhou, Xin Li, Botian Shi, Junjie Liu, Yuchen Yang, Zhizheng Liu, Liang He, Yu Qiao, Yikang Li, and Hongsheng Li. Detzero: Rethinking offboard 3d object detection with long-term sequential point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6736–6747, October 2023.

[160] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual MLP framework. In *International Conference on Learning Representations*, 2022.

[161] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. Trafficpredict: trajectory prediction for heterogeneous traffic-agents. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI Press, 2019.

[162] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[163] Jiageng Mao, Minzhe Niu, Chenhan Jiang, Hanxue Liang, Jingheng Chen, Xiaodan Liang, Yamin Li, Chaoqiang Ye, Wei Zhang, Zhenguo Li, Jie Yu, Hang Xu, and Chunjing Xu. One million scenes for autonomous driving: Once dataset, 2021. URL https://arxiv.org/abs/2106.11037.

[164] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3164–3173, October 2021.

[165] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015.

[166] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[167] Gregory P. Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K. Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[168] Gregory P. Meyer, Jake Charland, Shreyash Pandey, Ankit Laddha, Shivam Gautam, Carlos Vallespi-Gonzalez, and Carl K. Wellington. Laserflow: Efficient and probabilistic object detection and motion forecasting. *IEEE Robotics and Automation Letters*, 6(2): 526–533, 2021.

[169] 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, 2019.

[170] Sambit Mohapatra, Senthil Yogamani, Heinrich Gotzig, Stefan Milz, and Patrick Mader. Bevdetnet: Bird's eye view lidar point cloud based real-time 3d object detection for autonomous driving. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 2809–2815, 2021.

[171] General Motors. Key to the future, 1956.

[172] Alexey Nekrasov, Jonas Schult, Or Litany, Bastian Leibe, and Francis Engelmann. Mix3d: Out-of-context data augmentation for 3d scenes. In *2021 International Conference on 3D Vision (3DV)*, pages 116–125, 2021.

[173] Om Ji Omer, Prashant Laddha, Gurpreet S Kalsi, Anirud Thyagharajan, Kamlesh R Pillai, Abhimanyu Kulkarni, Anbang Yao, Yurong Chen, and Sreenivas Subramoney. Accss3d:

Accelerator for spatially sparse 3d dnns, 2020. URL https://arxiv.org/abs/2011.12669.

[174] Dhruv Parikh, Bingyi Zhang, Rajgopal Kannan, Viktor Prasanna, and Carl Busart. Performance of graph neural networks for point cloud applications. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–7, 2023.

[175] Seongmin Park, Minjae Lee, Junwon Choi, and Jungwook Choi. Selectively dilated convolution for accuracy-preserving sparse pillar-based embedded 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 8104–8113, June 2024.

[176] Abhishek Patil, Srikanth Malla, Haiming Gang, and Yi-Ting Chen. The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9552–9557, 2019.

[177] Shuwen Peng, Xiaohuan Xi, Cheng Wang, Rongchang Xie, Pu Wang, and Hongwu Tan. Point-based multilevel domain adaptation for point cloud segmentation. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2022. doi: 10.1109/LGRS.2020.3037702.

[178] Jonathan Petit, Bas Stottelaar, Michael Feiri, and Frank Kargl. Remote attacks on automated vehicles sensors: Experiments on camera and lidar. *Black Hat Europe*, 11(2015): 995, 2015.

[179] Quang-Hieu Pham, Pierre Sevestre, Ramanpreet Singh Pahwa, Huijing Zhan, Chun Ho Pang, Yuda Chen, Armin Mustafa, Vijay Chandrasekhar, and Jie Lin. A*3d dataset: Towards autonomous driving in challenging environments, 2019. URL https://arxiv.org/abs/1909.07541.

[180] Quang-Hieu Pham, Pierre Sevestre, Ramanpreet Singh Pahwa, Huijing Zhan, Chun Ho Pang, Yuda Chen, Armin Mustafa, Vijay Chandrasekhar, and Jie Lin. A*3d dataset: Towards autonomous driving in challenging environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2267–2273, 2020. doi: 10.1109/ICRA40945.2020.9197385.

[181] Matthew Pitropov, Danson Evan Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, and Steven Waslander. Canadian adverse driving conditions dataset. *The International Journal of Robotics Research*, 40(4-5):681–690, 2021.

[182] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.

[183] CIBBY PULIKKASERIL. *Baraja Spectrum-Scan White Paper*. Baraja Pty Ltd., 2021. URL https://www.baraja.com/en/technology/white-paper.

[184] Charles R. Qi, Hao Su, Matthias Niessner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[185] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[186] 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 (CVPR)*, June 2018.

[187] Charles R. Qi, Yin Zhou, Mahyar Najibi, Pei Sun, Khoa Vo, Boyang Deng, and Dragomir Anguelov. Offboard 3d object detection from point cloud sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6134–6144, June 2021.

[188] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[189] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. In *Advances in Neural Information Processing Systems*, volume 35, pages 23192–23204, 2022.

[190] Shoumeng Qiu, Xinrun Li, XiangYang Xue, and Jian Pu. Pc-bev: An efficient polar-cartesian bev fusion framework for lidar semantic segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

[191] Huilin Qu and Loukas Gouskos. Jet tagging via particle clouds. *Physical Review D*, 101 (5):056019, 2020.

[192] Niranjan Ravi and Mohamed El-Sharkawy. Addressing the gaps of iou loss in 3d object detection with iiou. *Future Internet*, 15(12), 2023.

[193] Ryan Razani, Ran Cheng, Ehsan Taghavi, and Liu Bingbing. Lite-hdseg: Lidar semantic segmentation using lite harmonic dense convolutions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9550–9556, 2021.

[194] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[195] Christoph B. Rist, Markus Enzweiler, and Dariu M. Gavrila. Cross-sensor deep domain adaptation for lidar detection and segmentation. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1535–1542, 2019. doi: 10.1109/IVS.2019.8814047.

[196] RobotShop. Ouster os2 128 rev 7 long range lidar sensor, 2024. URL https://eu.robotshop.com/products/general-laser-og-os2-128-rev-7. Accessed: 2024-10-09.

[197] Mrigank Rochan, Shubhra Aich, Eduardo R. Corral-Soto, Amir Nabatchian, and Bingbing Liu. Unsupervised domain adaptation in lidar semantic segmentation with self-supervision and gated adapters. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2649–2655, 2022. doi: 10.1109/ICRA46639.2022.9811654.

[198] Mrigank Rochan, Xingxin Chen, Alaap Grandhi, Eduardo R. Corral-Soto, and Bingbing Liu. Domain adaptation in 3D object detection with gradual batch alternation training, 2022. URL https://arxiv.org/abs/2210.10180. arXiv: 2210.10180.

[199] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, 2015.

[200] Khaled Saleh, Ahmed Abobakr, Mohammed Attia, Julie Iskander, Darius Nahavandi, Mohammed Hossny, and Saeid Nahvandi. Domain adaptation for vehicle detection from bird's eye view lidar point cloud data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, October 2019.

[201] Cristiano Saltori, Stéphane Lathuilière, Nicu Sebe, Elisa Ricci, and Fabio Galasso. Sf-uda3d: Source-free unsupervised domain adaptation for lidar-based 3d object detection. In *International Conference on 3D Vision (3DV)*, pages 771–780, 2020. doi: 10.1109/3DV50981.2020.00087.

[202] Peer Schutt, Radu Alexandru Rosu, and Sven Behnke. Abstract flow for temporal semantic segmentation on the permutohedral lattice. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5139–5145, 2022. doi: 10.1109/ICRA46639.2022.9811818.

[203] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision, 2024. URL https://arxiv.org/abs/2407.08608.

[204] Hualian Sheng, Sijia Cai, Na Zhao, Bing Deng, Jianqiang Huang, Xian-Sheng Hua, Min-Jian Zhao, and Gim Hee Lee. Rethinking iou-based optimization for single-stage 3d object detection. In *Computer Vision – ECCV 2022*, pages 544–561, 2022.

[205] Hanyu Shi, Guosheng Lin, Hao Wang, Tzu-Yi Hung, and Zhenhua Wang. Spsequencenet: Semantic segmentation network on 4d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4574–4583, 2020.

[206] 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 (CVPR)*, June 2019.

[207] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[208] 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, 2021.

[209] 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. *International Journal of Computer Vision*, 131(2):531–551, February 2023.

[210] Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[211] Hocheol Shin, Dohyun Kim, Yujin Kwon, and Yongdae Kim. Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, pages 445–467, Cham, 2017. Springer International Publishing. ISBN 978-3-319-66787-4.

[212] Inc. SiLC Technologies. Fmcw and long range lidar machine vision solution, 2024. URL https://www.silc.com/product/. Accessed: 2024-10-09.

[213] Martin Simony, Stefan Milzy, Karl Amendey, and Horst-Michael Gross. Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[214] Santokh Singh. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. Technical report, National Highway Traffic Safety Administration, 2015.

[215] Leslie N. Smith and Nicholay Topin. Super-convergence: very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, page 1100612, 2019.

[216] Roman Solovyev, Weimin Wang, and Tatiana Gabruseva. Weighted boxes fusion: Ensembling boxes from different object detection models. *Image and Vision Computing*, 107: 104117, 2021. ISSN 0262-8856. doi: https://doi.org/10.1016/j.imavis.2021.104117.

[217] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[218] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[219] Pei Sun, Weiyue Wang, Yuning Chai, Gamaleldin Elsayed, Alex Bewley, Xiao Zhang, Cristian Sminchisescu, and Dragomir Anguelov. Rsn: Range sparse net for efficient, accurate lidar 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5725–5734, June 2021.

[220] Pei Sun, Mingxing Tan, Weiyue Wang, Chenxi Liu, Fei Xia, Zhaoqi Leng, and Dragomir Anguelov. Swformer: Sparse window transformer for 3d object detection in point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 426–442, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-20080-9.

[221] N. Takeuchi, N. Sugimoto, H. Baba, and K. Sakurai. Random modulation cw lidar. *Appl. Opt.*, 22(9):1382–1386, May 1983.

[222] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *Computer Vision – ECCV 2020*, pages 685–702, 2020.

[223] Haotian Tang, Zhijian Liu, Xiuyu Li, Yujun Lin, and Song Han. TorchSparse: Efficient Point Cloud Inference Engine. In *Conference on Machine Learning and Systems (MLSys)*, 2022.

[224] Haotian Tang, Shang Yang, Zhijian Liu, Ke Hong, Zhongming Yu, Xiuyu Li, Guohao Dai, Yu Wang, and Song Han. Torchsparse++: Efficient training and inference framework for sparse convolution on gpus. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2023.

[225] Mei Qi Tang, Sean Sedwards, Chengjie Huang, and Krzysztof Czarnecki. How hard is snow? a paired domain adaptation dataset for clear and snowy weather: Cadc+, 2025.

[226] Vladimir Tankovich, Christian Hane, Yinda Zhang, Adarsh Kowdle, Sean Fanello, and Sofien Bouaziz. Hitnet: Hierarchical iterative tile refinement network for real-time stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14362–14372, June 2021.

[227] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[228] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020.

[229] Hesai Technology. Global leader in lidar sensor solutions — hesai technology. https://www.hesaitech.com/, 2024. Accessed: 2024-10-09.

[230] Hesai Technology. At128 auto-grade 120° long-range lidar, 2024. URL https://www.hesaitech.com/product/at128/. Accessed: 2024-10-09.

[231] Arulmolivarman Thieshanthan, Amashi Niwarthana, Pamuditha Somarathne, Tharindu Wickremasinghe, and Ranga Rodrigo. Hpgnn: Using hierarchical graph neural networks for outdoor point cloud processing. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 2700–2706, 2022.

[232] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[233] Hugues Thomas, Yao-Hung Hubert Tsai, Timothy D. Barfoot, and Jian Zhang. Kpconvx: Modernizing kernel point convolution with kernel attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5525–5535, June 2024.

[234] C. Thorpe, M.H. Hebert, T. Kanade, and S.A. Shafer. Vision and navigation for the carnegie-mellon navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362–373, 1988. doi: 10.1109/34.3900.

[235] C. Thorpe, M. Herbert, T. Kanade, and S. Shafer. Toward autonomous driving: the cmu navlab. i. perception. *IEEE Expert*, 6(4):31–42, 1991. doi: 10.1109/64.85919.

[236] Zhi Tian, Xiangxiang Chu, Xiaoming Wang, Xiaolin Wei, and Chunhua Shen. Fully convolutional one-stage 3d object detection on lidar range images. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 34899–34911. Curran Associates, Inc., 2022.

[237] Larissa T. Triess, Mariella Dreissig, Christoph B. Rist, and J. Marius Zöllner. A survey on deep domain adaptation for lidar perception. In *IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, pages 350–357, 2021. doi: 10.1109/IVWorkshops54471.2021.9669228.

[238] Darren Tsai, Julie Stephany Berrio, Mao Shan, Stewart Worrall, and Eduardo Nebot. See eye to eye: A LiDAR-agnostic 3D detection framework for unsupervised multi-target domain adaptation. *IEEE Robotics and Automation Letters*, 7(3):7904–7911, 2022.

[239] Darren Tsai, Julie Stephany Berrio, Mao Shan, Eduardo Nebot, and Stewart Worrall. Viewer-centred surface completion for unsupervised domain adaptation in 3d object detection. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 9346–9353, 2023. doi: 10.1109/ICRA48891.2023.10160707.

[240] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *2017 International Conference on 3D Vision (3DV)*, pages 11–20, 2017. doi: 10.1109/3DV.2017.00012.

[241] Tesla Software Update. Tesla started a new software update 2023.44.30.10 (fsd beta v12.1), 2023. URL https://x.com/TeslaSoftUpdate/status/1738865445090767001. Accessed: 2024-10-09.

[242] U.S. Department of Transportation, National Highway Traffic Safety Administration (NHTSA). National motor vehicle crash causation survey, report to congress. https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811059, 2008.

[243] U.S. Department of Transportation, National Highway Traffic Safety Administration (NHTSA). Overview of motor vehicle traffic crashes in 2022. *Traffic safety facts: research note*, pages 1–18, June 2024.

[244] Vladimir N Vapnik. *The Nature of Statistical Learning Theory*. Springer New York, NY, 1999.

[245] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[246] Inc. Velodyne Lidar. *Velodyne HDL-64E Datasheet*. Velodyne Lidar, Inc., 2015.

[247] Inc. Velodyne Lidar. Envision the future — velodyne lidar. https://velodynelidar.com/, 2023. Accessed: 2023-01-30.

[248] Devesh Walawalkar, Zhiqiang Shen, Zechun Liu, and Marios Savvides. Attentive cutmix: An enhanced data augmentation approach for deep learning based image classification, 2020. URL https://arxiv.org/abs/2003.13048.

[249] Sean Walsh, Jason Ku, Alex D. Pon, and Steven L. Waslander. Leveraging temporal data for automatic labelling of static vehicles. In *2020 17th Conference on Computer and Robot Vision (CRV)*, pages 134–141, 2020. doi: 10.1109/CRV50864.2020.00026.

[250] Cheng Wang, Ming Cheng, Ferdous Sohel, Mohammed Bennamoun, and Jonathan Li. Normalnet: A voxel-based cnn for 3d object classification and retrieval. *Neurocomputing*, 323:139–147, 2019.

[251] Chu Wang, Marcello Pelillo, and Kaleem Siddiqi. Dominant set clustering and pooling for multi-view 3d object recognition. In *British Machine Vision Conference 2017, BMVC 2017, London, UK, September 4-7, 2017*. BMVA Press, 2017.

[252] Haiyang Wang, Chen Shi, Shaoshuai Shi, Meng Lei, Sen Wang, Di He, Bernt Schiele, and Liwei Wang. Dsvt: Dynamic sparse voxel transformer with rotated sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13520–13529, June 2023.

[253] Peng-Shuai Wang. Octformer: Octree-based transformers for 3d point clouds. *ACM Trans. Graph.*, 42(4), July 2023.

[254] Song Wang, Jianke Zhu, and Ruixiang Zhang. Meta-rangeseg: Lidar sequence semantic segmentation using multiple feature aggregation. *IEEE Robotics and Automation Letters*, 7(4):9739–9746, 2022.

[255] Tianyu Wang, Xiaowei Hu, Zhengzhe LIU, and Chi-Wing Fu. Sparse2dense: Learning to densify 3d features for 3d object detection. In *Advances in Neural Information Processing Systems*, volume 35, pages 38533–38545. Curran Associates, Inc., 2022.

[256] Yan Wang, Xiangyu Chen, Yurong You, Li Erran Li, Bharath Hariharan, Mark Campbell, Kilian Q. Weinberger, and Wei-Lun Chao. Train in germany, test in the usa: Making 3d object detectors generalize. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[257] Yan Wang, Junbo Yin, Wei Li, Pascal Frossard, Ruigang Yang, and Jianbing Shen. Ssda3d: Semi-supervised domain adaptation for 3d object detection from point cloud. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2707–2715, 2023.

[258] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.*, 38(5), October 2019.

[259] Zilun Wang, Wendong Mao, Peixiang Yang, Zhongfeng Wang, and Jun Lin. An efficient fpga accelerator for point cloud. In *2022 IEEE 35th International System-on-Chip Conference (SOCC)*, pages 1–6, 2022.

[260] Xin Wei, Ruixuan Yu, and Jian Sun. View-gcn: View-based graph convolutional network for 3d shape analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[261] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*, 2021.

[262] Benjamin Wilson, Nicholas Autio Mitchell, Jhony Kaesemodel Pontes, and James Hays. What matters in range view 3d object detection. In *8th Annual Conference on Robot Learning*, 2024.

[263] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893, 2018.

[264] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4376–4382, 2019.

[265] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[266] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 33330–33342. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/d78ece6613953f46501b958b7bb4582f-Paper-Conference.pdf.

[267] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4840–4851, June 2024.

[268] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[269] Aoran Xiao, Jiaxing Huang, Dayan Guan, Kaiwen Cui, Shijian Lu, and Ling Shao. Polarmix: A general data augmentation technique for lidar point clouds. In *Advances in Neural Information Processing Systems*, volume 35, pages 11035–11048, 2022.

[270] Aoran Xiao, Jiaxing Huang, Weihao Xuan, Ruijie Ren, Kangcheng Liu, Dayan Guan, Abdulmotaleb El Saddik, Shijian Lu, and Eric P. Xing. 3d semantic segmentation in the wild: Learning generalized models for adverse-condition point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9382–9392, June 2023.

[271] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, Yunlong Wang, and Diange Yang. Pandaset: Advanced sensor suite dataset for autonomous driving. In *IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3095–3101, 2021. doi: 10.1109/ITSC48978.2021.9565009.

[272] Shimin Xiong, Bin Li, and Shiao Zhu. Dcgnn: A single-stage 3d object detection network based on density clustering and graph neural network. *Complex & Intelligent Systems*, 9 (3):3399–3408, 2023.

[273] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 1–19, Cham, 2020. Springer International Publishing.

[274] Jianyun Xu, Ruixiang Zhang, Jian Dou, Yushi Zhu, Jie Sun, and Shiliang Pu. Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16024–16033, October 2021.

[275] Jinglin Xu, Xiangsen Zhang, Wenbin Li, Xinwang Liu, and Junwei Han. Joint multi-view 2d convolutional neural networks for 3d object classification. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 2021.

[276] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3173–3182, June 2021.

[277] Qiangeng Xu, Yin Zhou, Weiyue Wang, Charles R. Qi, and Dragomir Anguelov. Spg: Unsupervised domain adaptation for 3d object detection via semantic point generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15446–15456, October 2021.

[278] Xu Yan, Jiantao Gao, Jie Li, Ruimao Zhang, Zhen Li, Rui Huang, and Shuguang Cui. Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3101–3109, 2021.

[279] Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui, and Zhen Li. 2dpass: 2d priors assisted semantic segmentation on lidar point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 677–695, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-19815-1.

[280] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely embedded convolutional detection. *Sensors*, 18(10), 2018. ISSN 1424-8220. doi: 10.3390/s18103337.

[281] Anqi Joyce Yang, Sergio Casas, Nikita Dvornik, Sean Segal, Yuwen Xiong, Jordan Sir Kwang Hu, Carter Fang, and Raquel Urtasun. Labelformer: Object trajectory refinement for offboard perception from lidar point clouds. In *Conference on Robot Learning*, pages 3364–3383. PMLR, 2023.

[282] Bin Yang, Ming Liang, and Raquel Urtasun. Hdnet: Exploiting hd maps for 3d object detection. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 146–155. PMLR, 29–31 Oct 2018.

[283] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[284] Bin Yang, Min Bai, Ming Liang, Wenyuan Zeng, and Raquel Urtasun. Auto4d: Learning to label 4d objects from sequential point clouds, 2021. URL https://arxiv.org/abs/2101.06586.

[285] Jiacheng Yang, Christina Giannoula, Jun Wu, Mostafa Elhoushi, James Gleeson, and Gennady Pekhimenko. Minuet: Accelerating 3d sparse convolutions on gpus. In *Proceedings of the Nineteenth European Conference on Computer Systems*, page 786–802, 2024.

[286] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. St3d: Self-training for unsupervised domain adaptation on 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10368–10378, June 2021.

[287] Jihan Yang, Shaoshuai Shi, Runyu Ding, Zhe Wang, and Xiaojuan Qi. Towards efficient 3D object detection with knowledge distillation. In *Advances in Neural Information Processing Systems*, volume 35, pages 21300–21313. Curran Associates, Inc., 2022.

[288] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. St3d++: Denoised self-training for unsupervised domain adaptation on 3d object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):6354–6371, 2023. doi: 10.1109/TPAMI.2022.3216606.

[289] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10371–10381, June 2024.

[290] Xinhao Yang, Tianyu Fu, Guohao Dai, Shulin Zeng, Kai Zhong, Ke Hong, and Yu Wang. An efficient accelerator for point-based and voxel-based point cloud neural networks. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2023.

[291] Ze Yang and Liwei Wang. Learning relationships for multi-view 3d object recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[292] Zetong Yang, Yin Zhou, Zhifeng Chen, and Jiquan Ngiam. 3d-man: 3d multi-frame attention network for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1863–1872, June 2021.

[293] Dongqiangzi Ye, Zixiang Zhou, Weijia Chen, Yufei Xie, Yu Wang, Panqu Wang, and Hassan Foroosh. Lidarmultinet: Towards a unified multi-task network for lidar perception. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3231–3240, 2023.

[294] Maosheng Ye, Shuangjie Xu, and Tongyi Cao. Hvnet: Hybrid voxel network for lidar based 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[295] Li Yi, Boqing Gong, and Thomas Funkhouser. Complete & label: A domain adaptation approach to semantic segmentation of lidar point clouds. In *Proceedings of the IEEE/CVF*

*Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15363–15373, June 2021.

[296] Zeng Yihan, Chunwei Wang, Yunbo Wang, Hang Xu, Chaoqiang Ye, Zhen Yang, and Chao Ma. Learning transferable features for point cloud detection via 3D contrastive co-training. In *Advances in Neural Information Processing Systems*, volume 34, pages 21493–21504. Curran Associates, Inc., 2021.

[297] Junbo Yin, Jianbing Shen, Chenye Guan, Dingfu Zhou, and Ruigang Yang. LiDAR-based online 3D video object detection with graph-based message passing and spatiotemporal transformer attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[298] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11784–11793, June 2021.

[299] Yurong You, Carlos Andres Diaz-Ruiz, Yan Wang, Wei-Lun Chao, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Exploiting playbacks in unsupervised domain adaptation for 3d object detection in self-driving cars. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5070–5077, 2022. doi: 10.1109/ICRA46639.2022.9811722.

[300] Tan Yu, Jingjing Meng, and Junsong Yuan. Multi-view harmonized bilinear network for 3d object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[301] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[302] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[303] Jinlai Zhang, Lyujie Chen, Bo Ouyang, Binbin Liu, Jihong Zhu, Yujin Chen, Yanmei Meng, and Danfeng Wu. Pointcutmix: Regularization strategy for point cloud classification. *Neurocomputing*, 505:58–67, 2022.

[304] Lunjun Zhang, Anqi Joyce Yang, Yuwen Xiong, Sergio Casas, Bin Yang, Mengye Ren, and Raquel Urtasun. Towards unsupervised object detection from lidar point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9317–9328, June 2023.

[305] Weichen Zhang, Wen Li, and Dong Xu. Srdan: Scale-aware and range-aware domain adaptation network for cross-dataset 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6769–6779, June 2021.

[306] Weishuang Zhang, B Ravi Kiran, Thomas Gauthier, Yanis Mazouz, and Theo Steger. Simulation-to-reality domain adaptation for offline 3D object annotation on pointclouds with correlation alignment. In *International Conference on Image Processing and Vision Engineering*, 2022.

[307] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[308] Yi Zhang, Zhiyu Xiang, Chengyu Qiao, and Shuya Chen. Accurate and real-time object detection based on bird's eye view on 3d point clouds. In *2019 International Conference on 3D Vision (3DV)*, pages 214–221, 2019.

[309] Yifan Zhang, Qingyong Hu, Guoquan Xu, Yanxin Ma, Jianwei Wan, and Yulan Guo. Not all points are equal: Learning highly efficient point-based detectors for 3D LiDAR point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18953–18962, June 2022.

[310] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[311] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H.S. Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16259–16268, October 2021.

[312] Sicheng Zhao, Yezhen Wang, Bo Li, Bichen Wu, Yang Gao, Pengfei Xu, Trevor Darrell, and Kurt Keutzer. epointda: An end-to-end simulation-to-real domain adaptation framework for lidar point cloud segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3500–3509, 2021.

[313] Yiming Zhao, Lin Bai, and Xinming Huang. Fidnet: Lidar point cloud semantic segmentation with fully interpolation decoding. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4453–4458, 2021.

[314] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-iou loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI conference on artificial intelligence*, pages 12993–13000, 2020.

[315] Chao Zhou, Yanan Zhang, Jiaxin Chen, and Di Huang. Octr: Octree-based transformer for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5166–5175, June 2023.

[316] Xingcheng Zhou, Mingyu Liu, Ekim Yurtsever, Bare Luka Zagar, Walter Zimmer, Hu Cao, and Alois C Knoll. Vision language models in autonomous driving: A survey and outlook. *IEEE Transactions on Intelligent Vehicles*, 2024.

[317] 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 (CVPR)*, June 2018.

[318] Zixiang Zhou, Yang Zhang, and Hassan Foroosh. Panoptic-polarnet: Proposal-free lidar point cloud panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13194–13203, June 2021.

[319] Zixiang Zhou, Xiangchen Zhao, Yu Wang, Panqu Wang, and Hassan Foroosh. Centerformer: Center-based transformer for 3d object detection. In *Computer Vision – ECCV 2022*, pages 496–513, 2022.

[320] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection, 2019. URL https://arxiv.org/abs/1908.09492.

[321] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9939–9948, June 2021.