Efficient Person Detection with LiDAR Sensors for Mobile Robots

Von der Fakultät für Informatik der RWTH Aachen University
zur Erlangung des akademischen Grades

eines Doktors der Naturwissenschaften

genehmigte Dissertation

vorgelegt von

Dan Jia, M.Sc.

Berichter:
Prof. Dr. Bastian Leibe
Prof. Dr. Kai Arras
Prof. Dr. Peter Rossmanith

Tag der mündlichen Prüfung: 11.07.2025

Abstract

In the era of surging robotic applications, robots are becoming increasingly mobile and ubiquitous, participating in diverse tasks ranging from household chores to navigating complex urban environments. This surge is attributed to advances in sensing, computing, and artificial intelligence, leading to the development of sophisticated autonomous systems. In contrast to industrial robots designed for factory automation, mobile robots are designed to operate in complex and unstructured environments, necessitating advanced scene understanding capabilities for safe interactions. One such capability is person detection, which is the focus of this thesis. While mature solutions exist for detecting persons using RGB(-D) cameras and deep learning-based object detectors, these come with limitations such as restricted field of view, increased computation with multi-camera setups, and privacy concerns. LiDAR sensors offer an alternative with advantages including a wide field of view, accurate range measurement, and reduced privacy intrusion. However, efficiently detecting persons with LiDAR sensors presents challenges due to the unique data structure of point clouds and the sparsity of the range scan.

This thesis proposes novel techniques to overcome these challenges and design efficient person detection algorithms for LiDAR sensors. The first contribution is DR-SPAAM, a fast person detector for 2D LiDAR sensors on mobile robots. The key component of DR-SPAAM is a novel SPatial Attention and Auto-regressive Module, which learns to associate and aggregate measurements from consecutive scans, addressing challenges from the sparse point cloud with minimal computational overhead. DR-SPAAM establishes a new state-of-the-art on the DROW dataset, reaching 70.3% AP. In addition, it has a high inference rate of 87.2 FPS on a laptop with a dedicated GPU or 22.6 FPS on an NVIDIA Jetson AGX with an embedded GPU. Further experiments show that pretrained DR-SPAAM is robust against changes in sensor specifications such as sampling resolution and rate, making it easy to deploy on platforms that are equipped with different sensors. The high inference rate and the robustness against changing sensor specifications make DR-SPAAM well-suited for mobile robotic applications.

Contrary to the abundance of labeled images, only a few datasets exist for training 2D-LiDAR-based person detectors such as DR-SPAAM. This limitation in data may cause the network to focus on false correlations that are only present in training data, leading to poor detection generalization during deployment. To mitigate this problem, we propose an approach that automatically generates training labels, called pseudo-labels, for the 2D LiDAR sensor, using a calibrated camera with an off-the-shelf image-based detector. We conduct experiments to confirm that training or fine-tuning a detector with these pseudo-labels indeed improves its performance during deployment, and the performance can be further improved with robust training techniques such as mixup regularization and partially Huberized cross-entropy loss. With the proposed approach, a mobile robot can fine-tune its person detector during deployment, with no additional manual labeling cost.

Despite their effectiveness and affordability, 2D LiDAR sensors are limited to measuring a single plane. In contrast, 3D LiDAR sensors scan multiple planes at various heights due to their additional rotation angle, but they come with higher costs. This difference presents a trade-off that should be considered when selecting sensors for designing new robots. We conduct experiments to understand the performance gap between these two types of LiDAR sensors for detecting persons, aiming to facilitate informed decisions on sensor selection. We use state-of-the-art CenterPoint and DR-SPAAM person detectors as proxies for 3D and 2D LiDAR sensors, respectively, and compare their performance in multiple aspects, including detection accuracy, inference speed, localization performance, and robustness. Our results show that 2D LiDAR sensors can detect visible persons as accurately as 3D LiDAR sensors. However, it is much easier for persons to be fully occluded in the single-plane 2D LiDAR scan, making detection impossible. For applications such as collision avoidance, where detecting nearby visible persons is sufficient, 2D LiDAR sensors are a good choice due to their high inference speed and low cost. However, for applications requiring reliable person detection over an extended range, the more expensive 3D LiDAR sensors should be used.

The permutation equivariant attention mechanism, originally proposed for language tasks, is a promising operation for processing LiDAR scans. However, the quadratic space complexity with respect to the number of points forbids its application on large point clouds. To address this issue, we propose a memory-efficient variant of the attention mechanism, called Global Hierarchical Attention (GHA), which approximates the regular attention by computing local attention on a series of resolution hierarchies. GHA offers two key advantages. First, it has linear complexity relative to the number of points, allowing it to process large point clouds. Second, GHA inherently biases towards spatially close points, giving more attention to local structures while maintaining global connectivity among all points. Combined with a feedforward network, GHA can be inserted into many existing network architectures. Our experiments with multiple baseline networks show that adding GHA consistently improves performance across various tasks and datasets. For the task of semantic segmentation, GHA gives a +1.7% mIoU increase to the MinkowskiEngine baseline on ScanNet. For the 3D object detection task, GHA improves the CenterPoint baseline by +0.5% mAP on the nuScenes dataset, and the 3DETR baseline by +2.1% mAP_{25} and +1.5% mAP_{50} on ScanNet.

Zusammenfassung

In der Ära der zunehmenden Robotikanwendungen werden Roboter immer mobiler und allgegenwärtiger und übernehmen verschiedenste Aufgaben, von der Hausarbeit bis zur Navigation in komplexen städtischen Umgebungen. Dieser Anstieg ist auf Fortschritte in den Bereichen Sensorik, Computertechnik und künstliche Intelligenz zurückzuführen, die zur Entwicklung hochentwickelter autonomer Systeme geführt haben. Im Gegensatz zu Industrierobotern, die für die Automatisierung von Fabriken entwickelt wurden, sind mobile Roboter für den Einsatz in komplexen und unstrukturierten Umgebungen konzipiert, die für eine sichere Interaktion fortschrittliche Fähigkeiten zum Verstehen von Szenen erfordern. Eine dieser Fähigkeiten ist die Personenerkennung, die im Mittelpunkt dieser Arbeit steht. Es gibt zwar ausgereifte Lösungen für die Erkennung von Personen mit Hilfe von RGB(-D)-Kameras und Deep-Learning-basierten Objektdetektoren, doch sind diese mit Einschränkungen verbunden, wie z. B. einem eingeschränkten Sichtfeld, einem erhöhten Rechenaufwand bei der Einrichtung mehrerer Kameras und Bedenken hinsichtlich der Privatsphäre. LiDAR-Sensoren bieten eine Alternative mit Vorteilen wie einem großen Sichtfeld, einer genauen Entfernungsmessung und einer geringeren Beeinträchtigung der Privatsphäre. Die effiziente Erkennung von Personen mit LiDAR-Sensoren ist jedoch aufgrund der einzigartigen Datenstruktur von Punktwolken und der spärlichen Entfernungsmessung eine Herausforderung.

In dieser Arbeit werden neue Techniken vorgeschlagen, um diese Herausforderungen zu überwinden und effiziente Algorithmen zur Personenerkennung für LiDAR-Sensoren zu entwickeln. Der erste Beitrag ist DR-SPAAM, ein schneller Personendetektor für 2D-LiDAR-Sensoren auf mobilen Robotern. Die Schlüsselkomponente von DR-SPAAM ist ein neuartiges SPatial Attention and Auto-regressive Module (räumliches Aufmerksamkeits- und autoregressives Modul), das lernt, Messungen aus aufeinanderfolgenden Scans zu assoziieren und zu aggregieren, um die Herausforderungen einer dünnen Punktwolke mit minimalem Rechenaufwand zu bewältigen. DR-SPAAM stellt mit einer AP von 70,3% einen neuen Spitzenwert im DROW-Datensatz auf. Darüber hinaus hat es eine hohe Inferenzrate von 87,2

FPS auf einem Laptop mit dediziertem Grafikprozessor oder 22,6 FPS auf einem NVIDIA Jetson AGX mit integriertem Grafikprozessor. Weitere Experimente zeigen, dass das vortrainierte DR-SPAAM robust gegenüber Änderungen der Sensorspezifikationen wie Abtastauflösung und -rate ist, was den Einsatz auf Plattformen, die mit unterschiedlichen Sensoren ausgestattet sind, erleichtert. Durch die hohe Inferenzrate und die Robustheit gegenüber sich ändernden Sensorspezifikationen ist DR-SPAAM für mobile Roboteranwendungen gut geeignet.

Im Gegensatz zu der Fülle an gelabelten Bildern gibt es nur wenige Datensätze für das Training von 2D-LiDAR-basierten Personendetektoren wie DR-SPAAM. Diese Datenbeschränkung kann dazu führen, dass sich das Netzwerk auf falsche Korrelationen konzentriert, die nur in den Trainingsdaten vorhanden sind, was zu einer schlechten Generalisierung der Erkennung während des Einsatzes führt. Um dieses Problem zu entschärfen, schlagen wir einen Ansatz vor, der automatisch Trainingslabels, so genannte Pseudolabels, für den 2D-LiDAR-Sensor generiert, indem eine kalibrierte Kamera mit einem handelsüblichen bildbasierten Detektor verwendet wird. Wir führen Experimente durch, um zu bestätigen, dass das Training oder die Feinabstimmung eines Detektors mit diesen Pseudo-Labels in der Tat seine Leistung während des Einsatzes verbessert, und die Leistung kann mit robusten Trainingstechniken wie Mixup Regularization und Partially Huberized Cross-Entropy Loss weiter verbessert werden. Mit dem vorgeschlagenen Ansatz kann ein mobiler Roboter seinen Personendetektor während des Einsatzes feinabstimmen, ohne dass zusätzliche Kosten für die manuelle Kennzeichnung anfallen.

Trotz ihrer Effektivität und Erschwinglichkeit sind 2D-LiDAR-Sensoren auf die Messung einer einzigen Ebene beschränkt. Im Gegensatz dazu scannen 3D-LiDAR-Sensoren aufgrund ihres zusätzlichen Drehwinkels mehrere Ebenen in verschiedenen Höhen, sind aber mit höheren Kosten verbunden. Dieser Unterschied stellt einen Kompromiss dar, der bei der Auswahl von Sensoren für die Entwicklung neuer Roboter berücksichtigt werden sollte. Wir führen Experimente durch, um den Leistungsunterschied zwischen diesen beiden Arten von LiDAR-Sensoren für die Erkennung von Personen zu verstehen, mit dem Ziel, fundierte Entscheidungen bei der Sensorauswahl zu erleichtern. Wir verwenden die hochmodernen CenterPoint- und DR-SPAAM-Personendetektoren als Stellvertreter für 3D- bzw. 2D-LiDAR-Sensoren und vergleichen ihre Leistung in Bezug auf mehrere Aspekte, darunter Erkennungsgenauigkeit, Inferenzgeschwindigkeit, Lokalisierungsleistung und Robustheit. Unsere Ergebnisse zeigen, dass 2D-LiDAR-Sensoren sichtbare Personen genauso genau erkennen können wie 3D-LiDAR-Sensoren. Es ist jedoch viel einfacher, dass Personen in einem 2D-LiDAR-Scan mit nur einer Ebene vollständig verdeckt sind, was eine Erkennung unmöglich macht. Für Anwendungen wie die Kollisionsvermeidung, bei denen es ausreicht, in der Nähe befindliche sichtbare Personen zu erkennen, sind 2D-LiDAR-Sensoren aufgrund ihrer hohen Inferenzgeschwindigkeit und ihrer geringen Kosten eine gute Wahl. Für Anwendungen, die eine zuverlässige Personenerkennung über einen größeren Bereich erfordern, sollten jedoch die teureren 3D-LiDAR-Sensoren verwendet werden.

Der permutationsäquivariante Aufmerksamkeitsmechanismus, der ursprünglich für Sprachaufgaben vorgeschlagen wurde, ist ein vielversprechender Mechanismus für die Verarbeitung

von LiDAR-Scans. Allerdings verbietet die quadratische Raumkomplexität in Bezug auf die Anzahl der Punkte seine Anwendung auf große Punktwolken. Um dieses Problem zu lösen, schlagen wir eine speichereffiziente Variante des Aufmerksamkeitsmechanismus vor, die als Globale Hierarchische Aufmerksamkeit (GHA) bezeichnet wird und die reguläre Aufmerksamkeit durch die Berechnung lokaler Aufmerksamkeit auf einer Reihe von Auflösungshierarchien approximiert. GHA bietet zwei wesentliche Vorteile. Erstens hat sie eine lineare Komplexität in Bezug auf die Anzahl der Punkte, so dass sie große Punktwolken verarbeiten kann. Zweitens neigt GHA von Natur aus zu räumlich nahen Punkten und schenkt lokalen Strukturen mehr Aufmerksamkeit, während die globale Konnektivität zwischen allen Punkten erhalten bleibt. In Kombination mit einem Feedforward-Netzwerk kann GHA in viele bestehende Netzwerkarchitekturen eingefügt werden. Unsere Experimente mit mehreren Basisnetzen zeigen, dass das Hinzufügen von GHA die Leistung bei verschiedenen Aufgaben und Datensätzen durchweg verbessert. Bei der Aufgabe der semantischen Segmentierung führt GHA zu einer Steigerung von +1,7% mIoU gegenüber der MinkowskiEngine-Baseline auf ScanNet. Bei der Aufgabe der 3D-Objekterkennung verbessert GHA die CenterPoint-Baseline um +0,5% mAP auf dem nuScenes-Datensatz und die 3DETR-Baseline um +2,1% mAP_{25} und +1,5% mAP_{50} auf ScanNet.

Acknowledgments

I am grateful to many people, without whose support this thesis would not have been possible.

I would like to express my deepest gratitude to my supervisor, Prof. Bastian Leibe, for the opportunity to pursue a PhD in his group and for providing invaluable guidance throughout my studies. His support and insights have been instrumental in shaping this research. Additionally, I extend my heartfelt thanks to Prof. Kai Arras and Prof. Peter Rossmanith for generously agreeing to be the co-examiner of this thesis.

I am profoundly grateful to my parents, whose love for knowledge has been a constant source of inspiration. They have unwaveringly supported my education, even though it meant being separated from their son by more than 7,000 kilometers.

I would also like to extend my thanks to all my colleagues: Alex Hermans, Alexey Nekrasov, Ali Athar, Christian Schmidt, Chrysa Pateromichelaki, Dora Kontogianni, Francis Engelmann, George Lydakis, Idil Esen Zülfikar, István Sárándi, Jens Piekenbrinck, Jonas Schult, Jono Luiten, Markus Knoche, Paul Voigtlaender, Sabari Mahadevan, Stefan Breuers, and Steffi Käs. Their passion for computer vision and technology, their talent, and their willingness to discuss and collaborate have made the lab an inspiring and enjoyable place to work.

A special thanks goes to my long-term collaborator, Alex Hermans, who has been a valuable and trustworthy co-author for all my publications. I am also grateful to István Sárándi for being a command-line wizard and for providing us with amazing IT infrastructure.

Last but not least, I owe my heartfelt thanks to my wife, LJ, whose love, care, and understanding have been my constant support during this final phase of my PhD studies.

Thanks to "the God who has been my shepherd all my life to this day" (Gen 48:15 NIV).

Contents

1 Introduction

1.1 Motivation

In the recent decade, the landscape of robotic applications is undergoing a remarkable transformation, with a surge in the deployment of mobile robots across diverse domains. From assisting with mundane household chores (Hess *et al.*, 2014; Avigal *et al.*, 2022) to navigating complex urban environments (Paez-Granados *et al.*, 2022; Chen *et al.*, 2023a), mobile robots are becoming integral to our daily lives. The International Federation of Robotics (2021) reports double-digit growth worldwide for service robots in 2021, with a turnover reaching 6.7 billion U.S. dollars. The surge in such applications has been fueled by advancements in sensing, computing, and artificial intelligence, collectively contributing to the development of sophisticated autonomous systems. In contrast to industrial robots, which are instrumental in automating production factories, mobile robots are designed to function within environments populated by humans, with the capability to interact and cooperate with them. As the reliance on mobile robots continues to expand, the need for enhancing their perception capabilities becomes increasingly imperative. One such capability is person detection, which plays a crucial role in ensuring safe and efficient human-robot interactions. This thesis delves into the realm of efficient learning-based person detection, specifically leveraging LiDAR sensors, to address this critical aspect of mobile robot functionality.

Person detection on mobile robots is often accomplished using multiple RGB(-D) cameras, in combination with a deep learning based object detector (Jafari *et al.*, 2014; Munaro and Menegatti, 2014; Wang *et al.*, 2023a). Cameras, despite their rich visual information and low cost, do have disadvantages. A camera typically has limited field of view and thus restricts the detection to a narrow frustum. For applications where a panoramic view of the environment is critical (*e.g.* autonomous driving), multiple cameras have to be used, resulting in increased computation requirement and cost. In addition, a (depth-)camera cannot provide accurate range measurement, hence struggles to accurately localize a person in 3D space, especially when the person is at far distance. Last but not least, privacy is of great importance in many

applications (*e.g.* health care), and it may be unacceptable to deploy a mobile robot with cameras in these scenarios.

LiDAR sensors provide an alternative way to address the limitations of cameras. A LiDAR sensor emits a laser ray and detects the reflected signal generated from hitting an obstacle. The distance to the obstacle can be accurately determined by measuring the time of flight between emission and receiving reflection. By mechanically rotating the LiDAR sensor, an accurate scan of the environment can be obtained in the form of a point cloud. (The recently emerged solid state LiDAR no longer needs a rotating component and promises greater mechanical robustness.) A LiDAR sensor typically has a large field of view, often reaching 360°, and can accurately measure range with centimeter accuracy. In addition, the sparse, colorless measurement makes a LiDAR sensor less privacy intrusive, in comparison with a camera. These advantages make the LiDAR sensors a promising choice for person detection. However, efficiently detecting persons with LiDAR sensors is a challenging task. The unique data structure of point cloud prevents a naive application of convolutional neural networks, which is the foundation of most deep-learning techniques on images. The sparsity in the point cloud poses additional challenge, where depending on the sensor resolution, a person at a far distance may not have more than a few points and is hard for a network to detect. Furthermore, the large amount of points contained in one scan demands high computation efficiency for working with real-time applications. Lastly, unlike person detection using images, there are only a few annotated LiDAR dataset specifically focusing on person detection with mobile robots, and annotating new LiDAR data is a laborious and time consuming process.

This thesis proposes person detection techniques for LiDAR sensors which address the aforementioned challenges. We introduce a fast person detector, DR-SPAAM, for detecting persons with 2D LiDAR sensors, which are set up on mobile robots at roughly the height of human ankles (Chapter 4). It uses a spatial attention and auto-regressive module to aggregate measurements from consecutive scans, which alleviates the problem caused by sparse LiDAR points with little computation overhead. To increase the generalization of the trained detector to unknown scenarios or new sensor setups, we propose a method to automatically generate pseudo-training labels (Chapter 5), which can be used to fine-tune the detector or even train it from scratch. The method leverages a camera with known extrinsic calibration, which is often present, or can be easily added, on many mobile robots. In addition, we conduct a large scale experiment analyzing the performance difference between using 2D or 3D LiDAR sensors for detecting persons (Chapter 6). The former is more affordable but scans only a horizontal plane, whereas the latter scans an entire 3D volume at the cost of a higher price tag. Understanding the performance difference of these two sensors provides insight into selecting the right sensors when designing robots. In the end, we propose an efficient variant of the attention mechanism (Vaswani *et al.*, 2017) that can be added to an existing 3D deep-learning network for improved performance (Chapter 7). The newly proposed variant, termed Global Hierarchical Attention, features linear space complexity with respect to the number of points, while maintaining the global receptive field. The reduced complexity is critical for applying attention to large point clouds.

1.2 Contributions

The main contributions of this thesis are:

- We present the DR-SPAAM (Distant Robust SPatial Attention and Auto-regressive Model) for person detection with 2D LiDAR sensors. The novelty of DR-SPAAM lies in its attention-based auto-regressive module to propagate information forward through time, leading to a stronger detection feature with little added compute cost. Evaluated on the DROW dataset, DR-SPAAM has achieved state-of-the-art performance, while having a high framerate of 87.2 FPS on a laptop with a dedicated GPU.
- We present a method to automatically generate training labels for 2D-LiDAR-based person detectors using a calibrated camera with an off-the-shelf image-based detector. We show that the proposed method, together with robust training techniques, is an effective way to train a detector from scratch or fine-tune the detector during deployment.
- We conduct extensive experiments to compare person detection performance using 2D or 3D LiDAR sensors. Our experiments show that both sensors give similar detection performance for visible persons, but it is easier for persons to be fully occluded in 2D LiDAR scans, thus making detection impossible. As a result, the 3D LiDAR sensors give overall higher performance when including the persons invisible to the 2D LiDAR sensor in evaluation.
- We present the Global Hierarchical Attention, a novel attention mechanism that has linear space complexity while maintaining global field of view. The attention module can be added to both point-based and voxel-based networks. We evaluate GHA on a suite of networks for both detection and segmentation tasks across multiple datasets and show improved performance over baseline networks.

1.3 Structure of the Thesis

The remainder of this thesis is structured as follows:

Chapter 2 discusses the preliminaries involved in the thesis. It begins with a description of the LiDAR sensor and different types of data structures used for representing LiDAR scans. Then it formally defines the object detection and segmentation tasks, and explains respective evaluation metrics. Next, it gives an overview of deep learning, which is the technique used for building state-of-the-art detectors. Finally, it discusses particular operations that are designed to work for 3D data, which act as building blocks for the neural networks discussed in this thesis.

Chapter 3 describes existing researches related to person detection with LiDAR sensors. Detecting persons using 2D LiDAR sensors has long-standing history, due to its relevance

with robotic tasks such as navigation and human-machine interaction. We review the existing approaches, ranging from the early methods that rely on detecting motions from the scene (Prassler *et al.*, 1999) to the most recent approaches that use deep learning to obtain state-of-the-art results (Beyer *et al.*, 2018). For 3D LiDAR sensors, person detection has mostly been studied under the categories of object detection with driving scenarios, where "person" or "pedestrian" is one class of objects to be detected. We summarize existing 3D object detection approaches and group them into six categories, based on the type of data representations they operate on. In the end, we list major public datasets used to train person detectors, and compare their statistical properties such as number of annotations and their distributions.

Chapter 4 focuses on person detections using 2D LiDAR sensors, which are commonly available on mobile robots for navigation and mapping. Detecting persons using a 2D LiDAR sensor is particularly challenging due to the low information content of the range measurements. To address this challenge, we propose DR-SPAAM, a person detection network with 2D LiDAR sensors, which alleviates the problem caused by sparse LiDAR measurements by aggregating temporal information in consecutive scans. The temporal aggregation is accomplished via the novel spatial attention and auto-regressive module, which keeps the intermediate features from the backbone network as a template and recurrently updates the template when a new scan becomes available. The updated feature template is in turn used for detecting persons currently in the scene. DR-SPAAM reaches state-of-the-art detection performance, and is particularly well-suited for deployment on mobile robots, having an inference rate as high as 87.2 FPS on a laptop with a dedicated GPU and 22.6 FPS on an NVIDIA Jetson AGX embedded GPU. The content of this chapter is based on the technical contribution presented in Jia *et al.* (2020).

Chapter 5 addresses the shortage of annotated data for training 2D-LiDAR-based person detectors. Significant effort is spent in the past decade on annotating images, resulting in an abundance of labeled image data across a diverse set of scenarios. However, only a few datasets with person annotation exist for 2D LiDAR sensors. This shortage of data can potentially damage the detector's performance when deployed in a new environment or with a new LiDAR model. To address this problem, we propose a method, which uses a calibrated camera and an off-the-shelf image-based detector, *e.g.* Faster R-CNN (Ren *et al.*, 2015), to automatically generate training labels (termed *pseudo-labels*) for the 2D-LiDAR-based detector. Our approach is an effective way to improve person detectors during deployment without any additional labeling effort. Through experiments on the JackRabbot dataset (Martin-Martin *et al.*, 2021) with two detector models, DROW3 (Beyer *et al.*, 2018) and DR-SPAAM (Jia *et al.*, 2020), we confirm that self-supervised detectors, trained or fine-tuned with pseudo-labels, outperform detectors trained only on a different dataset. Combined with robust training techniques, the self-supervised detectors reach a performance close to the ones trained using manual annotations of the target dataset. The content of this chapter is based on the technical contribution presented in Jia *et al.* (2021).

Chapter 6 analyzes the performance gap between 2D and 3D LiDAR sensors for person detection. 2D LiDAR sensors provide accurate range measurement at an affordable price, but their measurement is limited to only one scan plane. 3D LiDAR sensors are more expensive, but have the advantage of scanning with multiple beams, thus providing information for the 3D volume. We conduct a series of experiments on the JackRabbot dataset (Martin-Martin *et al.*, 2021) with state-of-the-art 2D and 3D LiDAR-based person detectors, DR-SPAAM (Jia *et al.*, 2020) and CenterPoint (Yin *et al.*, 2021a) respectively, which act as a proxy to the capabilities of the respective modalities. Our experiments include multiple aspects, ranging from the basic performance and speed comparison, to more detailed analysis on localization accuracy and robustness against distance and scene clutter. The insights from these experiments highlight the strengths and weaknesses of 2D and 3D LiDAR sensors as sources for person detection, and are especially valuable for designing mobile robots that will operate in close proximity to surrounding humans. This chapter is based on the technical contribution presented in Jia *et al.* (2022a).

Chapter 7 introduces a new attention mechanism, called Global Hierarchical Attention (GHA), for 3D point cloud analysis. The permutation-equivariance property of the attention mechanism (Vaswani *et al.*, 2017), along with its effectiveness in channeling long-range information, makes it a particularly promising operation for processing point clouds. However, the memory requirement of the attention operation scales quadratically with respect to the number of points, forbidding its application with modern LiDAR sensors whose point cloud contains several hundreds of thousands of points. The newly proposed GHA approximates the global attention via a series of local attention on multiple hierarchies, and thereby reduces the memory complexity to linear with respect to the number of points, while not suffering the reduced field of view as in other linear attention methods (Wang *et al.*, 2020a; Zhao *et al.*, 2021). Combined with a feedforward network, GHA can be inserted into many existing network architectures. We experiment with multiple baseline networks and show that adding GHA consistently improves performance across different tasks and datasets, *e.g.* a +1.7% mIoU increase to the MinkowskiEngine (Choy *et al.*, 2019a) baseline on ScanNet (Dai *et al.*, 2017) for semantic segmentation, or a +0.5% mAP increase to the CenterPoint (Yin *et al.*, 2021a) on the nuScenes dataset (Caesar *et al.*, 2020) for 3D object detection. This chapter is based on the technical contribution presented in Jia *et al.* (2022b).

Chapter 8 presents the conclusion of the thesis, summarizing its key contributions towards efficient person detection for mobile robots equipped with LiDAR sensors, and pointing out exciting new research perspectives.

Note: *The content of this thesis is based on conference papers mentioned above (Jia* et al.*, 2020, 2021, 2022a,b). Figures and texts are reproduced from these publications with added discussions on future research directions.*

2 Preliminaries

In this chapter, we present preliminary knowledge related to the LiDAR-based person detection methods developed in this thesis. Section 2.1 provides an explanation of the working mechanism of a LiDAR sensor and the 3D point cloud data structure used to represent LiDAR measurements. Section 2.2 formally defines the main tasks of this thesis, namely person detection and segmentation from LiDAR point clouds. Section 2.3 briefly reviews key developments made in the past half century on using deep neural networks and supervised learning to solve complex computer vision tasks, which serve as foundations to modern detection methods. Section 2.4 identifies challenges of using deep learning on LiDAR data and explains common neural networks suitable for working with point clouds.

2.1 LiDAR Sensors and Point Cloud Representation

LiDAR, an acronym for "light detection and ranging", is a technique for measuring the distance from the sensor to an obstacle. While the intricate electronic design and physics theory is beyond the scope of this thesis, at a high level, a LiDAR sensor works by emitting a laser ray toward a surface and detects the reflection. The distance to the surface can be determined by measuring the time of flight between sending the ray and registering a reflection. A single laser ray can only measure the distance to one point. In order to scan a surface, multiple points must be measured. This is achieved by mechanically rotating the laser emitter or using phased array antennas to steer the laser beam. Depending on the number of rotating angles, two types of LiDAR sensors exist: 2D and 3D.

A 2D LiDAR sensor has one angle of rotation, called the *azimuth* (or *horizontal*) angle, and scans only one plane. A 3D LiDAR sensor has an additional *elevation* (or *vertical*) angle, which allows it to provide scanning measurements for a 3D volume. Figure 2.1 shows the scans obtained from 2D and 3D LiDAR sensors. Angular resolution and field of view (FoV) are important performance specifications of LiDAR sensors. Most LiDAR sensors today provide 360° coverage for the azimuth angle while the elevation angle can have a FoV up to

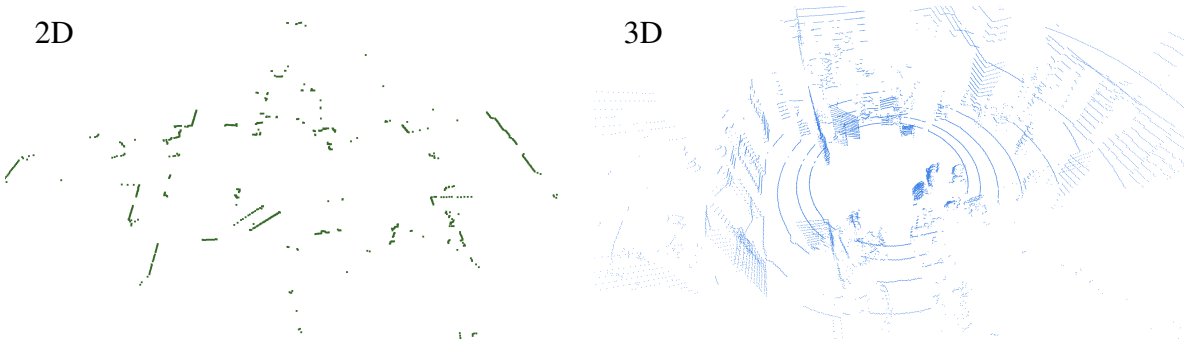


Figure 2.1: Scans from 2D and 3D LiDAR sensors. The 2D LiDAR sensor scans one plane by rotating the laser ray with an azimuth angle. The 3D LiDAR sensor has an additional elevation angle, and scans multiple planes.

40° for high-end models. 2D LiDAR sensors have been popular on mobile robots for purposes such as navigation, mapping, and safety, due to their affordable price. 3D LiDAR sensors were once limited to high-end applications like autonomous driving, but they are now becoming popular on mobile robots thanks to reduced prices driven by improved hardware technology. Other important performance specifications of LiDAR sensors include measurement range, acquisition rate, unit weight, and power consumption.

The measurements captured by LiDAR sensors are commonly represented as point clouds, voxels, or range images, which are shown in Figure 2.2. A point cloud is a set of points $\{X_i\}$ where each point $X_i = [x, y, z]^T$ corresponds to a measured location in a given coordinate. It is important to note that unlike an image, a point cloud does not have an order. This means that a point cloud with three points can be expressed in multiple ways, such as $\{X_1, X_2, X_3\}$ or $\{X_2, X_1, X_3\}$, among other permutations. Ideally, the learning mechanism (*e.g.* a neural network) operating on point cloud should be invariant or equivariant with respect to permutation, which will be explained further in Section 2.4.

Voxel representation is obtained by discretizing a point cloud into 3D bins. This provides a volumetric version of the point cloud that is similar to an image. The neighborhood structure of the voxel grid allows for efficient operations such as discrete convolution, but this comes at the cost of reduced resolution compared to the original point cloud. Memory requirement is additionally a limiting factor when choosing the bin size, as the number of voxels grows cubically with the increased resolution. A sparse representation is often used instead of a dense one because most voxels in a LiDAR scan are empty.

Another way to represent LiDAR measurements is using range images. A range image is similar to an RGB image, but instead of representing color at each "pixel", it represents the measured distance at the corresponding azimuth and elevation angle. The 3D and 2D LiDAR measurements are represented with 2D and 1D range images respectively.

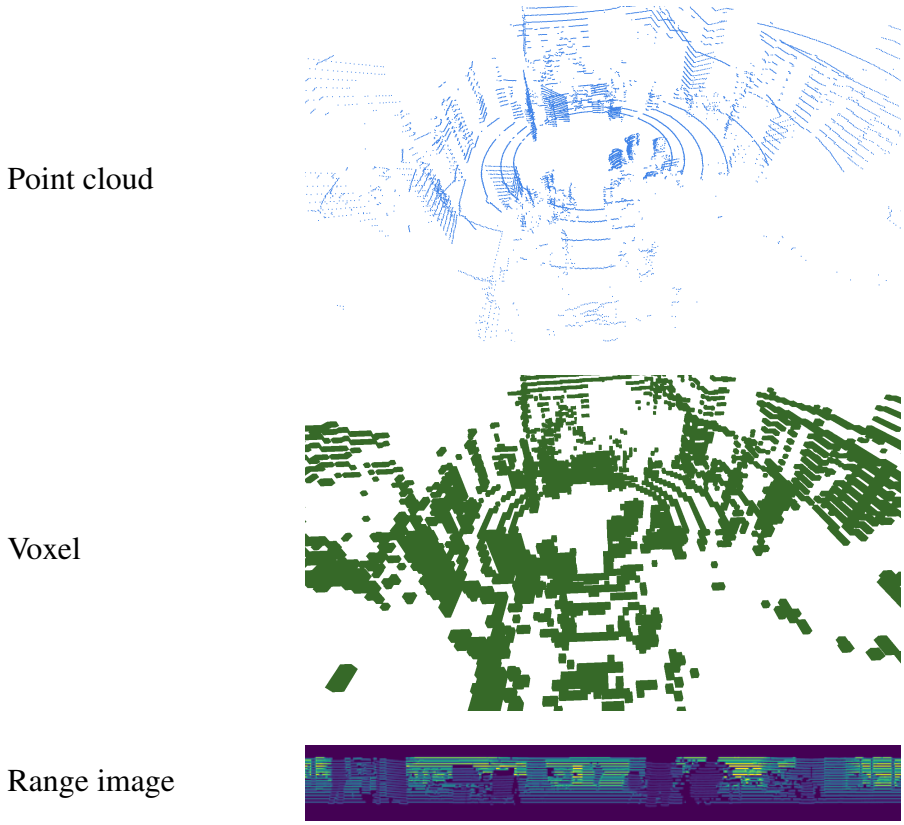


Figure 2.2: LiDAR scans are commonly represented as point clouds, voxels, or range images.

2.2 Object Detection and Segmentation

This thesis focuses on the problem of detection and segmentation with LiDAR point clouds. This section presents a definition of these two tasks and explains their evaluation procedure.

3D object detection is an important task in computer vision, and often an integral part of more advanced algorithms such as instance segmentation. The two-fold goal of object detection is to find what objects are in the scene (*classification*) and where are those objects (*localization*). The objects come from pre-defined categories. For example, the popular KITTI dataset (Geiger *et al.*, 2012) annotates bounding boxes of three categories of objects: cars, pedestrians, and cyclists, and the goal of a detector is to correctly predict the location of the bounding boxes and their class labels. Mathematically speaking, the goal of a detector is to predict a set of boxes $\{\hat{B}_i\}$ from an input image or point cloud. The parameterization of a bounding box depends on the task and dataset conventions. For 3D object detection, a bounding box is usually parameterized with 8 numbers: 3 for box centroid location, 3 for box dimensions, 1 for rotation along vertical axis, and 1 for class labels. An example is shown in Figure 2.3.

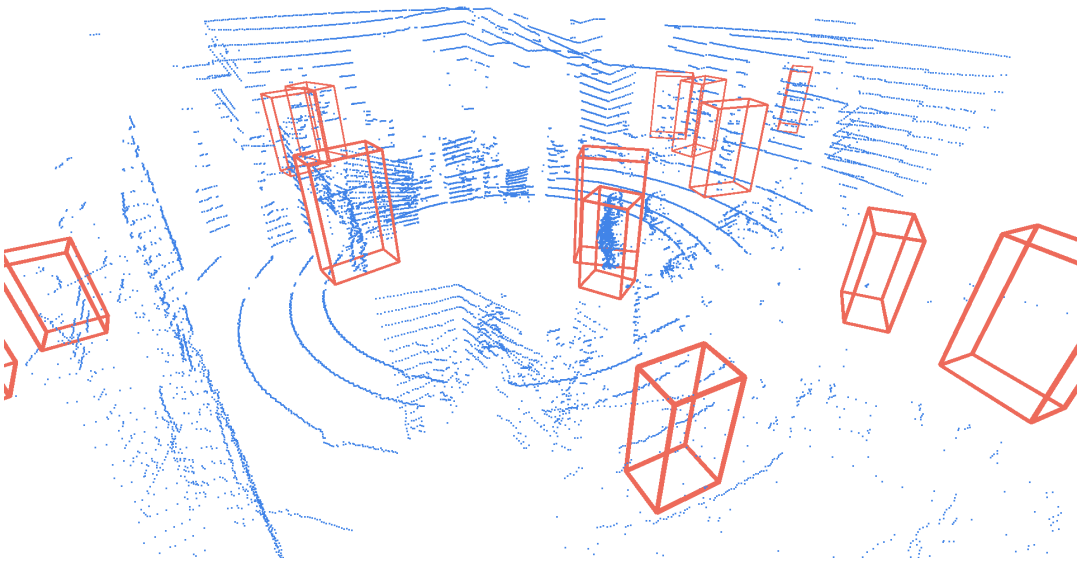


Figure 2.3: A LiDAR point cloud with 3D bounding boxes marking persons in the scene. The object detection task involves finding all objects, parameterized as bounding boxes, belonging to a given object class.

Average precision (AP) is the common metric used to evaluate the quality of detections. To compute AP, the predicted boxes $\{\hat{B}_i\}$ are first matched against ground truth boxes $\{B_j\}$. Two boxes are considered a positive match if their intersection over union (IoU) is greater than a threshold, and each prediction can be matched to at most one ground truth, and vice versa. While optimal solution exists for such a bipartite matching problem, in practice, most evaluation procedure simply implements a greedy matching strategy, where predictions with high confidence take priority in matching with the closest ground truth boxes. Given the bipartite matching between prediction and ground truth, three classification statistics can be collected: True Positive (TP)—number of matched predictions; False Positive (FP)—number of unmatched predictions; and False Negative (FN)—number of unmatched ground truth. These three statistics are used to compute the precision and recall metrics

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.1}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}\,, \tag{2.2}$$

each ranging in $[0, 1]$. Intuitively, the precision metrics measures the quality of positive predictions, whereas the recall metrics measures the coverage of predictions over ground truth.

The confidence scores associated with each predicted boxes are real numbers in $[0, 1]$, and a thresholding needs to be applied before computing precision and recall. By varying the confidence threshold, either more or fewer predictions will be produced, resulting in a higher

recall or precision. By continuously varying the confidence value, a precision-recall curve can be produced. The average precision (AP), the mean of precision over all recall values, is the area under this precision-recall curve. AP is the common metric used to compare detector. In practice, rather than computing the continuous integration, the area under precision-recall curve is approximated by averaging precision at a number of fixed recall thresholds. For example, the popular COCO and KITTI dataset evaluates precision at 41 evenly distributed recall levels (Lin *et al.*, 2014; Geiger *et al.*, 2013). Usually AP is computed with different IoU threshold to match predictions and ground truth, and the localization performance of a detector can be gauged by examining AP evaluated at different IoU thresholds. The popular KITTI dataset computes AP with three IoU thresholds: 30%, 50%, and 70%. In detections where there are multiple classes, the APs are computed separately for each class. It is common to report the mean average precision (mAP) by computing the arithmetic mean of all AP values. The 2D-LiDAR-based person detection tasks covered in this thesis only annotates a point of a person location, rather than a bounding box. In this case, the distance is used for matching predictions with ground truth.

Segmentation is another important computer vision task, where the goal is to segment objects or areas in an image or a point cloud. Different type of segmentation tasks exist: *Semantic segmentation* annotates each point with a semantic class (*e.g.* wall, curtains, *etc.*) and the task of the network is to predict the class label for each point. *Instance segmentation* is similar to object detection, but rather than producing bounding boxes, the network is tasked with predicting a segmentation mask for each object instance. Compared to semantic segmentation, instance segmentation requires not only properly predicting the class of each point, but also separating each instance (by assigning object ID) in case where there are multiple object instances belonging to the same class. An example of instance segmentation is shown in Figure 2.4. *Panoptic segmentation* refers to the combination of semantic segmentation and instance segmentation, where the former is required for amorphous *stuff* like grass, walls, and the later is required for countable *things* where the instances can be clearly defined.

Intersection over union (IoU) between predicted and ground truth mask is the typical evaluation metrics for segmentation tasks, and is computed as

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} , \tag{2.3}$$

where the True Positive (TP), False Positive (FP) and False Negative (FN) counts the number of points whose predictions agree or disagree with the ground truth masks. In the setting where there are multiple classes, mean IoU (mIoU) is often used by computing the mean of all IoU values across all classes.

2.3 Supervised Learning and Neural Networks

Most modern detection and segmentation approaches, including those presented in this thesis, are accomplished via deep neural networks. While it is impossible to cover the numerous

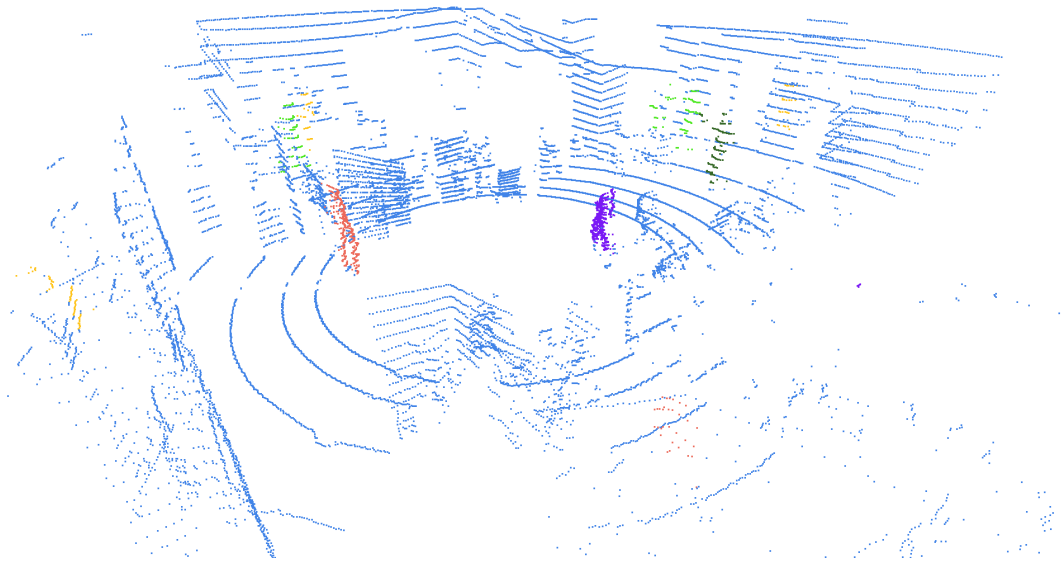


Figure 2.4: A LiDAR point cloud with persons in the scene shown as colored instance masks. The segmentation tasks involves segmenting the point cloud based on semantic classes or object instances.

topics related to DNN in a single thesis, this section aims to provide a concise discussion of several topics which are critical in order to understand the rest of this thesis.

2.3.1 Supervised Learning

The topics covered in this thesis is concerned with supervised learning. Given a data pair $(x, y) \in (\mathcal{X} \times \mathcal{Y})$ where $x \in \mathcal{X}$ is the input (*e.g.* point cloud) and $y \in \mathcal{Y}$ the label (*e.g.* segmentation mask), the goal of supervised learning is to obtain a model $\mathcal{F}_\theta$ parameterized by θ such that

$$\hat{y} = \mathcal{F}_\theta(x) \tag{2.4}$$

best predicts the label y for the input x. This is typically set up as an optimization problem, where the goal is to find the optimal θ^* that would minimize the *true risk*, expressed as a loss function L, over the entire data distribution

$$\theta^* = \arg\min_\theta \int L(y, \mathcal{F}_\theta(x)) dP(x, y) \ . \tag{2.5}$$

Explicitly computing the integral is clearly intractable. Instead, it is common to sample a set of *training* data, and find the optimal θ^* that minimizes the training loss

$$\theta^* = \arg\min_\theta \sum_{(x,y) \in \text{train}} L(y, \mathcal{F}_\theta(x)) \ . \tag{2.6}$$

The choice of loss function depends on the task itself. For example, the cross-entropy loss $L_{\text{CE}} = -\sum y \log(\hat{y})$ is often used for classification, and the $L1$ loss $L_1 = |y - \hat{y}|$ for regression. After the optimization is done, a set of *testing* data, which is withheld during the optimization, is used to estimate the true risk and to evaluate the performance of the model. Note that the final training loss obtained after the optimization is under usual condition not a good estimate of the true risk due to model overfitting on the training data.

The discussion up to this point has purposefully taken a general stance, where neither the model $\mathcal{M}$ nor the optimization procedure is specified. Indeed, a plethora of models can be used for supervised learning, ranging from Support Vector Machines (Cortes and Vapnik, 1995) to Decision Trees (Breiman, 2001), each with its own different optimization procedure and learning property. The rest of this thesis focuses on a specific type of model known as the neural networks. For a full discussion on supervised learning, the readers are advised to consult a machine learning textbook (Hastie *et al.*, 2009).

2.3.2 Multi-Layer Perceptron (MLP) and Convolutional Neural Network (CNN)

Artificial neural networks are layered networks composed of interconnected neurons, and are originally inspired by the biological neurons in the human brain. This section reviews two particularly influential network architectures, the Multi-Layer Perceptron (MLP) and Convolutional Neural Network (CNN). These two architectures are developed over the course of the past half-century, and a modern complete network, including those presented in this thesis, often combines them in an interwoven fashion. We omit the discussion on other important type of neural networks such as Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) or Graph Neural Network (Bronstein *et al.*, 2016), since they are not directly related to the topics covered in this thesis.

For the ease of discussion, this section assumes images, rather than point clouds, are the input to the neural network. Most of the networks discussed in this section are originally proposed to operate on images. These well-studied image-based approaches often serve as inspirations when developing methods that work with point clouds. However, working with 3D data does have its unique challenges, and are elaborated in Section 2.4.

Perceptron, also known as a linear layer or a fully-connected layer, is a simple neural network with one linear layer for binary classification (Mcculloch and Pitts, 1943; Rosenblatt, 1958). It can be expressed as

$$f(x) = \begin{cases} 1 & w^T x + b > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{2.7}$$

where $x \in \mathbb{R}^N$ is the input (*e.g.* an image flattened as a vector), $w \in \mathbb{R}^N, b \in \mathbb{R}$ are the weights and biases of the network, and the output $f(x)$ is the binary class prediction (*e.g.* if the image is of a cat). It is shown however that this simple single layer perceptron is incapable

of classifying patterns that are not linearly separable, an example of which would be the XOR function (Minsky and Papert, 1969).

Multi-layer Perceptron (MLP), often loosely referred to as a feedforward network, is a composition of multiple perceptron layers

$$\begin{aligned} h_1 &= w_0^T x \ + b_0, \ z_1 = \phi(h_1) \\ h_2 &= w_1^T z_1 + b_1, \ z_2 = \phi(h_2) \\ &\dots \\ y &= w_n^T z_n + b_n \end{aligned} \tag{2.8}$$

where the intermediate output of one layer, after passing through a non-linear activation function ϕ, is passed on as the input to the layer after. The intermediate layers are referred as hidden layers, and they are parametrized by weights $w_i \in \mathbb{R}^{W_i \times W_j}$ and bias $b_i \in \mathbb{R}^{W_j}$. Both tanh and the logistic function are popular activation functions in the past, whereas the Rectified Linear Units (Agarap, 2018) or similar variants are more popular among more recent networks. The output y can be used for binary classification if combined with a step function or for regression. With the internal representation learned from the hidden layers, a Multi-layer Perceptron can solve classification problems where the pattern is not linearly separable (Rumelhart *et al.*, 1986). Furthermore, a theorem known as the Universal Approximation proves that a Multi-layer Perceptron with one or more hidden layers can approximate any continuous function to any arbitrary degrees of accuracy (Cybenko, 1989).

Convolutional Neural Network (CNN) is perhaps the single most used network architecture in modern-day image processing (Fukushima, 1980; LeCun *et al.*, 1989). Contrary to the MLP which uses linear operation, a CNN uses the *convolution operation* to connect the neurons between two layers

$$h_{i+1} = \text{conv}_{w_i, b_i}(z_i), \ z_{i+1} = \phi(h_{i+1}) \ , \tag{2.9}$$

where the kernel weights w_i and biases b_i are the learnable parameters, and ϕ is the activation function. Note that in case of an image, the feature map z_i will be a $(W_i \times H_i \times C_i)$ tensor, where W, H corresponds to the spatial dimension of an image and C is the number of feature channels. The size of the convolution kernel and the number of output channels are hyperparameters chosen beforehand. *Pooling* is used to reduce the spatial dimensions of the network. This is accomplished by moving a strided window along the feature map, similar to a convolution, but only keeps the maximum or the average value of features in the window. There is no learned parameters in a pooling layer, and the type of pooling operation and the size of the pooling window are hyperparameters.

A convolutional neural network has two critical advantages compared to the Multi-layer Perceptron. First, in a convolution operation, the learnable kernel weights and biases are shared for neurons across the spatial feature map, resulting in a much lower parameter count compared to a densely connected linear layer. Second, a convolutional neural network satisfies many symmetries involved in common image processing tasks (Bronstein *et al.*, 2016). Here,

symmetry refers to the invariance and equivariance of an operation under certain transformation (Klein, 1893). The feature map of a CNN are equivariant with respect to the input image under translation, *i.e.* the feature map shifts accordingly when the input image is shifted. If this feature map is pooled into one vector and used for classification, the prediction will remain invariant. A MLP on the other hand, shares no symmetry under translation, due to the weights being assigned to fixed locations.

Modern CNN can have hundreds of stacked convolution layers. A problem that plagued such deep networks is the vanishing gradient, where the learnable parameters in early network layers receive little to no update during training. To solve this problem, He *et al.* (2016) propose to add a residual connection between two layers for propagating the training gradient. Such a residual connection (or skip connection) is now a common practice when designing deep neural networks.

Pooling operation used in a CNN increases compactness of the feature map. However, in dense prediction tasks like segmentation, the network needs to make prediction at full resolution. A common practice is to upsample the compact feature map, either through non-learned techniques like interpolation, or learned ones like transposed convolution (Noh *et al.*, 2015; Dumoulin and Visin, 2016). The resulting network usually has two branches, one encoder branch that gradually downsamples the feature map, and one decoder branch that upsamples, with skip connections in-between two branches at matching resolutions. This type of network, known as U-Net, is often used in dense prediction tasks (Badrinarayanan *et al.*, 2015; Ronneberger *et al.*, 2015).

2.3.3 Training Neural Networks

This section gives an overview of several important techniques involved in training a neural network for supervised learning. Training a network refers to adjusts its learnable parameters in order to obtain a model that best fit the data. As discussed in Section 2.3.1, this training procedure is framed as an optimization problem.

Gradient-based Learning While early researches used various approaches to adjust the network parameters (Rosenblatt, 1958; Fukushima, 1980), gradient descent is the most common approach that is used today when training a deep neural networks. To minimize the loss $L(\theta)$ in Equation 2.6 on the training set, gradient descent iteratively updates the network parameter θ via

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta L(\theta^{(t)}) \tag{2.10}$$

with the superindex t refers to the number of iteration. Here η is the *learning rate* and ∇_θ is the gradient operator with respect to the network parameters. The initial parameter, $\theta^{(0)}$, is usually randomly sampled from a uniform or normal distribution (Glorot and Bengio, 2010; He *et al.*, 2015). The learning rate η can be a fixed constant or be adjusted according to a schedule through the course of the training (Loshchilov and Hutter, 2017; Smith and Topin, 2019). Setting a proper learning rate is critical. A learning rate that is set too high may cause the training to diverge, whereas setting the learning rate too low leads to prolonged training.

The training loss $L(\theta)$, as shown in Equation 2.6, takes form as a summation of losses on all individual samples. In practice, the training set is usually too large for such a computation to be carried out at each gradient descent step. Instead of computing the true training loss over the entire dataset, the loss on a few randomly drawn samples, or a *batch*, are used to approximate the true loss and its gradient. This technique is known as the *stochastic gradient descent*, and the size of the batch is a training hyperparameter specified before-hand. A larger batch size provides better approximation to the true gradient at the cost of increased computation.

To improve the stability of the optimization, Rumelhart *et al.* (1986) proposed to add a *momentum* term to Equation 2.10 by mixing in previous gradient update

$$\begin{aligned} \Delta\theta^{(t)} &= \alpha\Delta\theta^{(t-1)} - \eta\nabla_\theta L(\theta^{(t)}) \\ \theta^{(t+1)} &= \theta^{(t)} + \Delta\theta^{(t)} \ , \end{aligned} \tag{2.11}$$

with α being a hyperparameter controlling the strength of momentum. This additional term reduces the randomness in gradient direction caused by stochastic gradient descent.

The magnitude of elements in the gradient vector in Equation 2.10 may differ significantly, leading to unbalanced update on the parameters. In order to address this problem, the *Adam* optimizer seeks to adaptively adjusts the learning rate for each element independently (Kingma and Ba, 2015). It keeps a running average of the gradient and its second moment

$$\begin{aligned} m^{(t+1)} &= \beta_1 m^{(t)} + (1-\beta_1)\nabla_\theta L(\theta^{(t)}) \\ v^{(t+1)} &= \beta_2 v^{(t)} + (1-\beta_2)(\nabla_\theta L(\theta^{(t)}))^{\circ 2} \end{aligned} \tag{2.12}$$

and updates the parameters via

$$\begin{aligned} \hat{m} &= \frac{m^{(t+1)}}{1-\beta_1^t} \\ \hat{v} &= \frac{v^{(t+1)}}{1-\beta_2^t} \\ \theta^{(t+1)} &= \theta^{(t)} - \eta\frac{\hat{m}}{\sqrt{\hat{v}}+\epsilon} \ . \end{aligned} \tag{2.13}$$

β_1, β_2 are the forgetting factors set beforehand (*e.g.* 0.9 and 0.999), ϵ is a constant for numerical stability, and $(\cdot)^{\circ 2}$ refers to the Hadamard square operation. The *AdamW* optimizer (Loshchilov and Hutter, 2019) combines Adam with a weight decay regularization

$$\theta^{(t+1)} = \theta^{(t)} - \eta\frac{\hat{m}}{\sqrt{\hat{v}}+\epsilon} + w\,\theta^{(t)} \ , \tag{2.14}$$

and the hyperparameter w controls strength of weight decay.

An efficient technique known as *backpropagation* can be used to compute the gradient of a neural network in Equation 2.10 (Werbos, 1981; Rumelhart *et al.*, 1986). An explanation on this standard technique can be found in most deep learning textbooks (Goodfellow *et al.*,

Algorithm 1 Gradient-based learning for a neural network

Input: Network $\mathcal{F}_\theta$ with randomly initialized parameters θ, loss function $L(y, \hat{y})$
  repeat
    Sample a training batch $\{(x_i, y_i)\}$
    Make prediction $\hat{y}_i = \mathcal{F}_\theta(x_i)$, compute loss $L = \sum_i L(y_i, \hat{y}_i)$ ▷ Forward pass
    Compute gradient $\nabla_\theta L$ via backpropagation ▷ Backward pass
    Update parameters θ using Equation 2.10 or alternatives
  until Finish training

2016) or lecture notes, and is thus omitted from here. With the gradient being available, the overall procedure of training a network is summarized in Algorithm 1.

Normalization It has been shown that normalizing the input and activation of each layer is an effective technique in helping large networks converge. Among the most well-known normalization techniques is the batch normalization (Ioffe and Szegedy, 2015; Santurkar *et al.*, 2018), which can be expressed as

$$x_{\text{norm}} = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta \ , \tag{2.15}$$

where $\mathrm{E}[\cdot], \mathrm{Var}[\cdot]$ are the mean and variance, γ, β are learnable parameter vectors, and ϵ is a small constant for numerical stability. Here the normalization is carried out over all examples in a batch for each feature channels independently. This batch normalization layer is often added before or after the activation function, effectively normalizing the input to the next layer. During training, a running average of the batch mean and variance are kept. These statistics are used during inference on individual examples.

One drawback of batch normalization is its reliance on batch. In order to reliably estimate batch statistics, sufficiently large batch size is needed, which can be expensive for certain vision tasks. Using running average statistics during inference, where no batch is available, introduces discrepancy between training and inference. Recent development on normalization techniques has aimed to remove the dependency on batch. Layer normalization (Ba *et al.*, 2016) normalizes each individual examples independently for all channels, whereas instance normalization (Ulyanov *et al.*, 2016) normalizes each channel separately. Group normalization (Wu and He, 2018) splits features into groups and normalizes each group independently. These newer normalization techniques all operate on each examples independently without relying on batch statistics, and operate in a same way during training and inference. These techniques have been proposed for their specific use cases, and the community has not yet reached a consensus on what is the overall best technique.

Dropout In order to achieve good generalization performance, it is imperative to regularize a machine learning model. One effective technique for regularization of a neural network is through the use of Dropout, as described by Hinton *et al.* (2012). This technique involves randomly setting some elements of a tensor to zero during training, based on a Bernoulli

distribution with a chosen probability of p, while re-scaling the remaining elements. The dropout layer is usually added after activation and serves as an efficient method for overall network regularization.

2.4 Deep Learning on 3D Point Cloud

The design of deep neural networks for point cloud processing has in recent years garnered considerable research attention, due to the increased availability of 3D data collected using range sensors or through image-based reconstruction. This section reviews two deep learning operations which have been specifically designed to work with 3D data, namely the PointNet (Qi *et al.*, 2017b,a) and the submanifold sparse convolution (Graham *et al.*, 2018). In addition, this section discusses the transformer architecture (Vaswani *et al.*, 2017), which is originally proposed for language tasks and is recently adapted to work with 3D data. These operations are closely related to the person detection approaches presented in this thesis, and are used as the basic blocks in a complete neural network, much like the convolution and feedforward layer in a network that processes images.

2.4.1 PointNet and PointNet++ (Qi *et al.*, 2017b,a)

Point cloud is perhaps the most common representation of 3D data, especially those obtained by LiDAR sensors. Mathematically speaking, a point cloud is a set of vectors, where each element in the set represents the x, y, z coordinate of point in 3D space. Even though the point cloud is often implemented as an $N \times 3$ array, there is no order within a point cloud, meaning that every permutation of the points (the rows) is a representation of the same point cloud. This poses unique challenges for designing neural networks, since the ideal predictions need to be invariant or equivariant with respect to permutation of the input (Bronstein *et al.*, 2016). A network that classifies the point cloud should predict the same logits for all possible permutation of the input point cloud (*i.e.* the prediction is invariant to permutation), whereas a segmentation network should predict a per-point segmentation mask that permutes consistently with the input (*i.e.* the prediction is equivariant to permutation). It is clear that naively applying convolution or MLP on the point cloud array, treating it like an image, does not provide the desired symmetry with respect to permutation, and poor results are reported even if a canonical ordering is imposed by first sorting the point cloud.

A pioneering architecture for 3D point cloud processing is the PointNet (Qi *et al.*, 2017b), which directly operates on a point cloud via permutation-invariant operations (Zaheer *et al.*, 2017). The building block of PointNet is the set abstraction (SA) operation, which can be expressed as

$$\mathrm{SA}(\,\{x_i\}\,) = \gamma(\,G(\,\{\phi(x_i)\}\,)\,)\,, \tag{2.16}$$

where $\{x_i\}$ is the input point cloud, γ, ϕ are MLP networks, and G is a permutation-invariant operation like average- or max-pooling. It is proven that this set abstraction operation is able

to approximate any continues functions $f : 2^{\mathbb{R}^d} \mapsto \mathbb{R}^{d'}$ that maps a set of vectors to a single vector.

Given an input point cloud, PointNet (Qi *et al.*, 2017b) obtains a global scene feature by performing set abstraction on all points, and then concatenates the global feature with the input features for each individual points. In this way, global scene context is mixed into each individual points. This operation is repeated a few times. In the end, features from all points can be pooled together and a linear layer can be used for classification. For segmentation, a shared linear layer can be used to project each individual features into confidence predictions.

The PointNet++ (Qi *et al.*, 2017a) pairs the set abstraction with a hierarchical architecture. At each hierarchy level, a subset of seed points are sampled via farthest point sampling. For each seed point, set abstraction is performed on points with distance less than a predefined radius to extract local neighbor features. This subset of points, whose feature includes neighborhood structure from the set abstraction, is then passed to next hierarchy level. The hierarchies of PointNet++ allows the network to extract features at different scales, similar to the different receptive field in a convolutional neural network. After several hierarchy levels, features from all remaining points can be pooled into a scene feature for classification. For detection or segmentation tasks, PointNet++ implements an upsampling decoder to recover the full-resolution point cloud, with skip connections between encoder and decoder at each hierarchy levels. A shared final linear layer can then be used for predicting bounding box proposal or class confidence for each point.

The key characteristic of PointNet and PointNet++ is its ability to directly operate on a point cloud. The set abstraction operation and its variants are commonly used as building blocks in many point-based approaches, including the Point-GHA discussed in Chapter 7. Besides the set abstraction, there are other permutation-invariant operations, like the continuous convolution (Wang *et al.*, 2018a; Thomas *et al.*, 2019) or local attention (Zhao *et al.*, 2021), that can directly operate on points. We refer the interested readers to the respective publications for details.

2.4.2 Submanifold Sparse Convolution (Graham *et al.*, 2018)

Voxelizing the point cloud is an alternative to working directly with points, and has the benefit of being able to leverage efficient convolution operation. The challenge, however, is the memory consumption, since the number of voxels grow cubicly with respect to the scene size or resolution. To reduce memory consumption, sparse voxel representation is often used, which only stores features for non-empty voxels, along with their coordinates. This sparse representation is particularly suited for LiDAR data which normally contain abundance of empty spaces. However, naively applying convolution on sparse representation would still lead to high memory consumption, due to the dilation effect of convolution operation. With a regular convolution, any non-empty positions within the convolution kernel will lead to a non-empty output, gradually reducing the sparsity of the feature volume. To maintain the sparsity throughout the network, Graham *et al.* (2018) proposes a modified convolution, whereas the

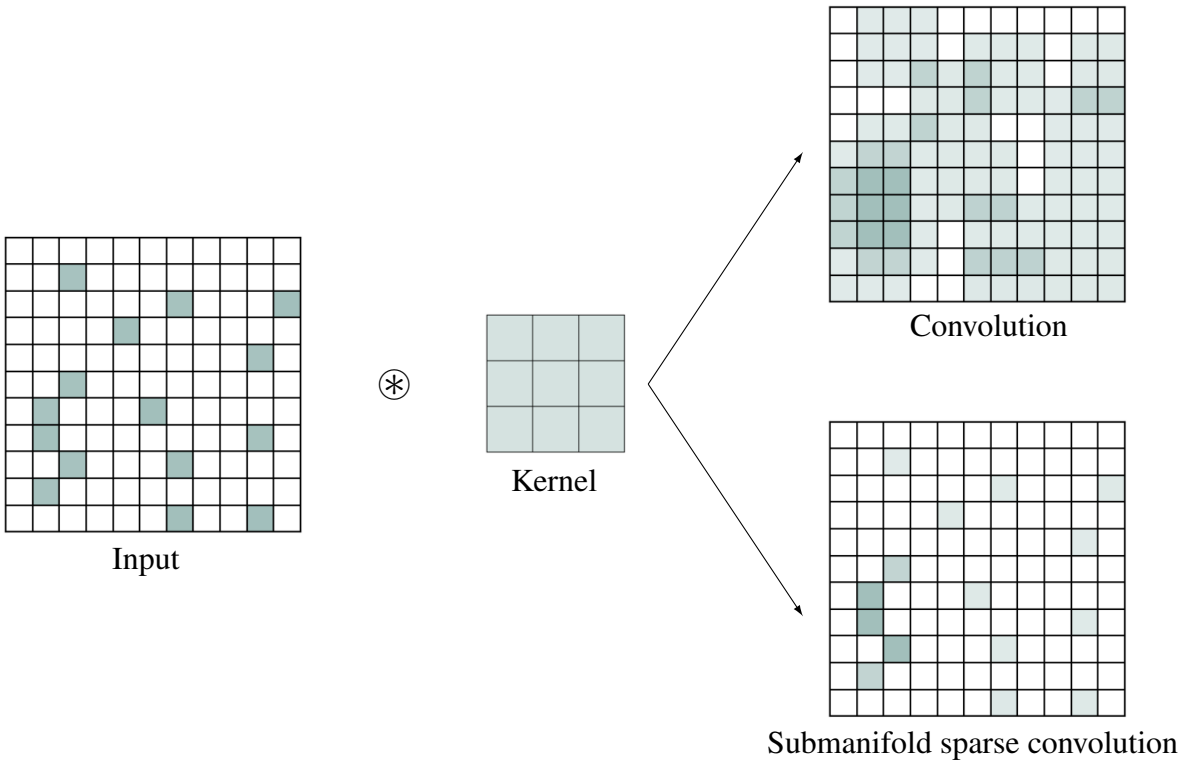


Figure 2.5: Effect of convolution *vs.* submanifold sparse convolution (Graham *et al.*, 2018) on sparse input voxels, shown in 2D for clarity. The convolution operation has an dilation effect and and increases the number of non-empty voxels. Submanifold sparse convolution restricts the output activation to only non-empty input voxels, thus maintaining sparsity.

output is limited to only the non-empty voxels, as shown in Figure 2.5. This operation, termed submanifold sparse convolution, has become the common operation in voxel-based network.

There are several implementations of the submanifold sparse convolution, each originally published under different context. MinkowskiEngine (Choy *et al.*, 2019a) is a particularly popular implementation, and contains various U-Net architecture that can be used as the backbone network for processing a voxelized point cloud. The encoder part is a ResNet-like network (He *et al.*, 2016) implemented with submanifold sparse convolution. The decoder uses transposed sparse convolution to upsample the feature map, and has skip connections from the encoder at each resolution. MinkowskiEngine is originally showcased with segmentation tasks for indoor ScanNet (Dai *et al.*, 2017) and S3DIS (Armeni *et al.*, 2016). TorchSparse (Tang *et al.*, 2022) is a similar implementation that uses GPU-based voxel hashing and various optimizations, and reports speed advantage over MinkowskiEngine. SECOND (Yan *et al.*, 2018) is another implementation of the submanifold sparse convolution, and emphasizes object detection in outdoor LiDAR dataset.

2.4.3 Attention and Transformers (Vaswani *et al.*, 2017)

The transformer architecture is originally proposed for language tasks (Vaswani *et al.*, 2017) and in recent years has gained significant popularity in computer vision (Dosovitskiy *et al.*, 2021a; Liu *et al.*, 2021b). The core of the transformer architecture is the attention mechanism, which takes as input a set of feature vectors (*e.g.* language tokens or image patches), refereed to as *tokens*. First, the tokens are independently projected into three sets of feature vectors called *queries*, *keys*, and *values*, and these are then used to compute a scaled dot-product attention

$$Z = \text{softmax}(\frac{QK^T}{\sqrt{d}})V \ . \tag{2.17}$$

Here Q, K, $V \in \mathbb{R}^{N\times d}$ are the queries, keys, and values, N is the number of tokens, d is the feature dimensions, and $Z \in \mathbb{R}^{N\times d}$ is the attention output. In essence, the attention mechanism computes cosine similarities between all pairs of queries and keys, and performs weighted average on the values using as weights the normalized similarities. Equation 2.17 can equivalently be expressed in the matrix form

$$\begin{aligned} A &= \exp[\frac{QK^T}{\sqrt{d}}] \\ D &= \text{diag}(A \cdot \mathbf{1}_N) \\ Z &= D^{-1}AV \end{aligned} \tag{2.18}$$

where $\mathbf{1}_N = [1, 1, \ldots, 1]^T$ represents a vector of 1 with N dimension and $\exp[\cdot]$ denotes the element-wise exponential. The matrix product $D^{-1}A$ is often referred to as the normalized attention matrix. A wide-spread practice is to use multi-head attention, where the embedding vectors are chunked to multiple groups (*i.e.* multi-head) along the channel dimension and attentions are separately computed for each group.

The attention mechanism is used to build a full transformer architecture, which typically features an encoder-decoder architecture. The encoder computes the *self-attention* among the set of input tokens via a series of attention blocks, exchanging information within the set. Each attention block includes a multi-head attention operation, followed by a feedforward and normalization layer. The decoder takes the encoder feature tokens and a set of *query embeddings*, and computes in an alternating fashion *self-attention* within query embeddings and *cross-attention* between the feature tokens and query embeddings. These query embeddings, in the most common settings, are simply a set of learnable vectors with random initialization. Through the decoder, the query embeddings are transformed into the final output (*e.g.* classification label, detection boxes, or text predictions).

The attention mechanism, as shown in Equation 2.18, is equivariant with respect to permutation, making it an especially promising candidate for processing 3D point cloud. In addition, the attention mechanism can channel long range information, having in effect a global receptive field. In the next chapter (Chapter 3) we provide a summary on many recent developments that seek to adapt transformer architecture for 3D vision tasks. Many of these developments

focus on reducing the quadratic memory consumption of the attention operation, in order to cope with the large amount of points. A particular adaption, the Global Hierarchical Attention (Jia *et al.*, 2022b), is the central theme discussed in Chapter 7.

3 Related Work

This chapter reviews existing research on detecting persons with LiDAR sensors, and is structured into three sections. Section 3.1 reviews existing research on detecting persons from a 2D range scan, which has a long history due to its importance related to robotic applications like navigation and human-machine interaction. Section 3.2 focuses on object detection with 3D LiDAR point cloud, of which person or pedestrian is often one class of objects to be detected. Section 3.3 provides a summary of the public datasets for LiDAR-based detection, which are used to train and test various detectors mentioned in this thesis.

3.1 Person Detection with 2D LiDAR sensors

Detecting persons using a 2D range scanner has long been a relevant task in the robotics community, due to its relevance to a wide range of robotic applications in human-populated environments. In this section, we present an overview of the important developments in 2D-LiDAR-based person detection. From a historical perspective, these developments can be divided into three stages. Methods in the first stage focus on finding motions in the scene by comparing consecutive frames. Clusters of points that move in a consistent fashion are considered to be coming from some moving objects, presumably persons. In the second stage, learning-based classifiers are introduced, to determine if points or groups of points belong to a person or background. These classifiers use manually engineered features, and temporal filtering based on some motion model is indispensable to obtain reliable performance. In the third stage, deep learning becomes the primary method for detecting persons in the 2D LiDAR scan. These deep-learning-based detectors give high-quality detection, even using only a single frame with no motion filtering.

3.1.1 First Period: Tracking Moving Points

The task of finding persons using a range scanner arises from the effort of building autonomous systems that can navigate in human-populated environments. Rather than focusing on a sin-

gle frame, early attempts leverage scan sequences from the LiDAR sensor with the goal of tracking moving objects. An early work by Prassler *et al.* (1999) converts LiDAR scans into occupancy grids in bird's-eye-view, and compares two consecutive occupancy grids aligned using the onboard odometry. A moving object is said to exist in a grid if the grid is occupied in the current frame but is free in the previous one. This moving object is then assigned to one of the maintained trajectories based on nearest neighbour search (Li and Bar-Shalom, 1996). Similar temporal comparison is utilized by Lindstrom and Eklundh (2001), but rather than using an occupancy grid, Lindstrom and Eklundh (2001) propose to find and track scan points that violate the free space, which is estimated as the union of scan polygons using the past few frames. The scan polygons are obtained by connecting neighboring scan points with lines. Kluge *et al.* (2001) propose to segment the range scan by checking if the range difference between two consecutive points (sometimes referred to as the jumping distance) exceeds a given threshold. Points within each segment are further split into disjoint convex hulls (Preparata and Shamos, 1985), each of which is considered to be one object. Object tracking across scans is accomplished via a bipartite matching (Ahuja *et al.*, 1993) on the segmented convex hulls in the consecutive frames.

Filtering techniques are often used in early approaches for temporal data association. For example, Fod *et al.* (2002) present an approach that tracks people using multiple static laser scanners. The approach uses background modeling (Toyama *et al.*, 1999) to isolate and group foreground points, and tracks the grouped points via a Kalman filter across consecutive frames. Montemerlo *et al.* (2002) propose a particle filtering model that simultaneously estimates the sensor ego-motion and tracks people in pre-mapped environments. The distance between a LiDAR point and the hypothesised person location is used to assign probabilities to the hypothesis. Schulz *et al.* (2001, 2003) combine particle filter with Joint Probabilistic Data Association Filters (JPDAF) (Cox, 1993). The former is used to track the states of the objects, while the latter is used to assign the measurements to each individual object. The local minima in the range scan are assigned with high probability as they are likely caused by foreground objects such as persons. Taylor and Kleeman (2004) combines multi-hypothesis tracking (MHT) (Reid, 1979) with a dynamic model on the movement of a person which takes into account the physical assumptions about the legs. Mucientes and Burgard (2006) propose to track clusters of objects and solve data association within each tracked cluster, in order to improve tracking of objects moving in groups. Instead of treating a person as one object to be tracked, Arras *et al.* (2008) propose to track individual legs with MHT, and group legs into persons with heuristics taking into account of self-occlusion.

Detecting stationary persons or differentiating persons from other moving objects is often a challenge for approaches that rely solely on motion clues. A parallel line of research seeks to improve person detection by building a geometric description of the leg patterns. Xavier *et al.* (2005) propose to group points in a scan by distance to neighboring points, and detect legs of persons by looking for groups with circular shapes and right dimensions. Instead of circles, MacLachlan and Mertz (2006) propose to fit rectangles to the scan segment, and track the corners of fitted rectangles. Topp and Christensen (2005) combine earlier works relying on the convex hull for detecting persons (Kluge *et al.*, 2001) and JPDAF for tracking (Schulz

et al., 2003), and propose heuristics to accept or reject a convex pattern as a person, taking into account the consistency with the tracking hypothesis. Similar person tracking strategy is used by Gockley *et al.* (2007) and Hemachandra *et al.* (2011) to build a human-following robot with a 2D LiDAR sensor. Navarro-Serment *et al.* (2008) propose a set of heuristics to classify if a tracked segment corresponds to a person. The heuristics take into account the size and velocity of the segment, its variation, as well as the distance it traveled. Cui *et al.* (2007) accumulates temporal scan sequences by means of occupancy grid in bird's-eye view, and uses the local maxima in the grid as the location of a person. Another way to improve detection reliability is to fuse measurements from additional sensor modalities, typically cameras. Kleinehagenbrock *et al.* (2002) propose to fuse symbolic information extracted with a camera and LiDAR, such as face and legs of persons, using perceptual anchoring (Coradeschi and Saffiatti, 2001). Scheutz *et al.* (2004) propose a person tracking method that combines camera-based face detection and LiDAR-based leg detection. The latter is accomplished by searching for groups of points that are sufficiently far from the background, and have the right size and separation that matches the two legs of a person. A similar strategy is used by Cui *et al.* (2005), in combination with a Bayesian estimation formulation, to track people with stationary cameras and LiDAR sensors. Schulz (2006) tracks joint person exemplars in LiDAR and camera measurements via a Rao-Blackwellized particle filter and proposes a method to learn the examplars from the training data via unsupervised EM clustering.

3.1.2 Second Period: Supervised Learning with Hand-Crafted Features

The early developments in detecting persons rely on tracking motions in scan sequences, and the focus has been on designing tracking modules, leaving detections to simple heuristics like finding convex hulls or changes in occupancy. However, the large amount of false detections produced by these simple heuristics can overwhelm the data association, producing poor performance. Based on this observation, subsequent developments have focused on improving the quality of detection. In particular, methods using supervised learning begin to emerge, motivated by their success in vision tasks.

Arras *et al.* (2007) propose the first system that uses supervised learning techniques to detect persons in 2D LiDAR scans. The system segments a LiDAR scan using the jumping distance between adjacent points, and computes for each segment a total of 14 features. These hand-crafted features cover geometric properties of the segment, such as its radius, curvature, and the number of contained points. Based on these features, an AdaBoost classifier (Viola and Jones, 2002) is trained to identify segments that correspond to legs of people. A small amount of data is collected using a stationary laser scanner and manually labeled for training and testing the classifier. Two different environments, office and corridor, are covered by the collected data, and good generalization ability is reported.

Pantofaru (2010) implement a ROS package that pairs the learned leg classifier from Arras *et al.* (2007) with an algorithm that groups the legs into persons based on distance and tracks

the resulting persons, but the details of the system are not documented in a publication. Lu and Smart (2013) discuss efficient methods for robots to navigate around humans, and use the methods from Pantofaru (2010) to provide person locations for the navigation planner.

Leigh *et al.* (2015) similarly segment the LiDAR scan into clusters using the jumping distance between adjacent points. The threshold is tuned such that the two legs of a human can result in two distinct clusters, but each human rarely generates more than two clusters. A Random Forest (Breiman, 2001) is used to classify clusters into human or non-human, using features extended from Arras *et al.* (2007). A self-collected dataset, composed of 1700 positives and 4500 negatives, is used to train the classifier. To collect the positive training examples, the scanner is set in an open area with significant human traffic, and every cluster resulting from the jumping distance segmentation is considered to be human. For the negative examples, the scanner is moved around in an area void of humans, and all clusters are considered non-human. Then a Kalman filter with global nearest neighbor association is used to track all clusters, and the human tracks are used in a closed-loop controller for human following.

3.1.3 Third Period: Detection with Deep Learning

The early works with supervised learning (Arras *et al.*, 2007; Leigh *et al.*, 2015) focus on improving feature selection and alleviating the burden of data association, but the classifier with hand-crafted features introduced on top of the jumping distance segmentation is not sufficient to produce reliable detections that can be used without motion filtering. Instead of using hand-crafted features, later work resorts to deep learning to detect persons directly from the range scan. These detectors are able to produce reliable detections even from a single range scan, and do not require motion-based filtering to limit false predictions.

The DROW detector proposed by Beyer *et al.* (2016) is the first deep-learning-based detector to work with 2D range data. It is first developed for detecting wheelchairs, and a later extension includes person detection (Beyer *et al.*, 2018). The detector pipeline consists of three stages. First, the raw scans are preprocessed into small per-point windows, which are referred to as *cutouts*, in order to normalize the appearance across different distances. These cutouts are then separately classified by a 1D CNN network as either object or background, and a possible object center is regressed for every cutout. Finally, all regressed object centers, referred to as *votes*, are collected and aggregated into a final set of detections.

The input to the detector is a 2D LiDAR scan composed of N points, expressed in polar coordinates as $\{(r_n, a_n)\}_{n=0}^{N-1}$. (The measurements are made at a fixed angular grid, and r_n stands for the range measured at the n-th angle.) Similar to the jumping distance segmentation, the cutout preprocessing aims at isolating, for each point, its neighboring points that fall within a fixed-size window in Euclidean space. To accomplish this, Beyer *et al.* (2016) propose to first compute an angular opening Δa_n for each point using

$$\Delta a_n = \arctan \frac{0.5 \cdot \Delta W}{r_n}, \tag{3.1}$$

with ΔW being a hyperparameter specifying the width of the window we would like to isolate in Euclidean space. Then the cutout C_n is generated by taking neighboring points that fall within the angular window

$$C_n = \{(r_m, a_m) | m = 0, \ldots, N-1, |a_m - a_n| \leq \Delta a_n\}. \tag{3.2}$$

These points are then re-sampled to a fixed number of M points, centered by subtracting the range r_n of the central point, clipped to within $\pm\Delta r$ to eliminate background and foreground points (Δr is a hyperparameter), and normalized to $[-1, 1]$.

The cutout preprocessing alleviates problems caused by unequal sampling densities at different distances (LiDAR points are sparser at far ranges), and also allows the DROW detector to work with LiDAR sensors of different angular resolutions without requiring re-training. Furthermore, the clipping operation removes the background information, allowing the network to focus on the neighboring points that are in the same distance region.

The normalized cutouts obtained from the preprocessing step are passed through a 1D CNN. The network outputs a confidence score for classification, as well as the 2D bird's-eye view location of the person, expressed as an offset from the LiDAR point. In the postprocessing step, predictions coming from all points are accumulated into a voting grid. A non-maximum suppression is applied to the voting grid to obtain the final set of detections.

The DROW dataset collected by Beyer *et al.* (2018) is publicly available and is used extensively in experiments in this thesis. Specifically, the DROW dataset is used to develop and benchmark the DR-SPAAM detector (Jia *et al.*, 2020), which extends the DROW detector with a spatial attention and auto-regressive module to incorporate temporal information from sequential scans. The DR-SPAAM detector is the main topic of Chapter 4.

3.2 Object Detection with 3D LiDAR Sensors

Person detection with 3D LiDAR sensors has mostly been investigated in the context of autonomous driving, where "person" or "pedestrian" is a class of objects to be detected. In this section, we provide an overview of 3D object detection networks for LiDAR point clouds. These networks are designed in a class-agnostic fashion, and can be repurposed for person detection on mobile robots. However, in order to achieve good performance, it is important to select network hyperparameters and training data that suit the scenario at testing or deployment, as highlighted in Chapter 6. A large body of research exists on 3D object detection, the scope of which cannot be captured in this section alone. We recommend interested readers consult recent surveys by Qian *et al.* (2022) and Mao *et al.* (2022).

The methods discussed in this section are grouped into six categories, based on the point cloud representation used and the type of network. *Projection-based methods* use one or more projected images such as bird's-eye view or range view, and process the views with a 2D image-based network. *Voxel-based methods* voxelize the point cloud and encode 3D features with a sparse 3D network, which is then converted to a dense 2D feature map and processed with a 2D network. Discretization is involved in both projection and voxelization, whereas

point-based methods directly operate on raw point clouds, *i.e.* an ordered set of vectors. *Hybrid methods* combine two or more representations from points, voxels, or views. The remaining two categories are dedicated to two recently emerged trends: *fully sparse voxel networks*, where the sparse features are directly converted to bounding box predictions without the intermediate 2D dense feature map, and *transformer-inspired architectures* (Vaswani *et al.*, 2017). This taxonomy is summarized in Table 3.1.

3.2.1 Projection-Based Methods

A series of early works project 3D point clouds into range or bird's-eye view, and use 2D convolutional neural networks to process the obtained point cloud images. Multiple views are often processed by independent network branches, and the features from all branches are then combined to make predictions. For example, MV3D (Chen *et al.*, 2017) uses three independent networks to extract features from bird's-eye view, front view, and the RGB image. 3D proposals are generated using features from the BEV branch, and these proposals, after being projected to the respective view planes, are used to pool and fuse features from all branches, which in turn generates refined predictions. AVOD (Ku *et al.*, 2017) similarly relies on two network branches, one for the RGB image and the other for the BEV point cloud, and uses 3D anchors to combine features from the two branches to generate predictions. A concurrent work by Liang *et al.* (2018) uses a continuous fusion scheme that merges multi-scale features from the RGB image branch into the BEV feature branch for final predictions.

A single view, either BEV or range view, can be sufficient for detecting objects. For example, PIXOR (Yang *et al.*, 2018b) uses a simple CNN to directly predict BEV bounding boxes from the top-down view of point clouds. HDNet (Yang *et al.*, 2018a) seeks to augment the input BEV image with a semantic map, exploiting the connection between the objects and where they appear (cars for most of the time drive on the road). BirdNet (Beltrán *et al.*, 2018) predicts BEV bounding boxes with a network using the top-down view of the point cloud as the input, and uses ground plane estimation to recover the height of the 2D boxes, assuming the bottom of the boxes are stationed on the ground. Instead of using the BEV image, LaserNet (Meyer *et al.*, 2019) uses 2D convolution on the range view, and proposes a novel probabilistic detection head to predict bounding boxes and the associated uncertainty. LaserFlow (Meyer *et al.*, 2020) extends LaserNet to motion forecasting by processing consecutive frames. RCD (Bewley *et al.*, 2020) proposes to use a dilated convolution conditioned on range in place of the regular 2D convolution, in order to cope with the scale change at different distances. RangeDet (Fan *et al.*, 2021b) adds a convolution layer with a PointNet kernel before the regular 2D backbone, and uses a target assignment strategy based on the range of the points. PPC (Chai *et al.*, 2021) experiments with multiple kernel choices, including 2D convolution, PointNet (Qi *et al.*, 2017b), self-attention (Vaswani *et al.*, 2017), and EdgeConv (Wang *et al.*, 2019c). The range view representation has also been used in LiDAR semantic segmentation (Milioto *et al.*, 2019).

By converting point clouds into images, projection-based approaches can leverage well-crafted networks that are originally proposed for image tasks. With an efficient 2D feature

Table 3.1: Existing works in 3D object detection with LiDAR sensors.

Method	Venue	Category
MV3D (Chen *et al.*, 2017)	CVPR	Projection-based (Section 3.2.1)
AVOD (Ku *et al.*, 2017)	IROS	
HDNet (Yang *et al.*, 2018a)	CoRL	
PIXOR (Yang *et al.*, 2018b)	CVPR	
Cont Fuse (Liang *et al.*, 2018)	ECCV	
BirdNet (Beltrán *et al.*, 2018)	ITSC	
LaserNet (Meyer *et al.*, 2019)	CVPR	
RCD (Bewley *et al.*, 2020)	CoRL	
LaserFlow (Meyer *et al.*, 2020)	RA-L	
PPC (Chai *et al.*, 2021)	CVPR	
RangeDet (Fan *et al.*, 2021b)	ICCV	
VoxNet (Maturana and Scherer, 2015)	IROS	Voxel-based with dense feature map (Section 3.2.2)
Vote3D (Wang and Posner, 2015)	RSS	
VoxelNet (Zhou and Tuzel, 2017)	CVPR	
SECOND (Yan *et al.*, 2018)	Sensors	
CBGS (Zhu *et al.*, 2019a)	arXiv	
MVF (Zhou *et al.*, 2019b)	CoRL	
PointPillars Lang *et al.* (2019)	CVPR	
CIA-SSD (Zheng *et al.*, 2021)	AAAI	
TANet (Liu *et al.*, 2020b)	AAAI	
Voxel R-CNN (Deng *et al.*, 2020)	AAAI	
HVNet (Ye *et al.*, 2020)	CVPR	
SA-SSD (He *et al.*, 2020a)	CVPR	
AFDet (Ge *et al.*, 2020)	CVPRW	
HotSpotNet (Chen *et al.*, 2020c)	ECCV	
Pillar-OD (Wang *et al.*, 2020b)	ECCV	
SSN (Zhu *et al.*, 2020)	ECCV	
CVCNET (Chen *et al.*, 2020b)	NeurIPS	
Voxel-FPN (Kuang *et al.*, 2020)	Sensors	
Part-A^2Net (Shi *et al.*, 2020a)	T-PAMI	
CenterPoint (Yin *et al.*, 2021a)	CVPR	
CyliNet (Rapoport-Lavie and Raviv, 2021)	ICCVW	
CenterNet3D (Wang *et al.*, 2021b)	T-ITS	
AFDetV2 (Hu *et al.*, 2022)	AAAI	
Focals Conv (Chen *et al.*, 2022c)	CVPR	
INT (Xu *et al.*, 2022)	ECCV	
MPPNet (Chen *et al.*, 2022b)	ECCV	

Continued on next page

Table 3.1: Existing works in 3D object detection with LiDAR sensors. (Cont.)

Method	Venue	Category
PillarNet (Shi *et al.*, 2022a)	ECCV	
SPS-Conv (Liu *et al.*, 2022a)	NeurIPS	
LargeKernel3D (Chen *et al.*, 2023b)	CVPR	
MMF (Liang *et al.*, 2019)	CVPR	Voxel-based with dense feature map on LiDAR-camera fusion (Section 3.2.2)
PointPainting (Vora *et al.*, 2019)	CVPR	
MVX-Net (Sindagi *et al.*, 2019)	ICRA	
3D-CVF (Yoo *et al.*, 2020)	ECCV	
PointAugmenting (Wang *et al.*, 2021a)	CVPR	
MVP (Yin *et al.*, 2021b)	NeurIPS	
BEVFusion (Liu *et al.*, 2023a)	ICRA	
IPOD (Yang *et al.*, 2018c)	arXiv	Point-based (Section 3.2.3)
Frustum-PointNet (Qi *et al.*, 2018)	CVPR	
PointFusion (Xu *et al.*, 2018a)	CVPR	
PointRCNN (Shi *et al.*, 2019)	CVPR	
3DSSD (Yang *et al.*, 2020)	CVPR	
EPNet (Huang *et al.*, 2020)	ECCV	
IA-SSD (Zhang *et al.*, 2022)	CVPR	
VoteNet (Qi *et al.*, 2019)	ICCV	Point-based on indoor RGB-D scan (Section 3.2.3)
HGNet (Chen *et al.*, 2020a)	CVPR	
ImVoteNet (Qi *et al.*, 2020)	CVPR	
MLCVNet (Qian *et al.*, 2020)	CVPR	
3D-MPA (Engelmann *et al.*, 2020)	CVPR	
H3DNet (Zhang *et al.*, 2020b)	ECCV	
BRNet (Cheng *et al.*, 2021)	CVPR	
VENet (Xie *et al.*, 2021)	ICCV	
Fast Point R-CNN (Chen *et al.*, 2019)	ICCV	Hybrid (Section 3.2.4)
STD (Yang *et al.*, 2019)	ICCV	
RangeRCNN (Liang *et al.*, 2020b)	arXiv	
PV-RCNN (Shi *et al.*, 2020b)	CVPR	
HVPR (Noh *et al.*, 2021)	CVPR	
LIDAR-RCNN (Li *et al.*, 2021b)	CVPR	
RangeIoUDet (Liang *et al.*, 2021)	CVPR	
RSN (Sun *et al.*, 2021)	CVPR	
Graph R-CNN (Yang *et al.*, 2022a)	ECCV	
PV-RCNN++ (Shi *et al.*, 2022b)	IJCV	

Continued on next page

Table 3.1: Existing works in 3D object detection with LiDAR sensors. (Cont.)

Method	Venue	Category
GSDN (Gwak *et al.*, 2020)	ECCV	Voxel-based with fully sparse network (Section 3.2.5)
Person-MinkUNet (Jia and Leibe, 2021)	CVPRW	
FCAF3D (Rukhovich *et al.*, 2022)	ECCV	
FSD (Fan *et al.*, 2022b)	NeurIPS	
FSDv2 (Fan *et al.*, 2023a)	arXiv	
VoxelNeXt (Chen *et al.*, 2023c)	CVPR	
TR3D (Rukhovich *et al.*, 2023)	ICIP	
FSD++ (Fan *et al.*, 2023b)	T-PAMI	
VoTr (Mao *et al.*, 2021b)	ICCV	Transformer-inspired (Section 3.2.6)
CT3D (Sheng *et al.*, 2021)	ICCV	
M3DETR (Guan *et al.*, 2021)	WACV	
VISTA (Deng *et al.*, 2022)	CVPR	
VoxSeT (He *et al.*, 2022a)	CVPR	
GHA (Jia *et al.*, 2022b)	GCPR	
3DETR (Misra *et al.*, 2021)	ICCV	Transformer-inspired with DETR-like decoder (Carion *et al.*, 2020) (Section 3.2.6)
Object DGCNN (Wang and Solomon, 2021)	NeurIPS	
CenterFormer (Zhou *et al.*, 2022)	ECCV	
ConQueR (Zhu *et al.*, 2023)	CVPR	
PVT-SSD (Yang *et al.*, 2023b)	CVPR	
SST (Fan *et al.*, 2022a)	CVPR	Transformer-inspired with shifted window attention (Liu *et al.*, 2021b) (Section 3.2.6)
SWFormer (Sun *et al.*, 2022)	ECCV	
DSVT (Wang *et al.*, 2023b)	CVPR	
FlatFormer (Liu *et al.*, 2023b)	CVPR	
GD-MAE (Yang *et al.*, 2023a)	CVPR	
3D-MAN (Yang *et al.*, 2021)	CVPR	Transformer-inspired on multi-sensor fusion (Section 3.2.6)
DeepFusion (Li *et al.*, 2022a)	CVPR	
TransFusion (Bai *et al.*, 2022)	CVPR	
UVTR (Li *et al.*, 2022b)	NeurIPS	

extractor, these detectors can often achieve a high frame rate, and are favorable for applications that require real-time detection. However, the information loss caused by projection and discretization on a 2D view may lead to sub-optimal performance.

3.2.2 Voxel Networks with Dense Feature Map

Projecting 3D point clouds into a 2D view risks the loss of volumetric information. An alternative discretization approach is to voxelize the point cloud, and the resulting 3D feature volume can be processed using 3D convolution. Among the first works that explore this idea are Maturana and Scherer (2015) and Wang and Posner (2015). Using voxelization and 3D convolution, Maturana and Scherer (2015) develop a network that classifies a given 3D window of interest into specific object categories. Wang and Posner (2015) develop a 3D object detector by sliding a window classifier over the voxelized feature volume.

More recent approaches to detecting 3D objects attempt to extract a feature volume over the entire scene using a 3D CNN encoder and convert the feature volume into a 2D feature map by means of strided convolution or pooling along the height dimension. This 2D feature map then serves as the input to a subsequent 2D detector for end-to-end detection. An early example is VoxelNet (Zhou and Tuzel, 2017). It uses a few 3D convolution layers to process the voxel feature volume. Stride is applied to the vertical dimension, gradually flattening the feature volume into a feature map in bird's-eye view (BEV). In the end, an anchor-based region proposal network (RPN) is applied to the 2D feature map to predict bounding boxes. The RPN is similar to that for 2D object detection on images (Ren *et al.*, 2015), which creates multiple anchor boxes with pre-defined sizes for each pixel and tasks the network with classifying whether an anchor overlaps with any ground truth bounding boxes and with regressing a refined box proposal for positive anchors. VoxelNet additionally features a learnable operation for voxelizing a point cloud. This operation, termed voxel feature encoding (VFE), is adopted by many later approaches. The VFE layer partitions the point cloud into voxels, and applies a set abstraction operation from PointNet (Qi *et al.*, 2017b) on points within the same voxel. The learnable VFE stands in contrast with prior works (Wang and Posner, 2015) that use hand-crafted heuristics to voxelize a point cloud.

The dense voxel representation and 3D convolution in VoxelNet are memory intensive and slow to compute. To remedy this problem, Yan *et al.* (2018) propose the SECOND detector, which uses sparse convolution (Graham and van der Maaten, 2017) to process the feature volume after the VFE layer. Similar to VoxelNet, the 3D encoder of SECOND gradually downsamples the vertical dimension (via strided sparse convolution), and outputs, in the end, a feature map in BEV. Finally, this sparse feature map is converted to a dense tensor and fed into an RPN that is similar to the Single Shot MultiBox Detector (Liu *et al.*, 2016) for bounding box generation.

The SECOND detector is developed with the KITTI dataset (Geiger *et al.*, 2013), which is a driving dataset that has only a small number of pedestrians and cyclists. To increase the number of training examples for these rare classes, Yan *et al.* (2018) propose a cut-paste augmentation, where ground truth boxes from the whole training set are randomly selected and

pasted into the LiDAR scans. This augmentation greatly improves the network performance for rare classes, and has become a common practice for training detectors on driving datasets. Newer large-scale datasets like nuScenes (Caesar *et al.*, 2020) contain fine-grained classes that have similar appearances (*e.g.* truck and construction vehicle), and classes with few training examples. Zhu *et al.* (2019a) propose to group similar-looking classes into a single detection head, and try to balance the number of training samples between detection heads. This class-balanced grouping and sampling (CBGS) significantly improves the performance and is often used together with SECOND or subsequently developed detectors.

Both VoxelNet and SECOND require carefully picking the height of the voxel grid in order to obtain a flattened feature map at the end of the 3D encoder. This can be cumbersome for deploying the detector on different platforms. To remedy this problem, Lang *et al.* (2019) propose the PointPillars detector, which partitions the point cloud only using a BEV grid, eliminating the need to manually tune the voxel height.[1] A set abstraction operation is used to group points within the same pillar. This feature encoding scheme, in contrast to VFE, directly outputs a 2D feature map, which is then treated as an image and processed by a 2D CNN for object detection. Pillar-OD (Wang *et al.*, 2020b) is similar to PointPillars, but instead of using raw point clouds, it uses pre-extracted point-wise features when pillarizing the point cloud. The point-wise features are extracted by a 2D CNN on the BEV and range view projections of the point cloud. PillarNet (Shi *et al.*, 2022a) proposes to first process the feature pillars with 2D sparse convolution, similar to SECOND, and then convert the sparse pillars to a dense feature map.

CenterPoint (Yin *et al.*, 2021a) augments SECOND or PointPillars with a novel keypoint-based detection head, motivated by similar works from the image domain (Law and Deng, 2020; Duan *et al.*, 2019; Zhou *et al.*, 2019a). Contrary to the anchor-based approaches, the detection head in CenterPoint outputs a heatmap representing the proximity to object centers, and regresses corresponding bounding box parameters. A similar anchor-free detection head is seen in several other concurrent works, including HotSpotNet (Chen *et al.*, 2020c) and CenterNet3D (Wang *et al.*, 2021b). An optional second-stage refinement can be employed on the original bounding box predictions. At the time of its publication, CenterPoint achieved top results on the nuScenes (Caesar *et al.*, 2020) and Waymo (Sun *et al.*, 2020) datasets for 3D object detection from LiDAR data. In Chapter 6 of this thesis, we re-train CenterPoint for person detection, and use it as the baseline for benchmarking against different approaches.

Various subsequently developed networks take CenterPoint as the base module. AFDet (Ge *et al.*, 2020) augments CenterPoint with various optimization and training tricks, and won the 2020 Waymo Open Dataset Challenge. A follow-up work, AFDetV2 (Hu *et al.*, 2022), proposes to weight the confidence target for each proposal by its IoU with the ground truth box, in order to encourage better localized boxes with higher scores. It additionally uses self-calibrated convolution (Liu *et al.*, 2020a) in the backbone network that processes dense BEV features, and adds auxiliary supervision during training on the corner locations of the BEV box (Wang *et al.*, 2021b). SPS-Conv (Liu *et al.*, 2022a) takes a CenterPoint detec-

[1] A BEV grid can be considered as a 3D voxel grid with infinite height, hence "pillars".

tor and augments its 3D sparse convolution with customized spatial pruning. The pruning is done using the magnitude of the voxel features (Zagoruyko and Komodakis, 2017; Yang *et al.*, 2022b). Features with large magnitude are considered important and passed into the convolution, whereas features with small magnitude are directly passed to the next layer via skip connections. LargeKernel3D (Chen *et al.*, 2023b) modifies the 3D backbone network of CenterPoint with large kernels of various sizes ranging from $7 \times 7 \times 7$ to $17 \times 17 \times 17$, motivated by the success of using large convolution kernels in image tasks (Liu *et al.*, 2022b; Ding *et al.*, 2022). To reduce memory consumption, LargeKernel3D proposes to reduce the number of trainable parameters by sharing weights over several pre-defined kernel locations, and uses learnable positional embeddings to encode fine-grained positional information. INT (Xu *et al.*, 2022) and MPPNet (Chen *et al.*, 2022b) extend CenterPoint to work with temporal data. The former works by storing previous scans in a memory bank, and the latter introduces a refinement stage that combines features from box proposals in multiple frames.

Other works using voxelized point clouds include MVF (Zhou *et al.*, 2019b), HVNet (Ye *et al.*, 2020), Voxel-FPN (Kuang *et al.*, 2020), TANet (Liu *et al.*, 2020b), SSN (Zhu *et al.*, 2020), SA-SSD (He *et al.*, 2020a), CIA-SSD (Zheng *et al.*, 2021), CyliNet (Rapoport-Lavie and Raviv, 2021), CVCNET (Chen *et al.*, 2020b), Focals Conv (Chen *et al.*, 2022c), Part-A^2Net (Shi *et al.*, 2020a), and Voxel R-CNN (Deng *et al.*, 2020). MVF (Zhou *et al.*, 2019b) seeks to improve the voxel feature encoding (VFE) layer used in VoxelNet for extracting voxel features from point clouds. It uses a dynamic voxelization scheme which eliminates the need to subsample or repeat random points and voxels in the previous implementation. In addition, it proposes to voxelize the point cloud in both BEV and frontal views, and combine the features from the two views. HVNet (Ye *et al.*, 2020) and Voxel-FPN (Kuang *et al.*, 2020) use VFE on multi-resolution voxels. The former concatenates features from different resolutions, whereas the latter passes the features into the 3D encoder at appropriate downsampling levels. TANet (Liu *et al.*, 2020b) replaces the PointNet-based VFE with a triple attention mechanism to extract voxel features from points inside the voxel. SSN (Zhu *et al.*, 2020) introduces an auxiliary loss function based on the shape of objects, motivated by the strong correlation between object shape and class. SA-SSD (He *et al.*, 2020a) adds an auxiliary network to the 3D encoder of the SECOND detector (Yan *et al.*, 2018). The auxiliary network, which is only present during training and detached during inference, performs dense prediction tasks like segmentation, encouraging the voxel-based encoder to maintain fine-grained structures. CIA-SSD (Zheng *et al.*, 2021) takes the 3D encoder from SECOND and processes the output dense BEV features with a customized spatial-semantic feature aggregation module. The output of the aggregation module is fed to the detection head. CyliNet (Rapoport-Lavie and Raviv, 2021) proposes to voxelize the point cloud in cylindrical coordinates, instead of the commonly used Cartesian coordinates. CVCNET (Chen *et al.*, 2020b) voxelizes the point cloud in a coordinate system resembling spherical coordinates, except that only x and y are used in the Euclidean sum to compute the range coordinate (similar to cylindrical coordinates). Focals Conv (Chen *et al.*, 2022c) introduces a focal sparse convolution operation, which augments the submanifold sparse convolution with learned dilation. Part-A^2Net (Shi *et al.*, 2020a) is the voxel version of its predecessor PointRCNN (Shi *et al.*, 2019). It follows a two-stage de-

tection strategy. In the first stage, bounding box proposals are generated using a 3D U-Net on voxelized point clouds. Besides the box proposals, the network outputs, for each point, a foreground/background segmentation score and an offset prediction to its corresponding bounding box center, normalized by the bounding box size. In the second stage, the segmentation mask and offset prediction are used by a customized RoI-aware pooling to aggregate features within proposal boxes and produce refined detections. Voxel R-CNN (Deng *et al.*, 2020) augments the SECOND detector with a second-stage refinement. The output from the SECOND detector is considered as the proposal boxes. The sparse voxel features (at the end of the 3D encoder) within each proposal box are pooled via a customized Voxel RoI pooling, and the pooled features are used for generating refined detection boxes.

Many approaches focus on fusing camera information with a LiDAR-based detector to improve detection performance. MVX-Net (Sindagi *et al.*, 2019) augments VoxelNet with a separate image network. Pixel features from the image network are concatenated with the point features before being fed into the VFE layer. The matching between pixels and points is accomplished via known camera–LiDAR projection. Similarly, PointPainting (Vora *et al.*, 2019) augments the point features with the semantic class of their matching pixels. The semantic class is obtained via a pre-trained semantic segmentation network on images, *e.g.* DeepLabv3+ (Chen *et al.*, 2018; Zhu *et al.*, 2019b; Reda *et al.*, 2018). PointAugmenting (Wang *et al.*, 2021a) pops the camera image into 3D using LiDAR measurements, and processes the LiDAR points and the popped camera image using two separate 3D networks. The output feature volumes of these two networks are then combined via simple concatenation. MVP (Yin *et al.*, 2021b) segments object instances in the RGB image with a pre-trained CenterNetV2 (Zhou *et al.*, 2021), and generates virtual 3D points using the instance mask to create a denser point cloud around the objects. MMF (Liang *et al.*, 2019) fuses LiDAR and camera features, and combines detection with other complementary tasks like depth completion and ground plane estimation. 3D-CVF (Yoo *et al.*, 2020) focuses on multi-camera setups. It proposes an auto-calibration module to fuse image features from multiple cameras into a unified BEV view, and a gated fusion module to combine the LiDAR and image features. BEVFusion (Liu *et al.*, 2023a) projects the camera features to BEV utilizing the known extrinsic calibration between sensors, and combines LiDAR features and image features with a channel attention mechanism.

3.2.3 Networks with Point Features

Another line of 3D object detectors seeks to produce bounding boxes directly from raw points, without discretizing the point cloud into voxels or projected views. Early approaches attempt to guide the 3D detector with information from an image-based network. For example, Frustum-PointNet (Qi *et al.*, 2018) uses a 2D object detector on the RGB image to obtain a set of 2D box proposals. Points within the frustum of each 2D proposal are then collected and fed into a PointNet, which produces the final 3D bounding boxes. PointFusion (Xu *et al.*, 2018a) follows a similar approach, but additionally fuses the image feature with the point cloud feature in the frustum to produce the final 3D box. IPOD (Yang *et al.*, 2018c) finds foreground

points by using a semantic segmentation network on the RGB image, and creates 3D proposal boxes for all foreground points. Points within a proposal are fed into a PointNet to predict the refined box. The detection quality of these early methods heavily relies on the performance of the underlying 2D network, and they do not work in the absence of the RGB image or the calibration between the camera and the LiDAR sensor.

To remove the reliance on RGB images, PointRCNN (Shi *et al.*, 2019) proposes to directly generate proposal boxes from the LiDAR point cloud, following the architecture of the well-known Faster R-CNN detector (Ren *et al.*, 2015). It uses a PointNet++ (Qi *et al.*, 2017a) as the backbone to extract per-point features, and classifies each point into foreground (inside a ground-truth bounding box) or background. 3D box proposals are generated for each foreground point. Then, a second-stage refinement is conducted by passing the features of the foreground points within each proposal to a PointNet (Qi *et al.*, 2017b) to obtain the final detection boxes. Besides the main architecture, PointRCNN proposes a classification-based approach to encode the orientation of a 3D bounding box in order to remedy the challenge from angular periodicity. The approach splits the interval $[0, 2\pi)$ into twelve bins, and tasks the network to classify which bin the target angle belongs to, and regress the remainder between the bin center and the actual value. EPNet (Huang *et al.*, 2020) is an extension of the PointRCNN network, which introduces a separate image network to extract features from the RGB image, and fuses the image feature with the PointNet++ backbone.

3DSSD (Yang *et al.*, 2020) is a single-stage detector from raw point clouds that aims at high inference speed. It uses the downsampling branch of a PointNet++ as the backbone to extract per-point features, and the downsampled point features are passed into the detection head without upsampling. The detection head predicts, for each point, a centroid candidate, similar to VoteNet (Qi *et al.*, 2019), and for each centroid point, features from its neighboring centroid points are pooled and converted into a box detection, without an additional refinement stage. When processing large LiDAR scans (with hundreds of thousands of points), point-based approaches need to aggressively downsample the number of points in order to limit the memory consumption. Farthest point sampling (FPS) is often used to optimize the spatial coverage of the downsampled points. Yang *et al.* (2020) argue that naive FPS risks under-sampling the foreground points (points corresponding to an object), as most points in a LiDAR scan belong to the background. To resolve this problem, 3DSSD uses a novel feature-augmented FPS, where both spatial and feature distances are used when finding the next farthest point. IA-SSD (Zhang *et al.*, 2022) seeks to further increase the coverage of the foreground points by keeping points with the highest foreground score, computed using a point-wise MLP applied after the PointNet++ backbone.

Another line of point-based object detectors is designed for indoor datasets like ScanNet (Dai *et al.*, 2017) or SUN RGB-D (Song *et al.*, 2015), whose point clouds are generated by scanning static indoor scenes using RGB-D scanners. One popular work along this line is VoteNet (Qi *et al.*, 2019), which encodes the point cloud with a PointNet++ backbone, and generates a set of estimated box centroids, *i.e.* votes. Points whose estimated centroids are within a certain radius are then grouped together and converted into a final box prediction. VoteNet is the building block of many follow-up works along this line. ImVoteNet (Qi *et al.*,

2020) enriches the point cloud before voting with features from an image-based detector. 3D-MPA (Engelmann *et al.*, 2020) augments the grouping stage of VoteNet with a graph neural network for generating refined box proposals. HGNet (Chen *et al.*, 2020a) improves VoteNet by introducing a graph neural network among the proposals. H3DNet (Zhang *et al.*, 2020b) extends VoteNet by predicting multiple geometric primitives besides the box center, which are used for improved grouping and final prediction. MLCVNet (Qian *et al.*, 2020) introduces a multi-level context module to refine features extracted by the backbone, before passing them to the voting head. BRNet (Cheng *et al.*, 2021) traces the source points for each vote cluster, and fuses features from the traced source points. VENet (Xie *et al.*, 2021) introduces vote weighting and a new loss function to mitigate the distractions caused by background or nearby objects.

3.2.4 Hybrid Networks that Combine Points, Voxels, and Range

Whereas methods using discretized representations, such as voxels or range views, may suffer from information loss, directly operating on raw point clouds typically results in high memory usage or slow inference speed, due to the lack of structure in the representation. This shortcoming is especially relevant when working with more recent LiDAR sensors that reach sampling densities as high as hundreds of thousands of points per scan. To make the best out of voxel, range, and point representations, hybrid approaches seek to mix several representations in one network.

One popular recipe is to utilize both voxels and raw points in one network. Fast Point R-CNN (Chen *et al.*, 2019) uses a voxel-based backbone to generate 3D bounding box proposals, similar to VoxelNet (Zhou and Tuzel, 2017). The raw points within the proposals, as well as the features from the convolutional backbone, are then pooled together and passed into a refinement head to produce the final detection. PV-RCNN (Shi *et al.*, 2020b) takes a similar approach of using a voxel-based network to generate proposals and pool points within voxels for refinement. But instead of simply pooling the raw coordinates, it first extracts per-point features using a novel voxel set abstraction module. The module applies the set abstraction operation (from PointNet) to points within the same voxel, and is performed with multiple downsampling stages in the voxel backbone. Point features computed from this set abstraction module are then pooled with a customized grid pooling module, whose output is used to refine the box proposal from the voxel network. The voxel set abstraction module is performed on a set of representative points, sampled using farthest point sampling, in order to limit the memory consumption. Due to the sparsity of a 3D scene, naive FPS does not provide good coverage of the foreground points. For this reason, in a follow-up work, Shi *et al.* (2022b) propose a new sampling approach that excludes background points (the ones not near a box proposal). In addition, the new sampling approach segregates the scene into multiple non-overlapping areas, and conducts FPS for each area in a parallel fashion, to speed up the sampling process. STD (Yang *et al.*, 2019) is another hybrid 3D detector. Instead of using a voxel network as the backbone, it uses a PointNet++ backbone to generate bounding box proposals and extract per-point features. Then, points within each proposal box are voxelized

via a VFE operation, and the resultant volumetric feature is flattened and passed into a fully connected network to produce the final detection. HVPR (Noh *et al.*, 2021) augments the voxel-based PointPillars (Lang *et al.*, 2019) with point features extracted from PointNet++. Pair-wise similarities are computed between the sparse voxel features and the point features, and each voxel feature is enriched by a weighted average of its k most similar point features. In addition, a set of learnable feature memories is trained to mimic prototypical point features during training. During inference, this memory set is used in place of the point features, increasing the throughput by not running a forward pass on the expensive point encoder. LIDAR-RCNN (Li *et al.*, 2021b) uses proposals generated from the voxel-based PointPillars detector and proposes a point-based proposal refinement approach. It augments the points within a proposal with a set of virtual points, encoding the relative location within the proposal, and applies a PointNet on the augmented points. Graph R-CNN (Yang *et al.*, 2022a) similarly proposes point-based refinement, but the refinement is predicted using a graph neural network (Wang *et al.*, 2019c) on points within the proposal box.

Range is another often-used representation in hybrid networks. RangeRCNN (Liang *et al.*, 2020b) extracts per-point features from the range view representation, and converts the point features to a BEV grid to make bounding box predictions. A follow-up work, RangeIoUDet (Liang *et al.*, 2021), introduces an auxiliary point-based network during training, which is used to convert the point-wise features into a segmentation mask. A point-based auxiliary loss is computed on the segmentation mask to guide the learning process. RSN (Sun *et al.*, 2021) uses a 2D network on the range view of the point cloud to segment the foreground points. A point-based network is then applied to the foreground points to predict bounding boxes.

3.2.5 Fully Sparse Voxel Networks

The voxel methods discussed in Section 3.2.2 use a sparse voxel-based network to encode the scene, and convert sparse voxel features to a dense BEV feature map for further processing. This sparse-to-dense scheme incurs wasted memory and computation, as the dense BEV feature map contains a large number of empty grids due to scene sparsity. To further improve computational efficiency, several approaches propose to abandon the intermediate 2D dense feature map and use sparse voxel representations in an end-to-end fashion, reducing the complexity to $O(n)$ with respect to the number of voxels. The reduced complexity is especially significant when working with large LiDAR scans like the ones in the Argoverse 2 dataset (Wilson *et al.*, 2021), which extend as far as 200 meters.

Several works have used 3D U-Nets built with submanifold sparse convolution (Graham *et al.*, 2018) for segmentation tasks (Choy *et al.*, 2019a; Tang *et al.*, 2020). GSDN (Gwak *et al.*, 2020) re-purposes the 3D U-Net for object detection in indoor scenes. Person-MinkUNet (Jia and Leibe, 2021) uses a similar architecture for person detection with LiDAR data. The 3D U-Net extracts voxel features, and outputs, for each voxel, a 3D bounding box. Non-maximum suppression is performed to directly obtain the final detections without a second refinement stage. FCAF3D (Rukhovich *et al.*, 2022) improves GSDN (Gwak *et al.*, 2020) with a multi-scale target assignment strategy based on bounding box size, and a new angular

parameterization. A follow-up work, TR3D (Rukhovich *et al.*, 2023), introduces improvements to the network architecture and loss function, and includes experiments with fusing image features.

A few recent works employ end-to-end sparse voxel networks on challenging driving datasets and obtain state-of-the-art results for multi-class object detection. FSD (Fan *et al.*, 2022b) takes SST (Fan *et al.*, 2022a) as the sparse voxel backbone, and predicts, for each voxel, an instance center. These instance centers are then grouped based on connected components, where two centers with a distance less than a threshold are considered as connected. A box proposal is generated for each instance group by passing voxel features belonging to the group through a point network. To correct potential mis-grouping, a second-stage refinement is performed, where voxels within a proposal are grouped, disregarding their connectivity. To address the computation overhead related to aggregating consecutive frames, FSD++ (Fan *et al.*, 2023b) proposes to include only points whose quantized coordinates are unoccupied in the previous frames and to subsample foreground points, keeping the number of voxels close to that of a single frame. Instead of using instance grouping, FSDv2 (Fan *et al.*, 2023a) proposes to regard the predicted centers as virtual points and add them to the point cloud. The augmented point cloud, along with extracted features, is then re-voxelized, and the virtual voxels are used to predict box proposals. VoxelNeXt (Chen *et al.*, 2023c) identifies that the lack of a receptive field is a major bottleneck for directly predicting bounding boxes with sparse voxel features, and in turn proposes to append a typical 3D sparse voxel encoder with additional downsampling blocks. This adapted voxel encoder directly outputs a bounding box for each voxel, without converting the sparse voxel features to a dense feature map. Besides the fully sparse backbone network, VoxelNeXt introduces a spatial pruning module, which increases the overall inference speed by reducing the number of background voxels, and proposes to use max pooling on the sparse feature volume to replace non-maximum suppression, following similar approaches to the center-based detection head (Yin *et al.*, 2021a).

3.2.6 Networks with Transformer Architecture

Transformer (Vaswani *et al.*, 2017), originally proposed for processing sequential data like text, has recently emerged as a powerful architecture for computer vision (Dosovitskiy *et al.*, 2021b; Carion *et al.*, 2020; Liu *et al.*, 2021b). Motivated by its success in image tasks, many recent works seek to apply transformers or their variants to 3D object detection. The core of the transformer is the attention operation, which takes as input a set of features (known as *tokens*) irrespective of their ordering. This invariance to permutation and the large receptive field make the attention operation an especially promising candidate for processing point clouds.

Many early works seek to reap the benefits of the attention operation by augmenting a well-established detector with customized attention modules. VoTr (Mao *et al.*, 2021b) takes the 3D encoder from SECOND (Yan *et al.*, 2018) and replaces its sparse 3D convolution with customized attention modules. Two types of attention, local attention and dilated attention, are used in the customized module to compute self-attention between voxel features on the respective kernels, analogous to convolution. CT3D (Sheng *et al.*, 2021) takes SECOND as a

region proposal network and proposes a transformer-based proposal refinement module. The module operates by feeding points within a proposal through a transformer with an encoder and a decoder. The encoder computes self-attention among the point features, and the decoder computes cross-attention between one query embedding and the point features. The resulting value vector from the decoder is converted into proposal refinement. The decoder converts a set of object queries into boxes by computing cross-attention with the BEV feature. M3DETR (Guan *et al.*, 2021) extracts point features via PointNet++ (Qi *et al.*, 2017a), voxel features via VoxelNet (Zhou and Tuzel, 2017), and BEV features via a 2D network, and fuses multi-representation features via a customized attention module. Similarly, VISTA (Deng *et al.*, 2022) proposes to fuse features extracted from BEV and range views using an attention-based mechanism. VoxSeT (He *et al.*, 2022a) applies the set attention operation (Lee *et al.*, 2019) to points within a voxel to extract voxel features. The voxel features are then converted to a dense BEV feature map and processed with an image-based network, similar to SECOND or PointPillars. GHA (Jia *et al.*, 2022b) adds a novel global hierarchical attention operation to existing detectors to capture long-range information. The operation computes local attention on a series of resolution hierarchies, approximating self-attention while having $O(n)$ complexity.

New detectors inspired by the full transformer architecture have emerged in recent developments that go beyond augmenting existing detectors. A suite of works follows the strategy laid out by the seminal DETR detector in the image domain (Carion *et al.*, 2020). The overall architecture can be summarized in three steps. A feature extractor (*e.g.* a CNN) is first applied to the input image. The extracted features are then passed into a transformer encoder composed of several layers of self-attention. Finally, the decoder, composed of several layers of self-attention and cross-attention, converts a set of object queries into bounding boxes. The self-attention in the decoder is computed among object queries, and the cross-attention is between object queries and the encoder features. During training, the decoded bounding boxes are assigned to the ground truth via Hungarian matching (Kuhn, 1955) for supervision. 3DETR (Misra *et al.*, 2021) is the first end-to-end 3D object detector inspired by DETR. It extracts point features using PointNet for a subset of seed points and passes these features into a transformer encoder. A set of object queries is then decoded into 3D bounding boxes by cross-attention with the point features. A concurrent work, Object DGCNN (Wang and Solomon, 2021), extracts dense BEV features via SECOND or PointPillars (Lang *et al.*, 2019), and uses a DETR-like decoder to obtain a set of bounding boxes. CenterFormer (Zhou *et al.*, 2022) pairs the centerness heatmap from CenterPoint (Yin *et al.*, 2021a) with a DETR decoder to predict bounding boxes. PVT-SSD (Yang *et al.*, 2023b) extracts voxel features using sparse convolution on the point cloud, and proposes a point-voxel transformer to decode object queries into bounding boxes, which computes cross-attention with both nearby voxels and points within the voxel in order to obtain contextual information and fine-grained details, respectively. ConQueR (Zhu *et al.*, 2023) encodes the point cloud into dense BEV features via 3D sparse convolution, and applies a transformer decoder to predict bounding boxes. The decoder resembles the decoder in DETR, but instead of using a set of randomly initialized learnable embeddings, ConQueR evaluates object scores for the tokens from the encoder us-

ing a feed-forward network, and simply selects k tokens with the highest scores as the object queries. In addition, it applies a query contrast mechanism during training, encouraging the queries to spread out and cover the entire scene.

Another group of works takes inspiration from the image-based Swin Transformer (Liu *et al.*, 2021b), with the goal of building, using the attention operation, a general-purpose backbone that extracts point cloud features which can be used for a suite of downstream tasks including object detection. SST (Fan *et al.*, 2022a) partitions sparse voxels into non-overlapping windows, and computes self-attention between points within voxels. The partitioning pattern shifts based on a predefined cycle from one layer to the next, allowing information exchange across different windows. After the backbone, composed of several SST blocks, the sparse voxel features are scattered to a dense BEV feature map and passed to an image-based detection head. SWFormer (Sun *et al.*, 2022) employs a similar idea of computing local attention within alternated partitioning windows, and adds multi-scale fusion in the backbone. In addition, it proposes segmenting and dilating foreground voxels before predicting bounding boxes, in order to have voxel features close to the object center. Motivated by the observation that the partition windows contain an uneven number of voxels, which leads to slow computational efficiency, FlatFormer (Liu *et al.*, 2023b) and DSVT (Wang *et al.*, 2023b) propose first sorting the voxels based on coordinates, and then partitioning the sorted voxels into equal-sized groups. x and y coordinates are used for sorting alternately, mimicking the window-shifting operation. GD-MAE (Yang *et al.*, 2023a) processes sparse BEV pillar features with a backbone network built with local attention and submanifold sparse convolution, and proposes a self-supervised pretext training task similar to the Masked Autoencoder (He *et al.*, 2022b) for images. The pretext task asks the network to reconstruct certain areas that are purposefully masked out.

The attention mechanism provides a learnable way of fusing features from different sources (Akbari *et al.*, 2021). Several works use attention for fusing LiDAR and camera features. DeepFusion (Li *et al.*, 2022a) takes existing LiDAR-based detectors, PointPillars (Lang *et al.*, 2019), CenterPoint (Yin *et al.*, 2021a), and 3D-MAN (Yang *et al.*, 2021), and adds cross-attention between LiDAR features and features from RGB cameras. TransFusion (Bai *et al.*, 2022) uses a 3D backbone to extract dense BEV features from a voxelized point cloud, and predicts bounding box proposals using a DETR decoder with a sampled set of object queries. In addition, an image network is used to extract image features from RGB cameras, and cross-attention is computed between object queries and the image features to refine the box proposals. A Gaussian kernel based on spatial distance is used to weight the cross-attention, encouraging the LiDAR-based query vectors to focus on nearby image features. UVTR (Li *et al.*, 2022b) proposes converting image features into the voxel grid of LiDAR features by predicting the depth distribution. Features from different modalities are then combined in the unified voxel space, and a transformer decoder akin to that of Deformable DETR (Zhu *et al.*, 2021) is used to predict bounding boxes. The attention mechanism can also be used for fusing temporal features. 3D-MAN (Yang *et al.*, 2021) uses a single-frame detector to predict box proposals, and keeps the proposals and the associated features in a memory bank. Cross-

attention is computed between current proposal features and those from previous frames, and the temporally enriched features are used to make the final prediction.

Beyond detecting objects in LiDAR point clouds, transformers and their variants have also been used with indoor or synthetic 3D data for a wide range of tasks. Here we provide a few examples. GroupFree (Liu *et al.*, 2021a) adds, on top of VoteNet (Qi *et al.*, 2019), several stacked attention layers to extract features with stronger global context for object detection. Pointformer (Pan *et al.*, 2021) and Point Transformer (Zhao *et al.*, 2021) build a PointNet++ (Qi *et al.*, 2017a) architecture, but replace the set abstraction with attention-based operations. Fast Point Transformer (Park *et al.*, 2022) replaces the point representation in Point Transformer with voxel representations for computational efficiency and uses learned devoxelization to recover the original resolution. Stratified Transformer (Lai *et al.*, 2022) augments the local attention in Point Transformer with sampled distant points, incorporating long-range information. Spherical Transformer (Lai *et al.*, 2023) computes local attention within windows parameterized in spherical coordinates, observing the structure of LiDAR scans. P4Transformer (Fan *et al.*, 2021a) uses attention to model point cloud video for action recognition. DETR3D (Wang *et al.*, 2021c) uses transformers for 3D object detection with a multi-view camera setup, where the object queries attending to the image features are decoded into 3D boxes.

3.3 Dataset for LiDAR-based Person Detection

Labelled data are indispensable for developing person detectors that rely on supervised deep learning, and the quality of training data heavily impacts the performance of such detectors. Over the past decade, significant effort has gone into collecting high-quality, large-scale datasets for detecting persons with LiDAR sensors, many of which are made publicly available for research purposes in order to facilitate development and comparison of algorithms on a common basis. Some popular datasets are listed in Table 3.2.

Many of the existing LiDAR datasets with person annotations are collected using sensors mounted on vehicles driving on urban streets. These driving datasets are not always suited for developing person detectors to be used on mobile robots, due to the large domain gap between driving and mobile robotic scenarios. Compared to vehicles, mobile robots encounter more crowded environments with a higher number of persons per LiDAR scan, as shown in Table 3.3. In addition, in mobile robotic applications, most persons are in closer proximity to the LiDAR sensors, as shown in Figure 3.1. The inherent biases in the datasets can mislead the detector toward spurious correlations that are present only in the training data (*e.g.* persons always appear on pedestrian paths), hurting the detector's generalization ability to other scenarios (*e.g.* in an audience hall).

Table 3.2: Publicly available datasets with person annotations collected using LiDAR sensors on different sensor platforms.

Dataset	Year	LiDAR		Platform
		2D	3D	
KITTI (Geiger *et al.*, 2013)	2012		✓	Vehicle
DROW (Beyer *et al.*, 2018)	2018	✓		Mobile robot
Argoverse (Chang *et al.*, 2019)	2019		✓	Vehicle
A*3D (Pham *et al.*, 2020)	2019		✓	Vehicle
H3D (Patil *et al.*, 2019)	2019		✓	Vehicle
Lyft Level 5 (Kesten *et al.*, 2019)	2019		✓	Vehicle
JRDB (Martin-Martin *et al.*, 2021)	2019	✓	✓	Mobile robot
nuScenes (Caesar *et al.*, 2020)	2019		✓	Vehicle
Waymo (Sun *et al.*, 2020)	2019		✓	Vehicle
A2D2 (Geyer *et al.*, 2020)	2020		✓	Vehicle
CADC (Pitropov *et al.*, 2021)	2020		✓	Vehicle
Cirrus (Wang *et al.*, 2020c)	2020		✓	Vehicle
RADIATE (Sheeny *et al.*, 2020)	2020		✓	Vehicle
PandaSet (Xiao *et al.*, 2021)	2020		✓	Vehicle
ONCE (Mao *et al.*, 2021a)	2021		✓	Vehicle
Argoverse 2 (Wilson *et al.*, 2021)	2021		✓	Vehicle
KITTI-360 (Liao *et al.*, 2022)	2021		✓	Vehicle
AIODrive (Weng *et al.*, 2021)	2021		✓	Vehicle
ZOD (Alibeigi *et al.*, 2023)	2023		✓	Vehicle

Table 3.3: Number of persons in LiDAR datasets recorded for different scenario.

Dataset	Scenario	Number of Persons	
		Total	Per-frame
KITTI (Geiger *et al.*, 2013)	Driving	4.49×10^3	0.59
nuScenes (Caesar *et al.*, 2020)	Driving	6.12×10^4	1.79
Waymo (Sun *et al.*, 2020)	Driving	1.75×10^6	7.68
DROW (Beyer *et al.*, 2018)	Mobile robotics	2.90×10^4	1.21
JRDB (Martin-Martin *et al.*, 2021)	Mobile robotics	8.15×10^5	29.17

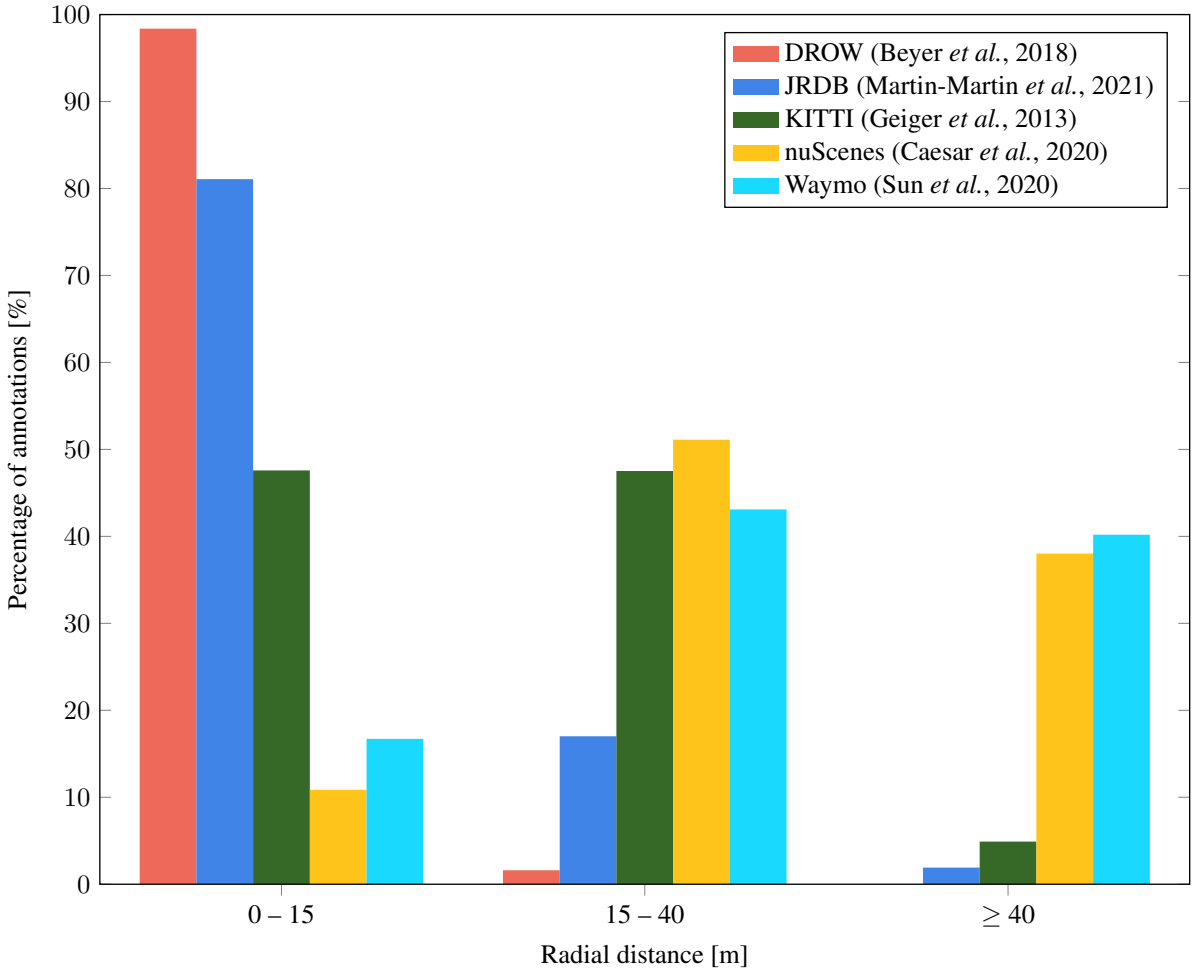


Figure 3.1: Percentage of person annotations grouped by different proximity to the sensor. Persons are closer to the sensor in the mobile robotic datasets (DROW and JRDB), and are more spread out in the driving datasets (KITTI, nuScenes, and Waymo).

The remainder of this section briefly reviews several influential public datasets, which are closely related to the development of various methods discussed in this thesis.

DROW (Beyer *et al.*, 2018). The DROW dataset is recorded using a SICK S300 laser scanner in an elderly care facility. The laser scanner is mounted on a mobile robot at 37 cm above the ground, and scans a 225° field of view with a 0.5° angular resolution at a 12.5 Hz sampling rate. A total of 10 hours of recordings are collected and are split into small sequences. Every fifth frame in a selected subset of sequences is annotated, resulting in a total of 5% of laser scans being annotated. In the initial release (Beyer *et al.*, 2016), only wheelchairs and walking aids are annotated, and a later extension (Beyer *et al.*, 2018) adds persons to the annotated objects. The annotation includes the 2D location of the object in the BEV plane and the class labels, but does not provide the size of the object or the bounding box. The dataset features an official

split into training, validation, and testing sets based on the recording areas, in order to evaluate the network's capability to generalize to new environments.

The DROW dataset is the first public dataset that is successfully used for training deep-learning-based object detectors. It is recorded in a natural environment and features a large amount of annotated scans. Prior datasets are not sufficient for training deep neural networks, due to their limited amount of data (Weinrich *et al.*, 2014), static LiDAR scanners (Luber *et al.*, 2011), or the usage of artificially constructed scenes, such as having one person walking around a pre-mapped empty room to obtain training samples (Leigh *et al.*, 2015).

JRDB (Martin-Martin *et al.*, 2021). The JackRabbot Dataset and Benchmark (JRDB) is collected using a mobile robot moving on university campuses. The robot is equipped with a suite of sensors, including two Velodyne 16 Puck LITE LiDAR sensors, two SICK LMS500-20000 laser scanners, one spherical panorama rig of ten cameras, and one RGB-D stereo camera. The two Velodyne 3D LiDAR sensors are mounted on the upper and lower parts of the robot, respectively, each producing scans with approximately 18,000 points. The two SICK 2D laser scanners are mounted at the height of the lower legs, facing front and rear, respectively, and their scans are merged into a single 360° scan with 1,091 points. The availability of both 3D and 2D LiDAR sensors enables the comparison of detection algorithms across sensor modalities.

The JRDB dataset features a significantly larger scale than the earlier DROW dataset. It contains 54 sequences, separated into 27 sequences for training and validation, and 27 for testing. The recordings cover a wide range of scenarios, including both indoor and outdoor environments. Persons are annotated with 3D bounding boxes, and the dataset contains a total of roughly 1.8M annotations. At the time of writing this thesis, JRDB is the only large-scale dataset that focuses on mobile robot scenarios, while also featuring LiDAR point clouds with 3D person annotations.

KITTI (Geiger *et al.*, 2013). The KITTI dataset is recorded using a sensor-equipped vehicle driving on urban streets. The sensor suite includes an IMU, two RGB cameras, and a Velodyne HDL-64E 3D LiDAR sensor. Using the recorded data, benchmarks are constructed for a wide range of vision tasks, including optical flow, visual odometry, 2D/3D object detection, and semantic segmentation. The KITTI dataset is a pioneer in evaluating visual recognition tasks for autonomous driving applications, and has enabled comparisons of hundreds of methods since its release.

For the task of object detection, the KITTI dataset features 7,481 samples for training and validation, and 7,518 samples for testing. Three classes of objects (*cars*, *pedestrians*, *cyclists*) are labelled with 3D bounding boxes, and there are a total of 80,256 annotations. The annotation only covers objects within the front-facing camera frustum, not the entire LiDAR scan.

nuScenes (Caesar *et al.*, 2020). Compared to the earlier KITTI dataset, the nuScenes dataset features a greater number of samples and covers more diverse scenarios. The sensor suite includes six cameras, five radars, and a 32-beam LiDAR sensor, and the data are recorded in Boston and Singapore. The nuScenes dataset contains 1,000 short sequences of approximately

20 seconds, split into training, validation, and test sets of 700, 150, and 150 sequences, respectively. The 32-beam LiDAR sensor used in recording produces scans with roughly 30,000 points at 20 Hz. Every tenth frame is annotated with 3D bounding boxes, covering a total of ten object classes, including *pedestrians*. The annotation covers the entire 360° LiDAR field of view. Since the nuScenes dataset provides LiDAR scans in sequential order, it is a common practice to accumulate consecutive scans into a dense point cloud during training and inference for improved performance (Caesar *et al.*, 2020; Zhu *et al.*, 2019a; Yang *et al.*, 2020; Yin *et al.*, 2021a).

Waymo (Sun *et al.*, 2020). The Waymo Open Dataset (WOD) is another well-known large-scale dataset aiming at developing autonomous driving algorithms suitable for diverse scenarios. It is recorded in San Francisco, Mountain View, and Phoenix using vehicles equipped with five cameras, four short-range LiDAR sensors, and one mid-range LiDAR sensor. The dataset contains 1,150 short sequences, each spanning 20 seconds. Four classes of objects (*vehicles*, *pedestrians*, *cyclists*, *signs*) are densely annotated in the 360° LiDAR scans, resulting in 12.6M bounding boxes and 230K annotated frames.

4 DR-SPAAM: A Spatial-Attention and Auto-regressive Model for Person Detection in 2D Range Data

4.1 Introduction

Detecting persons in the surrounding environment is a crucial requirement for many robotic applications, including search and rescue, security, and healthcare. Typically, this is achieved through the use of multiple RGB(-D) cameras in combination with deep learning-based object detectors (Girshick, 2015; He *et al.*, 2017; Redmon and Farhadi, 2018). However, the limited field of view of cameras restricts detection to a narrow frustum. Additionally, the depth measurement from RGB-D cameras is often inaccurate at far distances, which can result in imprecise detection outcomes.

Contrary to RGB-D cameras, 2D LiDAR sensors, commonly equipped on mobile robots for navigation and mapping purposes, provide accurate range measurements with a large field of view at high acquisition rates. These advantages make them a promising sensor choice for person detection. Previous attempts at developing 2D-LiDAR-based person detector employed hand-crafted features with a learned classifier (Arras *et al.*, 2007; Leigh *et al.*, 2015). More recent developments have leveraged deep learning techniques to achieve state-of-the-art detection performance (Beyer *et al.*, 2018; Jia *et al.*, 2020).

The limited information contained in the sparse range measurements from a 2D LiDAR scan is a key challenge towards reliable person detection. To remedy the problem caused by sparse scans, Beyer *et al.* (2018) propose to accumulate consecutive scans in order to get a stronger prediction signal, and showed improved detection results comparing to using only a single frame. The accumulation is accomplished using an expensive alignment operation, which compensates sensor ego-motion and the motion of objects in the scene. The downside,

This chapter is based on the 2020 IROS conference paper titled *DR-SPAAM: A Spatial-Attention and Auto-regressive Model for Person Detection in 2D Range Data* by Jia, Hermans, and Leibe. I was responsible for the method design, experiment evaluation, and writing. Hermans and Leibe contributed through regular discussions and revisions.

however, is the reduced inference throughput, making the overall pipeline too expensive for real-time applications on mobile platforms. Besides, the odometry information needs to be available for performing the alignment.

Aligning and fusing previous scans follows a backward looking paradigm for aggregating temporal information. In contrast, a forward looking paradigm simply keeps a representation based on current measurements and recurrently updates this representation when a new measurement becomes available. An example of such a forward looking paradigm is found in the field of video object detection, where memory modules are used to recurrently take input features from individual consecutive frames (Tripathi *et al.*, 2016; Lu *et al.*, 2017; Xiao and Lee, 2017).

In this chapter we discuss a novel person detector, termed Distance Robust SPatial Attention and Auto-regressive Model (DR-SPAAM), which aggregates temporal information following a forward looking paradigm. It augments the successful DROW detector (Beyer *et al.*, 2018) with an auto-regressive model to aggregate the intermediate features from the backbone network from sequential scans, propagating information forward through time. In addition, instead of explicitly aligning sequential scans, it uses a spatial attention mechanism to associate features from neighboring locations based on their similarity, significantly reducing the runtime of the detector. Evaluation on DROW dataset shows that the proposed forward-looking aggregation module improves the performance of a single frame baseline while adding little computational cost. The high throughput of DR-SPAAM makes it ideal for mobile robotic applications.

In summary, the main contribution of this chapter is following:

- We propose a novel SPatial Attention and Auto-regressive Model (SPAAM) that fuses information from previous LiDAR scans without the need of explicit alignment operation.
- We build DR-SPAAM, a fast 2D-LiDAR-based person detector, using the proposed aggregation module. DR-SPAAM establish a new state-of-the-art in person detection using 2D range data while having high inference throughput for real-time applications.
- We release our code, implemented in PyTorch with an easy-to-use detector ROS node, for robotic applications.[1]

4.2 Related Work

The forward-looking temporal integration draws inspiration from works in the field of video object detection. Video object detection aims to detect single or multiple objects in a video sequence, for which temporal propagation of information is important, since objects can undergo large appearance changes caused by fast motion, occlusion, or change of camera angle.

[1] https://github.com/VisualComputingInstitute/2D_lidar_person_detection

Earlier approaches in the field (Kang *et al.*, 2016; Han *et al.*, 2016; Feichtenhofer *et al.*, 2017) detect objects in each frame independently and apply sequence-level post-processing on the obtained bounding boxes. Such approaches cannot be optimized in an end-to-end fashion. Later approaches focus on directly aggregating features across frames, either explicitly aligning features using optical flow (Zhu *et al.*, 2017b,a; Wang *et al.*, 2018b), or using a memory network to aggregate features (Tripathi *et al.*, 2016; Lu *et al.*, 2017; Xiao and Lee, 2017).

Similar to these approaches, DR-SPAAM uses a network to aggregate features across consecutive scans. Instead of using a designated memory module, we use an auto-regressive model, which propagates information from the previous scan to the next with an exponentially decaying weight. In video object detection, the key is to be able to aggregate long term features, since neighboring frames often have similar appearance and introduce little new information. Thus, a memory network is often used. Our focus, however, is on aggregating short term features from consecutive scans, with the goal of enriching the information available for detection. Hence, an auto-regressive model is more suitable compared to a more complex memory module. Furthermore, we propose to use a similarity-based spatial attention model (Wu *et al.*, 2019a; Vaswani *et al.*, 2017) to first fuse nearby spatial information before the temporal aggregation.

4.3 Method

In this section, we provide a detailed description of our proposed method for person detection. First, we describe the DROW detector (Beyer *et al.*, 2016, 2018), which we utilize as the single-frame baseline. Next, we discuss different paradigms of aggregating temporal information. Finally, we introduce DR-SPAAM, our novel detector that enhances the single-frame baseline with a forward-looking aggregation module and achieves state-of-the-art performance.

4.3.1 DROW Detector

The DROW detector (Beyer *et al.*, 2016, 2018) is the first deep-learning-based approach that detects persons from 2D range data. It consists of three stages. First, the raw scans are preprocessed into small per-point windows, which are referred to as *cutouts*, in order to normalize the appearance across different depths. These cutouts are then separately classified by a network as either object or background, and a possible object center is regressed for every cutout. Finally, all regressed object centers, referred to as *votes*, are collected and aggregated to a final set of detections.

The input to the detector is a 2D LiDAR scan composed of N points, expressed in polar coordinate as $\{(r_n, a_n)\}_{n=0}^{N-1}$. (The measurements are made at a fixed angular grid, and r_n stands for the range measured at the n-th angle.) The goal of the preprocessing step is to

isolate, for each point, its neighboring points that fall in a fixed-size window in Euclidean space. To accomplish this, we first compute an angular opening Δa_n for each point using

$$\Delta a_n = \arctan \frac{0.5 \cdot \Delta W}{r_n}, \tag{4.1}$$

with ΔW being a hyperparameter specifying the width of the window we would like to isolate in Euclidean space. Then the cutout C_n is generated by taking neighboring points that fall within the angular window

$$C_n = \{(r_m, a_m) | m = 0, \dots, N-1, |a_m - a_n| \leq \Delta a_n\}. \tag{4.2}$$

These points are then re-sampled to a fixed number of M points, centered by subtracting the range r_n of the central point, clipped to within $\pm\Delta r$ to eliminate background and foreground points, and normalized to $[-1, 1]$. M and Δr are hyperparameters.

The cutout preprocessing alleviates problems caused by unequal sampling densities at different distances (LiDAR points are sparser at far range), and allows the DROW detector to work with LiDAR sensors with different angular resolutions without requiring re-training. Furthermore, the clipping operation removes the background information, allowing the network to focus on the neighboring points that are in the same distance region.

The normalized cutouts obtained from the pre-processing step are passed through a 1D CNN. The network outputs a confidence score for classification, as well as the 2D bird's-eye view location of the person, expressed as an offset from the LiDAR point. The original DROW detector includes a voting mechanism to group predictions into detections. Here we opt for a simpler non-maximum-suppression (NMS) approach, where predictions closer than a defined distance are considered duplicate (*i.e.* they are from the same person) and only the most confident one is kept.

4.3.2 Temporal Information Aggregation

Due to the sparsity of LiDAR scans (especially at far range), it is common to accumulate few consecutive scans into a denser point cloud. Operating on such a denser point cloud usually leads to improved detection performance (Beyer *et al.*, 2018; Zhu *et al.*, 2019a). This accumulation strategy, which fuses measurements for the past few steps, follows the backward looking paradigm. Spatial misalignment often exists between these measurements, due to the sensor ego-motion or dynamic objects, and this misalignment has to be corrected based on odometry or point cloud registration.

Similarly, the DROW detector accumulates temporal information by looking backward (Beyer *et al.*, 2018). It computes cutouts on the past T scans $\{C_n^{t-T+1}, \cdots, C_n^t\}$ and fuses features $\{F_n^{t-T+1}, \cdots, F_n^t\}$, obtained from an intermediate stage of the network, by a simple summation. The fused features are then fed into the later stage of the network for classification and regression. Due to the ego-motion of the sensor, two range measurements s_n^t and s_n^{t-1} made at the same angular index n will not correspond to a single aligned point in the

world, and the DROW detector uses robot odometry to correct the misalignment before fusing the features F_n^t and F_n^{t-1}. However, odometry alone is not sufficient to compensate for the misalignment caused by dynamic objects in the scene. In the case of persons, this is especially critical, since the LiDAR ray at the same n^{th} angular index could hit the leg of a person at time $t-1$, while passing between the legs and hitting a distant background structure at time t, resulting in significantly different features. Beyer *et al.* (2018) propose to fix the sampling location at which the cutout is centered to the location of the current point s_n^t. This means the cutouts of previous scans need to be recomputed at each time step. With the alignment using odometry and fixed location sampling, the DROW detector combines five scans from the past, resulting in improved detection accuracy compared to using only a single scan.

The performance gain from such a backward looking approach comes at the cost of increased computation time. The misalignment between the current measurement and each previous measurement has to be corrected, resulting in a linear increase in computation time with respect to the number of frames in the aggregation window. For the DROW detector, using only five scans already makes the overall detection pipeline too expensive for real-time applications on mobile platforms.

An alternative approach to aggregate temporal information is to follow a forward looking paradigm. Instead of explicitly aligning and combining multiple previous measurements, a forward looking approach simply keeps a representation based on the current measurements and recurrently updates the representation for each new measurement. Ideally, the update step only incurs a small computational overhead. As a result, a forward looking approach aggregates information from the past without the unfavorable runtime scaling behavior with respect to the size of the temporal window.

4.3.3 DR-SPAAM Detector

We propose the Distance Robust SPatial-Attention and Auto-regressive Model (DR-SPAAM), which follows a forward looking paradigm to aggregate temporal information. The network architecture is shown in Figure 4.1. Instead of re-performing the preprocessing step on the past scans to obtain spatially aligned cutouts, we use a similarity-based spatial attention module (Vaswani *et al.*, 2017), which allows the network to learn to associate misaligned features from a spatial neighborhood. Additionally, an auto-regressive model is used to update the representation, aggregating information forward through time. Our proposed detector outperforms the DROW detector, while being approximately four times faster. Figure 4.1 offers an overview of the proposed DR-SPAAM architecture.

Due to the misalignment, features F_n^t and F_n^{t-1} computed at two time steps cannot be naively combined. Instead of explicitly modeling the alignment as in (Beyer *et al.*, 2018), we propose to let the network learn to associate features using a similarity-based attention mechanism. For a point s_n^t, we look at its spatial neighbors $\{s_{n-w}^{t-1}, \cdots, s_{n+w}^{t-1}\}$ at previous time $t-1$ and compute a pairwise similarity

$$\Omega_{nj} = \psi(F_j^{t-1})^T \cdot \psi(F_n^t) \tag{4.3}$$

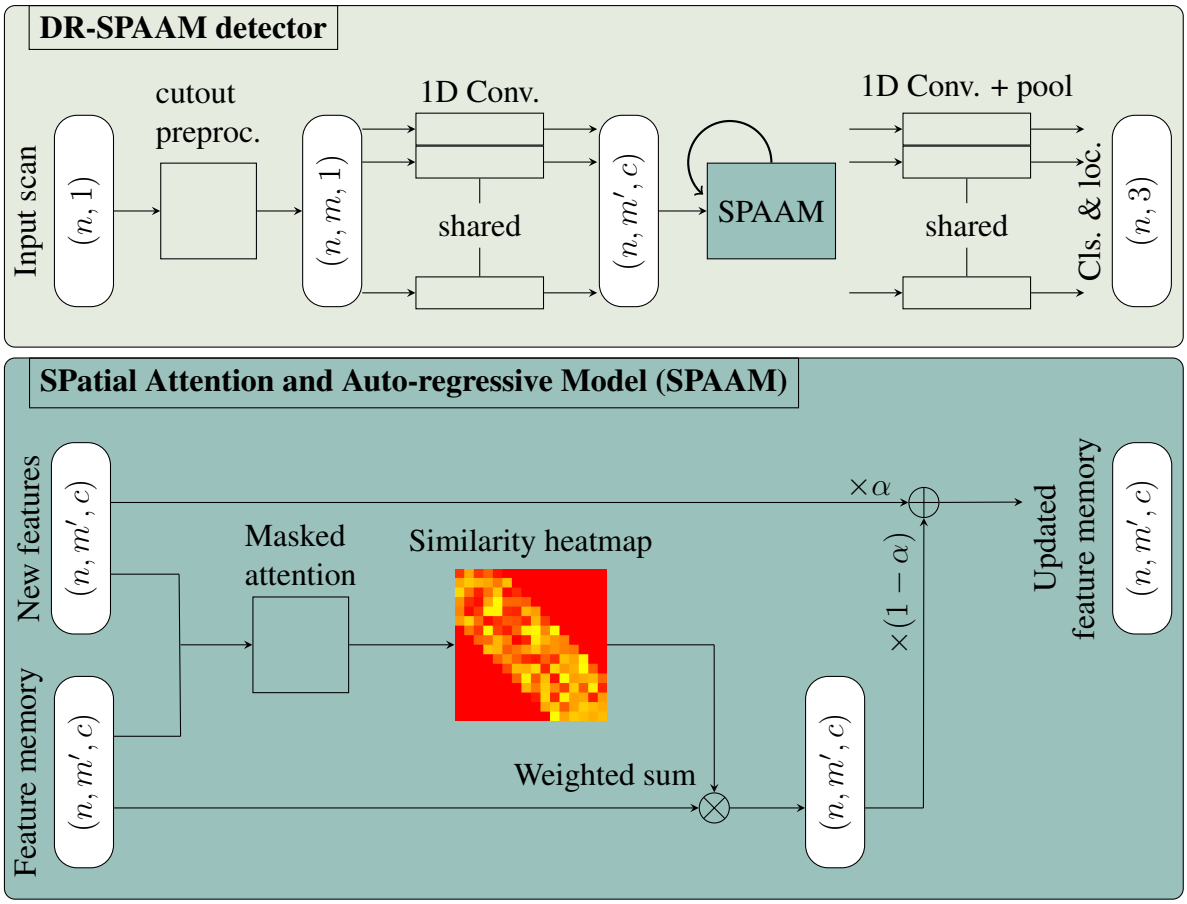


Figure 4.1: Top: Overview of the DR-SPAAM detector. The LiDAR scan is used to generate cutouts for each point, which are processed independently with a series of 1D convolutions (see Section 4.3.1). The SPAAM module is then applied to the intermediate features to aggregate temporal information from previous scans (see Section 4.3.3). Based on the temporal enriched features, the network outputs a classification label and predicts a relative object center for each point. **Bottom:** Overview of the proposed SPatial Attention and Auto-regressive Model. The SPAAM module keeps a feature memory. At each new frame, a similarity is computed for each new feature against its neighboring features in the memory using a masked spatial attention mechanism. A temporal enriched feature is then computed by taking a weighted sum over the feature memory using the computed similarity. Finally, the temporal enriched feature is combined with the new features in an auto-regressive fashion to enrich the features with temporal information.

between the features extracted at each previous point s_j^{t-1} and those from the current point s_n^t. Here, w is a parameter defining the size of the neighborhood, F_n^t is the intermediate feature of s_n^t, and ψ is a generic mapping function, realized by a neural network, that maps the feature to an embedding space. We then use a softmax function to convert the similarity into weighting factors and produce fused features $\tilde{F}_n^{t-1}$ from the previous frame:

$$\tilde{F}_n^{t-1} = \sum_{j=n-w}^{n+w} softmax(\Omega_n)_j F_j^{t-1}. \tag{4.4}$$

This model gives more weight to the points with a higher similarity score, which are more likely to contain information from regions near s_n^t, while suppressing features from other irrelevant points. The fused feature $\tilde{F}_n^{t-1}$ from the previous frame can then be combined with the current feature F_n^t and be used for further processing.

This model only combines information from two consecutive scans. In order to aggregate information from previous scans further back in the past, we propose to combine Equation 4.4 with an auto-regressive approach. We treat the fused features $\tilde{F}_n^{t-1}$ from time $t-1$ as a template, and when the new features F_n^t at time t become available, we compute an updated template:

$$\tilde{F}_n^t = \alpha F_n^t + (1-\alpha) \sum_{j=n-w}^{n+w} softmax(\tilde{\Omega}_n)_j \tilde{F}_j^{t-1}, \tag{4.5}$$

where $\alpha \in [0, 1)$ is a parameter that controls the update rate. Here the first term is our update to the stored template, and the second term summarizes the information from the past. Notice that unlike in Equation 4.3, the term $\tilde{\Omega}_{nj}$ here denotes the similarity between the current feature F_n^t and the neighboring features $\{\tilde{F}_{n-w}^{t-1}, \cdots, \tilde{F}_{n+w}^{t-1}\}$ from the previous templates, rather than from the previous scan, *i.e.*

$$\tilde{\Omega}_{nj} = \psi(\tilde{F}_j^{t-1})^T \cdot \psi(F_n^t). \tag{4.6}$$

The updated template $\tilde{F}_n^t$ is then passed to the later stage of the network for the final classification and offset regression.

Compared to the DROW detector, the DR-SPAAM detector has a significantly lower computational complexity, requiring neither robot odometry nor the cutout re-computation on previous scans. The auto-regressive model also allows DR-SPAAM to keep only a single template per angular index, without having to store multiple past scans, while being able to accumulate information within a larger temporal window.

4.4 Evaluation

We evaluate DR-SPAAM using the DROW dataset (Beyer *et al.*, 2016, 2018), which is recorded in an indoor rehabilitation facility using a SICK S300 scanner. The dataset contains 24,012 annotated scans, split into *train* (17,665), *validation* (3,919), and *test* (2,428) sets. The annotation includes the locations of three classes of objects: *wheelchair*, *walker*, and *person*. In this

work we specifically focus on detecting persons and ignore the annotations of the other two categories, albeit our method should be general enough to handle other classes with adjusted hyperparameters.

Following the standard in the object detection community, we use average precision (AP) at different association distances as our main evaluation metric. AP_d means that a detection is considered as positive if there exists an unmatched ground truth that is within d m radius of the predicted location. Notice that Beyer *et al.* (2016, 2018) reported the area under the precision-recall curve (AUC), which is equivalent to average precision by definition. Additionally, we report the peak-F1 score (using 0.5 m association distance), which is the maximum harmonic mean of the different precision and recall values, as well as the equal error rate (EER), the value at which precision and recall are equal.

All our models are trained on the *train* set with a batch size of 8 scans for 40 epochs. For our DR-SPAAM detector, we load 10 frames back into the past during training, since scans that are further back in time are less relevant due to the exponential decay. We use an Adam optimizer, with initial learning rate 10^{-3} with an exponential decay after each iteration to 10^{-6} during the complete training. For the classification we use the binary cross entropy loss and for the regression we use the $L1$-norm of the regression error. We use the same voting scheme introduced by Beyer *et al.* (2018) to convert the network output to final detections. The hyperparameters in the voting step are optimized with Hyperopt (Bergstra *et al.*, 2013) by maximizing $\text{AP}_{0.5}$ on the validation set for each model individually. During evaluation, similar as during training, we provide a temporal context of 10 past frames for each test scan. However, our approach readily generalizes to run on complete sequences.

4.4.1 Quantitative Results

We evaluate our proposed method using the test set and report the average precision, peak-F1 score, and equal error rate of our method at an association threshold of 0.5 m in Table 4.1. As an additional baseline, we report the performance of two re-trained DROW models, using a single scan and five scans, respectively. Compared to the original implementation, our re-trained models use a smaller cutout window and more sampling points within each cutout, which we selected based on a better performance on the validation set (see Section 4.4.3). In order to keep the comparison meaningful, the same cutout parameters are used for both the re-trained DROW baseline and DR-SPAAM, and no odometry information is used. The original person detection performance of the DROW detector from Beyer *et al.* (2018), as well as the detection performance from Leigh *et al.* (2015) and Arras *et al.* (2007) re-trained on the DROW dataset, are included in Table 4.1.

The results show that DR-SPAAM establishes a new state-of-the-art. It has the highest $\text{AP}_{0.5}$ of 70.3%, which is 2.4% above the baseline model and 2.2% above the original DROW detector, even without using odometry information. By comparing our re-trained DROW models with the original ones, we can observe the effectiveness of our proposed hyperparameters adjustment, especially in the single scan case. The precision-recall curves of all models are

Table 4.1: Detection accuracy on the test set with $0.5m$ association threshold. Note that DR-SPAAM and our re-trained DROW baseline do not use odometry information.

Method	$\mathbf{AP}_{0.5}$	peak-F1	EER
ROS leg detector (Pantofaru, 2010)	23.2	41.7	41.0
Arras, re-trained (Arras *et al.*, 2007)	47.6	50.3	50.1
Leigh, re-trained (Leigh *et al.*, 2015)	57.2	64.3	62.3
DROW, $T = 1$ (Beyer *et al.*, 2018)	59.4	61.5	61.4
DROW, $T = 5$ (Beyer *et al.*, 2018)	67.0	65.9	64.9
DROW, $T = 5$, with odometry (Beyer *et al.*, 2018)	68.1	68.1	**67.2**
DROW, $T = 1$	66.6	66.1	65.2
DROW, $T = 5$	67.9	65.1	63.8
DR-AM (ours, no spatial attention)	66.3	65.2	64.0
DR-SPA (ours, no auto-regression)	68.0	67.0	66.1
DR-SPAAM (ours, full)	**70.3**	**68.5**	**67.2**

shown in Figure 4.2. The curves show that DR-SPAAM outperforms all other setups, except for a small region in the high precision regime, where the DROW (T=5) baseline scores higher.

Table 4.1 additionally shows the results of an ablation study that highlights the contribution of the different components in our temporal aggregation module. DR-AM corresponds to a network where the auto-regressive model is updated with the new features directly, without using the weighted sum from the spatial attention mechanism. It has a slightly worse performance compared to a single-scan DROW baseline, showing that it is not beneficial to naively combine misaligned features. DR-SPA, on the other hand, only combines the features from the current and the previous scan using spatial attention, without using the accumulated feature template from the auto-regressive model. This two-scan approach already outperforms the five-scan DROW baseline, showing the benefit of using a learning-based approach to incorporate features from the previous measurements. The full model, DR-SPAAM, outperforms the two-scan DR-SPA, showing the benefit of aggregating information over a larger temporal window.

4.4.2 Inference Time

Besides the improved detection performance, DR-SPAAM significantly reduces the computational cost by eliminating the need to perform expensive re-computation of past cutouts. We implement all networks in Python and PyTorch without using any inference time acceleration framework such as TensorRT, and profile the run time of different components of the complete pipeline. The timing results for two mobile platforms are reported in Table 4.2. Here we use a laptop equipped with a mobile NVIDIA GeForce RTX 2080 Max-Q GPU and an Intel-i7 9750H CPU, as well as a Jetson AGX Xavier. Table 4.2 shows that DR-SPAAM has

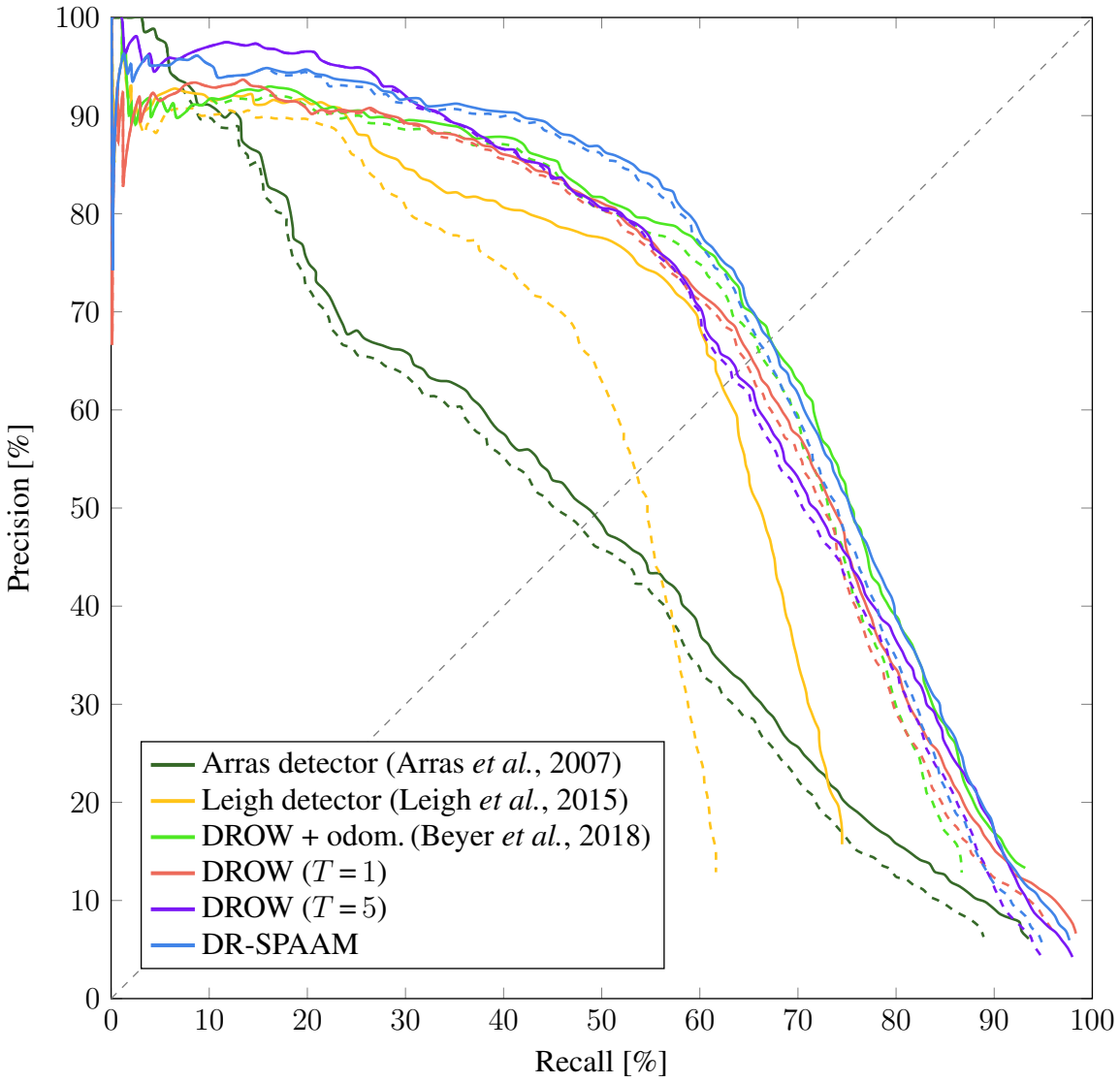


Figure 4.2: Precision-recall curves for several baselines and our DR-SPAAM detector, evaluated with association distance 0.5 m (solid) and 0.3 m (dashed).

a computation time similar to that of a single-scan DROW method. This is expected, since at each time step only the current scan needs to be processed and the fusion of DR-SPAAM adds a small computation overhead. However, even though the methods are similar in speed, the single-scan DROW method has a by far lower detection accuracy. DROW with temporal integration, on the other hand, has a runtime that scales linearly with the number of scans due to the required cutout re-computation on all past scans, and already becomes significantly slower when using five scans. Considering the Jetson AGX platform, a frame rate of 2.6 FPS is too slow for a real-time application, whereas our 9.7 FPS is still well within a usable range. On the laptop, we can in fact run all models faster than real-time, given that the DROW dataset was

Table 4.2: Computation time in milliseconds and frame rate of different setups on two different mobile platforms.

		Laptop (RTX 2080)				Jetson AGX Xavier			
Method	$\mathbf{AP}_{0.5}$	Cutout	Network	Vote	FPS	Cutout	Network	Vote	FPS
DROW, $T=1$	66.6	7.0	1.4	6.1	68.6	63.3	4.8	29.3	10.4
DROW, $T=5$	67.9	34.3	1.5	19.2	18.2	306.3	5.1	78.1	2.6
DR-SPAAM	**70.3**	7.0	2.0	7.7	59.8	62.0	6.9	33.6	9.7
DR-SPAAM†	**71.8**	1.1	1.9	8.5	87.2	4.2	7.7	32.4	22.6

recorded at a frame rate of roughly 13 FPS. However, DR-SPAAM can run at significantly higher frame rates if needed, providing detections with a lower latency and consuming less power, which is a relevant aspect for mobile platforms.

The cutout generation and the voting are the expensive steps in the whole pipeline. To further increase the frame rate, we resort to a faster implementation of the cutout generation and the resulting model, DR-SPAAM†, runs at 87.2 FPS on a laptop with a dedicated GPU, or at 22.6 FPS on a Jetson AGX, well-beyond the requirement for many real-time applications. The new implementation increases the network performance thanks to the improved numerical precision.

4.4.3 Hyperparameter Selection

Cutout. The cutout operation is parameterized by the width $\overline{W}$ and depth D of the window, as well as the number of points N used for re-sampling. Beyer *et al.* (2018) use a large window of 1.66 m $\times$ 2.0 m with 48 points in order to cope with the bigger walking-aid classes. Since we are only concerned with detecting persons, we propose to use a smaller window that tightly fits the footprint of a person, thus reducing distracting information from the surroundings. We conduct an experiment on the size of the cutout window on the validation set using a DROW network with a single scan. The results are shown in Table 4.3. Based on these results, we set our cutout window to 1.0 m $\times$ 1.0 m with 56 points and use these cutout parameters for all models we train.

One can observe that the scores on the validation set are significantly lower than those on the test set. The original DROW dataset is created for detecting three classes: *wheelchair*, *walker*, and *person*. In this work, we have only kept the annotations for the *person* class, and we observed that the validation set happens to contain more person annotations at farther distances. Even though we use a distance robust preprocessing, at farther distances information becomes so sparse that the detection will always become less reliable, thus rendering the validation set more challenging.

Table 4.3: Validation set scores of DROW ($T = 1$) detectors with different cutout parameters.

$\overline{W}$	D	N	$\mathbf{AP}_{0.3}$	$\mathbf{AP}_{0.5}$	**peak-F1**	**EER**
1.66	2.0	48	41.9	43.0	48.1	47.6
1.66	1.0	48	42.6	43.4	49.2	48.6
1.0	2.0	48	43.6	44.8	50.7	50.4
1.0	1.0	48	44.0	45.0	50.3	50.2
1.0	1.0	32	42.0	43.0	49.1	48.8
1.0	1.0	40	43.1	44.1	50.0	49.6
1.0	1.0	48	44.0	45.0	50.3	50.2
1.0	1.0	56	**45.1**	**46.3**	**50.9**	**50.8**
1.0	1.0	64	43.8	45.1	50.7	50.4

Table 4.4: Validation set scores of DR-SPAAM with different window sizes and update rates.

W	α	$\mathbf{AP}_{0.3}$	$\mathbf{AP}_{0.5}$	**peak-F1**	**EER**
7	0.3	45.0	46.2	52.5	52.5
7	0.5	49.5	50.9	54.6	53.6
7	0.8	46.8	48.3	54.1	54.0
11	0.3	51.5	53.0	56.8	56.4
11	0.5	**52.7**	**53.9**	**57.3**	**57.3**
11	0.8	47.4	48.7	53.6	53.2
15	0.3	51.5	52.8	56.1	55.3
15	0.5	50.7	52.1	55.0	54.7
15	0.8	47.0	48.2	53.1	53.0

SPAAM module. The Spatial-Attention and Auto-Regressive Model are parameterized by the update rate α and the size of the search window W. Although the meaning of these two parameters is intuitively clear, it is not a trivial task to select the proper combination. We train multiple DR-SPAAM networks with different combinations of α and W. Based on the validation set results in Table 4.4, we choose to use $W = 11$ and $\alpha = 0.5$ for our final model. No clear pattern can be seen in these results, neither larger search windows nor specific update rates consistently work better. A more thorough search through the parameter space could potentially result in models that perform even better.

4.4.4 Sampling Rate

Different LiDAR sensors often have different sampling rates. Since a detection network is likely to be deployed on different LiDAR sensors, its robustness against varying sensor specifications should be examined. We take two networks, DROW ($T = 5$) and DR-SPAAM, and

evaluate them on the test set using temporally sub-sampled sequences, simulating different sampling rates. Table 4.5 reports the detection accuracies at different temporal strides (a stride of n means keeping only every nth scan).

The evaluation results show that DR-SPAAM is very robust against changing sampling rate. Even at a five times lower scanning frequency (roughly 2 Hz), the $AP_{0.3}$ only reduces by 2.1%. This result shows the benefit of a learned spatial attention module, which combines information based on appearance similarity without relying on a fixed temporal context window. Hence, DR-SPAAM can be deployed on LiDAR sensors with a wide range of sampling rates, or operate with a reduced sampling rate if the computation capacity is limited. On the other hand, the DROW detector performance degrades rapidly with increased temporal stride. Larger time differences between consecutive scans lead to greater motion induced misalignments, which in turn hurt the accuracy of the DROW detector.

Table 4.5: Test set results with different temporal strides.

	DROW, $T = 5$				**DR-SPAAM**			
Stride	$AP_{0.3}$	$AP_{0.5}$	p-F1	EER	$AP_{0.3}$	$AP_{0.5}$	p-F1	EER
1	**66.6**	**67.9**	**65.1**	**63.8**	68.5	70.3	68.5	67.2
2	59.3	60.5	60.1	59.3	69.3	70.8	**68.8**	**67.6**
3	54.3	55.8	56.8	56.7	**69.4**	**70.9**	68.1	66.5
4	53.6	55.1	56.0	55.7	67.7	69.1	66.4	64.9
5	51.5	53.4	54.6	54.3	66.4	67.7	65.5	64.5

4.4.5 Temporal Association

During the temporal aggregation step, DR-SPAAM computes the similarity between the aggregated template and the latest scan features (Equation 4.6). This similarity can be further exploited for associating points across different scans. Here we provide a preliminary example. We take 200 consecutive frames from a sequence, roughly spanning 15 s, and detect persons using DR-SPAAM for each individual frame. For each detection D_i^t, we find its corresponding points in the previous scan, simply by selecting the ones with highest similarity score. If these corresponding points were grouped into a detection D_j^{t-1}, and if the distance between the two detections are smaller than a threshold of 0.5 m, we group both detections into a tracklet. Otherwise a new tracklet is started using D_i^t. After 200 frames, we compute the confidence of each tracklet as the mean of all its detections. In Figure 4.3 we plot all tracklets that have a confidence greater than 0.35 and that are composed of at least five detections. The trajectories of persons in the scene are clearly visible. These associations can provide extra information for tracking algorithms, with their full potential yet to be explored in future research. The velocity and movement direction of the persons can also be derived from the associated detection pairs, and this information can be helpful for motion planning.

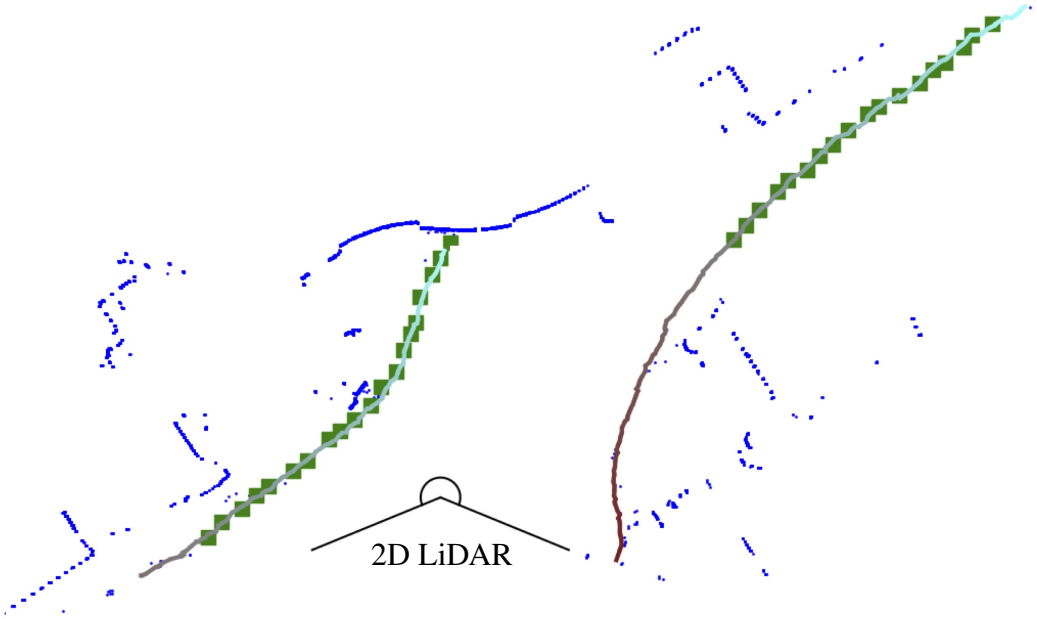


Figure 4.3: Tracklets generated by DR-SPAAM using 200 consecutive scans. The blue points are the overlaid scans, and the green squares are the ground truth annotations. Notice that for clarity, we omit the points that have been classified as persons from plotting. The colored lines are the tracklets, and the coloring encodes the time of detections. The sequence is taken from the training set, since the validation set does not have any sequence recorded using a stationary LiDAR, which is needed for plotting overlaid scans. Nevertheless, most of the scans have not been annotated (shown as the missing annotations along the tracklets) and have not been exposed to the network during training.

4.5 Discussion

In this chapter we introduced the DR-SPAAM person detector that combines the distance robust detection scheme of the DROW detector (Beyer *et al.*, 2018) with a powerful spatial attention and auto-regressive temporal integration model. The spatial attention is able to associate misaligned features from different frames using their appearance similarity, while the auto-regressive model aggregates temporal information forward through time. Compared to the previous state-of-the-art approaches, DR-SPAAM achieves higher detection accuracy, while being significantly faster and able to run in real-time even on low-powered mobile platforms. Experiments show that DR-SPAAM generalizes well to LiDAR sensors with different temporal sampling rates, thanks to the learned data association between continuous scans.

The high-frame rate and generalization capability makes DR-SPAAM an ideal candidate for deployment on mobile robots. In the EU H2020 project “CROWDBOT” (779942), pre-trained

DR-SPAAM has been deployed on four robotic platforms, ranging from smart wheelchairs to logistic robotics. All of these robotic platforms have different sensor setups, but we did not need to re-train the detector, thanks to its good generalization capability. By providing our code and ROS node, we believe that our model will be beneficial for many robotic applications.

Existing works in 2D-LiDAR-based person tracking primarily rely on person location and motion constraint, which are formulated either as filtering problems or heuristics (Schulz *et al.*, 2003). Our preliminary experiment demonstrated the possibility of leveraging the learnt temporal association of DR-SPAAM for person tracking. The similarity map from spatial attention is readily integrated into the popular tracking by detection paradigm (Ess *et al.*, 2008; Weng *et al.*, 2020). In such a paradigm, detections are first made for the incoming new frame, and these detections are assigned to new or existing tracks. The assignment is typically cast as a bipartite matching problem, where the association cost to be minimized is modeled by geometric proximity between the detections and the predicted locations for each track. With the similarity score from DR-SPAAM, a new association cost can be designed that combines both geometric constraints from motion modeling and the learned appearance information, similar to the vision-based person tracking with re-identification (Hermans *et al.*, 2017; Beyer *et al.*, 2017; Narayan *et al.*, 2018).

Even though DR-SPAAM uses 2D LiDAR sensor as its input source, many of its design choices are applicable to 3D LiDAR sensors. Range images are a popular representation of 3D LiDAR scans (Milioto *et al.*, 2019), where the indices of each pixel represents the polar angles and the value of the pixel represents the range of the points. In such a range image, the cutout preprocessing can be used to effectively separate a point from its fore- and background, and the SPAAM module can be used to associate nearby points across consecutive frames.

Collecting training data for DROW and DR-SPAAM is a laborious and resource consuming process. Ideally, these data should contain diverse examples in terms of sensor types and environments, in order to prevent overfitting to the training set. At the time when DR-SPAAM was published, the DROW dataset collected by Beyer *et al.* (2018) was the only publicly available datasets for 2D LiDAR data. To address this shortage in training data, in the next chapter, we introduce a method to automatically generate training labels for 2D LiDAR data with no manual annotations, which produces detectors with similar performance to the ones trained using expensive manual annotation.

5 Self-Supervised Person Detection in 2D Range Data using a Calibrated Camera

5.1 Introduction

Person detection is a critical task for many robotic applications, and the previous chapter shows advantage of using 2D LiDAR sensors for person detection, in particular their accurate range measurement and large field of view. While early approaches for detecting persons in 2D range data focus on heuristics with hand-crafted features (Leigh *et al.*, 2015; Arras *et al.*, 2007), recent studies, including the DR-SPAAM detector discussed in the previous chapter, use convolutional neural networks to obtain state-of-the-art results (Beyer *et al.*, 2018; Jia *et al.*, 2020).

Deep learning has become an integral part of modern detection algorithms, whether from 2D range data (Beyer *et al.*, 2018; Jia *et al.*, 2020) or from images (Tan *et al.*, 2020; He *et al.*, 2017; Redmon *et al.*, 2016; Liu *et al.*, 2016; Ren *et al.*, 2015). The success of these detectors hinges upon the availability of large and high-quality datasets. Over the past years, significant effort has gone into labeling images with bounding boxes or segmentation masks, whereas relatively little attention has been given to annotating 2D range data (see Figure 5.1). The limited availability of datasets for 2D LiDAR sensors restricts the variety of surrounding environments and sensor models exposed to the learning algorithm. Networks trained solely on these data may not generalize well at deployment, where both the environment and the sensor specification are likely to differ from those encountered during training.

To overcome the limitation imposed by insufficient training data, we propose a method to automatically generate labels for training LiDAR-based person detectors using the output of an image-based detector on a calibrated camera. Given a person bounding box in an image,

This chapter is based on the 2021 ICRA conference paper titled *Self-Supervised Person Detection in 2D Range Data using a Calibrated Camera* by Jia, Steinweg, Hermans, and Leibe. I was responsible for the method design, experiment evaluation, and writing. Steinweg conducted some early-stage experiments, the results of which were not included in the paper. Hermans and Leibe contributed through regular discussions and revisions.

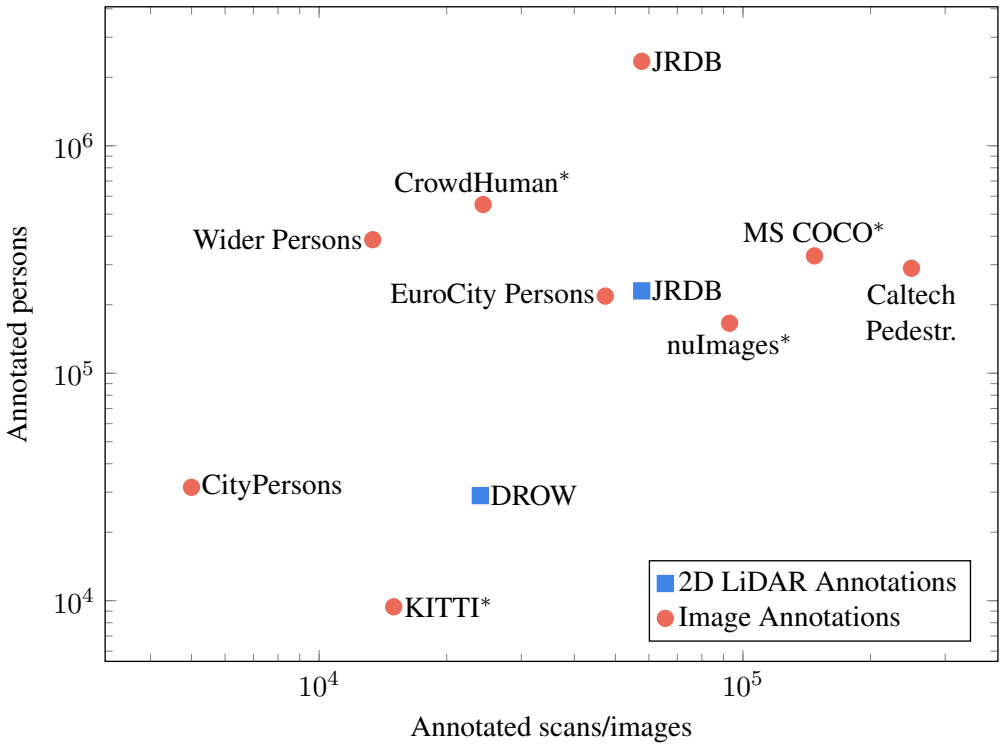


Figure 5.1: Only few 2D LiDAR datasets with person annotations exist, while a vast amount of image-based annotations are available. We utilize the image-based annotations to train 2D-LiDAR-based person detectors by generating pseudo-labels from image-based detections.
*: Numbers are estimated based on person-per-image statistics of the training set.

our method takes the 2D LiDAR points that fall within the box frustum and uses a clustering algorithm to locate the person in the LiDAR coordinate. The estimated locations of persons in the scene, as well as a set of negative points, are used as pseudo-labels for training a detector. This allows a robot to train or fine-tine a laser-based detector in a self-supervised fashion. We empirically demonstrate the validity of the pseudo-labels and show that they can be used for both training and fine-tuning a 2D-LiDAR-based person detector. Additionally, we experiment with robust training techniques to further improve the detector performance. Our method is an effective way to boost the performance of a person detector during deployment without any additional labeling effort, and has great potential for many robotic applications.

In summary, the main contributions of this chapter are:

- We propose a method to automatically generate pseudo-labels for training 2D-LiDAR-based person detectors, leveraging the output of an image-based detector and known extrinsic camera calibration.
- We demonstrate that the generated pseudo-labels can be used to train or fine-tune a person detector and experiment with robust training techniques to further improve its performance.

5.2 Related Work

5.2.1 2D-LiDAR-based Person Detection

Person detection from 2D range data has a long-standing history in the robotics community. Early approaches (Fod *et al.*, 2002; Scheutz *et al.*, 2004; Schulz *et al.*, 2003) focus on tracking moving blobs in sequential LiDAR scans. These blobs are detected using manually engineered heuristics in a non-learning fashion. Later developments (Leigh *et al.*, 2015; Arras *et al.*, 2007; Pantofaru, 2010) improved the detection stage by using supervised learning techniques. In these approaches, a clustering algorithm is first applied over scan points, using clues like proximity (Leigh *et al.*, 2015) or jumping distance (Arras *et al.*, 2007). A set of hand-crafted features is then extracted for each cluster, and these features are used to train a simple classifier, such as AdaBoost (Schapire, 2013). Additional heuristics from tracking are employed to post-process the detected clusters (Leigh *et al.*, 2015; Pantofaru, 2010).

Most recent developments (Beyer *et al.*, 2016, 2018; Jia *et al.*, 2020) use deep learning techniques to detect persons directly from data, without manually engineered heuristics or features. The DROW detector (Beyer *et al.*, 2016) is the first deep learning-based walking aid detector working on 2D range data and was later extended to additionally detect persons (Beyer *et al.*, 2018). The current state-of-the-art method is the DR-SPAAM detector (Jia *et al.*, 2020), which leverages a temporal aggregation paradigm to incorporate multiple scans into the detection process, alleviating the problems associated with the low information content in a single LiDAR scan while retaining real-time computation on a mobile robotic platform.

5.2.2 Automatic Label Generation

Supervised learning approaches demand intense effort to manually collect and annotate data for the purpose of training and testing. Automatic label generation has been attempted to reduce the required labeling effort. For example, Leigh *et al.* (2015) generate positive training examples by having one person moving in a pre-mapped space, and negative examples by moving the LiDAR sensor in environments devoid of people. This method limits the training examples to a few simple scenarios and cannot be adapted for dynamic data collection at deployment time. Aguirre and García-Silvente (2019) use Mask R-CNN (He *et al.*, 2017) with a calibrated RGB-D camera to automatically label 2D LiDAR scans for person detection. However, no evaluation of the labels' accuracy is performed. Furthermore, only very simple LiDAR-based person detectors are used and both the training and the evaluation are conducted using (potentially incorrect) generated labels, with no evaluation using real annotated data. The method also relies on depth measurements from an RGB-D camera, imposing an additional sensor requirement. Automatic label generation has also been attempted for 3D LiDAR sensors. Piewak *et al.* (2018) and Wang *et al.* (2019a) train point-cloud segmentation networks, using segmentation results on matching pixels as supervision.

In this chapter, we discuss an approach to automatically generate training data for a 2D-LiDAR-based person detector using a calibrated RGB camera and image-based person detec-

tions. Compared to Leigh *et al.* (2015), our pseudo-labels are dynamically generated and do not rely on specific conditions of the environment. Unlike Aguirre and García-Silvente (2019), our method operates on normal RGB cameras and does not require expensive segmentation on images or pixel-precise calibration between sensors. We conduct extensive experiments to analyze the quality of pseudo-labels and empirically prove their value for training state-of-the-art person detectors.

5.2.3 Learning with Noisy Labels

We resort to robust training techniques to mitigate problems caused by noisy pseudo-labels. Training neural networks with imperfect data is becoming an increasingly relevant topic. Proposed methods include robust loss functions (Ghosh *et al.*, 2017; Zhang and Sabuncu, 2018; Wang *et al.*, 2019d; Menon *et al.*, 2020), noise modeling (Xiao *et al.*, 2015; Goldberger and Ben-Reuven, 2017; Han *et al.*, 2018b), sample selection (Kumar *et al.*, 2010; Jiang *et al.*, 2018; Han *et al.*, 2018a), sample re-weighting (Ren *et al.*, 2018; Shu *et al.*, 2019), and regularization (Srivastava *et al.*, 2014; Jindal *et al.*, 2016; Menon *et al.*, 2020). In this chapter, we experiment with the *partially Huberized cross-entropy loss* (Menon *et al.*, 2020) and the *mixup* regularization (Zhang *et al.*, 2018). These two methods are picked for their applicability—they do not rely on specific noise properties, nor do they impose additional requirements such as a small set of perfect training data. For a more thorough study on robust training techniques, we refer readers to survey articles by Zhang *et al.* (2016), Algan and Ulusoy (2019), and Song *et al.* (2020).

5.3 Generating Pseudo-Labels

We use a calibrated camera to generate pseudo-labels for training a 2D-LiDAR-based person detector. These pseudo-labels include the location of persons in the LiDAR coordinate frame $\{(p_{x,i}, p_{y,i})\}_i$, and a set of scan points that belong to the background of the scene.

We first use an object detector, *e.g.* Faster R-CNN (Ren *et al.*, 2015), to obtain person bounding boxes. From all bounding boxes, a subset is selected using the following constraints: First, the *classification score* is greater than a threshold T_c. Second, the *aspect ratio*, the ratio between width and height, is smaller than T_{AR}. Third, the *overlap ratio* with any other bounding box is smaller than T_o. The overlap ratio is defined as the intersection area divided by the area of the box. These constraints aim to select boxes from which the location of persons can be confidently extracted, rather than locating *all* persons in the scene. In other words, precision is valued more than recall in this step.

Given a selected bounding box, we estimate the center location $(p_{x,i}, p_{y,i})$ of the person in the LiDAR coordinate frame. Utilizing the known camera calibration, we project the LiDAR points onto the image and extract points that fall within the bottom half of the bounding box, since the LiDAR sensor is mounted at the height of the ankle. These points either correspond to a person or to the background (see Fig. 5.2). To localize the person, we first run a k-means

clustering in the range space with $k = 2$, which groups points into a close and a far cluster. We take the average 2D location of points in the close cluster as the initial estimation, and iteratively refine this estimation using a mean shift procedure with a circular kernel of 0.5 m radius. The mean shift result is used as the estimated person location. This proposed method assumes that the person belongs to the foreground of the scene and is the dominant object in the cropped LiDAR scan, which is typically satisfied by the content of detection bounding boxes.

LiDAR points that do not project to any kept or discarded bounding boxes are taken as the negative training samples. For increased robustness, we enlarge the width of each bounding box by a factor of 0.1 when generating the negative samples.

5.4 Person Detection with Pseudo-Labels

5.4.1 DROW and DR-SPAAM Detector

We experiment with two state-of-the-art person detectors, DROW detector (Beyer *et al.*, 2018) and its successor DR-SPAAM detector (Jia *et al.*, 2020), which are discussed extensively in the previous chapter. The DROW detector takes as input a 2D LiDAR scan, expressed as a one-dimensional vector of range measurements. For each point, it outputs a classification label and, for the positive points, a location offset to the person center. This is accomplished by pooling a small window of neighboring points, which are processed by a 1D convolutional neural network. DR-SPAAM improves upon this approach by introducing a spatial attention and auto-regressive model that integrates temporal information to improve detection performance. Thus, it requires the input to be a sequence of scans.

We use the generated pseudo-labels to train DROW and DR-SPAAM. For supervising the classification branch, we use points less than 0.4 m away from an estimated person center (p_x, p_y) as the positive samples, and the marked out background points as the negative samples. For supervising the regression branch, we use points less than 0.8 m away from an estimated person center. To increase robustness, we discard pseudo-labels with less than five surrounding positive points. Points that are neither close to a person nor marked as a background are ignored during training.

5.4.2 Robust Training

In the default setup, the classification branch of DROW and DR-SPAAM is supervised with the cross-entropy loss, which is prone to label noise in the training samples (Ghosh *et al.*, 2017; Wang *et al.*, 2019d). To limit the influence of wrongly generated pseudo-labels, we resort to robust training techniques. We experiment with the *partially Huberized cross-entropy loss* (Menon *et al.*, 2020), a more robust loss function, and the *mixup* regularization (Zhang *et al.*, 2018).

Partially Huberized cross-entropy loss. The softmax cross-entropy loss is composed of a base loss (cross-entropy) with a sigmoid link. Menon *et al.* (2020) propose a composite loss-based gradient clipping, linearizing the base loss beyond a threshold while leaving the sigmoid link untouched. The overall loss function takes the form:

$$l = \begin{cases} -\tau \cdot p + log(\tau) + 1, & \text{if } p \leq \frac{1}{\tau} \\ -log(p), & \text{else,} \end{cases} \tag{5.1}$$

which asymptotically saturates for $p \leq \frac{1}{\tau}$, and was proven to be robust against label noise. The parameter τ should be set in proportion with the amount of noise in the training samples. In our experiments, we use $\tau = 5$.

Mixup regularization. Given two training samples (x_i, y_i) and (x_j, y_j), Zhang *et al.* (2018) propose to construct augmented training samples:

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j$$
$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j,$$

with $\lambda \sim \text{Beta}(\alpha, \alpha)$ and the parameter $\alpha \in (0, \infty)$ controlling the augmentation strength. Unlike the conventional data augmentation which operates on a single sample (e.g. randomly flipping an image), *mixup* generates virtual samples across training data. It encourages linear behavior in-between training examples, and was shown to improve generalization of a network and increase its robustness against label noise.

To adapt *mixup* for training a detection network, which includes both classification and regression, we use the following multi-task loss

$$l_{total} = l_{reg} + (1 - w) \cdot l_{cls} + w \cdot l_{mixup}, \tag{5.2}$$

where l_{reg} and l_{cls} are the regression and classification loss without *mixup* regularization, l_{mixup} is the classification loss with *mixup*, and w is a weighting factor. To avoid additional memory overhead, we split the cost into

$$l_1 = l_{reg} + (1 - w) \cdot l_{cls}$$
$$l_2 = w \cdot l_{mixup}$$

and perform gradient descent over l_1 and l_2 sequentially at each training iteration. In our experiment, we use $w = 0.7$ and $\alpha = 0.2$.

5.5 Evaluation

We first conduct experiments to validate the quality of the pseudo-labels and then demonstrate their effectiveness for training person detectors. We present two case studies: training detectors using pseudo-labels with pre-collected data as well as fine-tuning detectors using dynamically generated pseudo-labels.

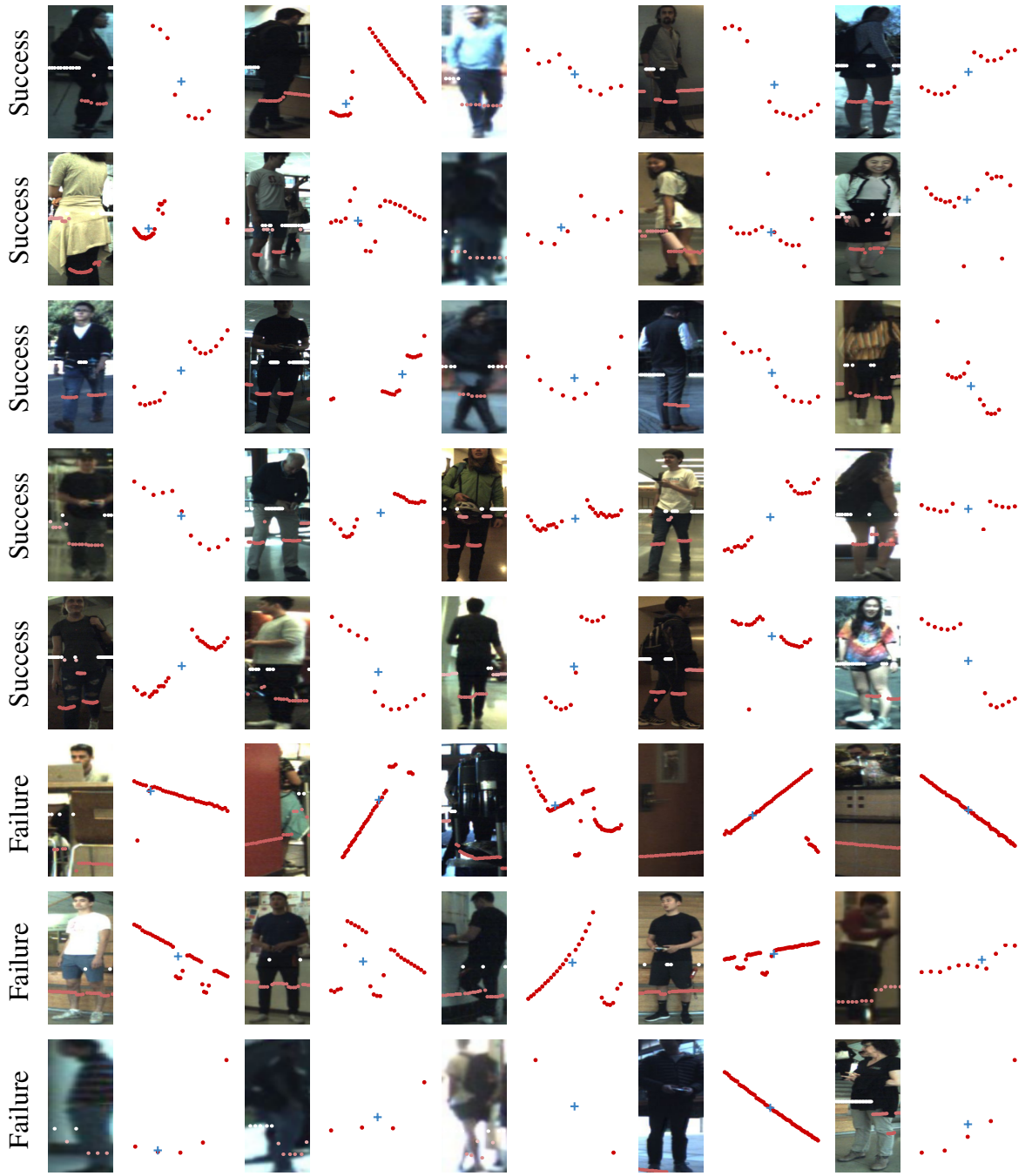


Figure 5.2: Person detections and the top-down view of the matching pseudo-labels (+) with surrounding LiDAR points (within a 0.5 m radius). The LiDAR points are overlaid in the detection images, where the color encodes measured distance with red being closest and white being farthest. Common failure cases are shown in the bottom three rows: heavy occlusion (row 6); background distraction (row 7); sparse LiDAR points at far distance (row 8, column 1-3); and faulty calibration or synchronization between sensors (row 8, column 4-5).

5.5.1 The JackRabbot Dataset and Benchmark (JRDB)

Our experiments are conducted on the JRDB dataset (Martin-Martin *et al.*, 2021). The dataset is collected using a mobile robot, the *JackRabbot*, in both indoor and outdoor environments. It includes point clouds from 3D LiDAR sensors, annotated with 3D bounding boxes for persons, and RGB camera images, annotated with 2D bounding boxes. Although not directly annotated, the JackRabbot dataset contains scans from two SICK 2D LiDAR sensors. These LiDAR sensors are mounted at the height of the lower legs, facing the front and the back of the robot, respectively. The dataset features a full 360° scan with 1091 points, generated by combining scans from the two LiDAR sensors, which is used as the LiDAR input in our experiments.

For the evaluation of our pseudo-labels and trained detectors, we convert the annotated 3D bounding boxes into 2D LiDAR annotations by using their center as the ground truth location of persons in the scene. Since many 3D bounding boxes are occluded in the view of the 2D LiDAR sensors, we only keep annotations that have at least five points within a 0.5 m radius.

The dataset includes person detections from a Faster R-CNN detector (Ren *et al.*, 2015), which we use as the input to our pseudo-label generation approach. However, using the included RGB images other detectors can be applied, allowing us to potentially further improve the label quality. For generating pseudo-labels, we use $T_c = 0.75$, $T_{AR} = 0.45$, and $T_o = 0.4$.

To evaluate our approach, we employ a custom train-test split as the JRDB test set annotations are not publicly available. We split the 27 sequences of the original train set into 17 sequences for training and 10 sequences for testing. To assess the person detection difficulty for each sequence, we use a pre-trained DROW detector. Our train-test split is balanced in terms of person detection difficulty and scene scenario (indoor *vs.* outdoor). Table 5.1 provides a list of the sequences included in each split.

5.5.2 Pseudo-Label Statistics

To evaluate the quality of our pseudo-labels, we calculate the true positive and the true negative rate (TPR, TNR) of the classification target generated using pseudo-labels by comparing them against the target generated using ground truth annotations. Table 5.2 shows that more than 90% of the training samples are labeled correctly, indicating the validity of the pseudo-labels. Qualitative results of pseudo-labels are shown in Figure 5.2. In most cases, locations of persons are successfully estimated in both indoor and outdoor environments, with people at different distances or having different poses. Common failure cases are identified. These include occluding objects, persons merging with the background, sparse LiDAR measurements at high distances, or faulty calibration between sensors. These failure cases result in noisy labels, which we deal with by using a robust training loss.

Table 5.1: Customized JRDB train-test split with performance of a pre-trained DROW detector.

Train Sequence (17 total)	$AP_{0.5}$	Scenario
bytes-cafe-2019-02-07_0	38.5	Indoor
clark-center-2019-02-28_1	73.4	Outdoor
clark-center-intersection-2019-02-28_0	82.3	Outdoor
cubberly-auditorium-2019-04-22_0	78.5	Indoor
forbes-cafe-2019-01-22_0	62.2	Indoor
gates-basement-elevators-2019-01-17_1	67.7	Indoor
hewlett-packard-intersection-2019-01-24_0	30.7	Outdoor
huang-basement-2019-01-25_0	52.3	Indoor
huang-lane-2019-02-12_0	77.2	Outdoor
jordan-hall-2019-04-22_0	79.7	Indoor
memorial-court-2019-03-16_0	82.3	Outdoor
nvidia-aud-2019-04-18_0	42.8	Indoor
packard-poster-session-2019-03-20_2	86.1	Indoor
stlc-111-2019-04-19_0	63.8	Indoor
svl-meeting-gates-2-2019-04-08_0	63.0	Indoor
tressider-2019-03-16_0	71.0	Outdoor
tressider-2019-04-26_2	79.3	Indoor
Test Sequence (10 total)		
clark-center-2019-02-28_0	67.8	Outdoor
gates-159-group-meeting-2019-04-03_0	67.1	Indoor
gates-ai-lab-2019-02-08_0	43.6	Indoor
gates-to-clark-2019-02-28_1	82.7	Outdoor
huang-2-2019-01-25_0	61.1	Indoor
meyer-green-2019-03-16_0	68.8	Outdoor
packard-poster-session-2019-03-20_0	80.0	Indoor
packard-poster-session-2019-03-20_1	82.9	Indoor
svl-meeting-gates-2-2019-04-08_1	69.0	Indoor
tressider-2019-03-16_1	65.5	Outdoor

Table 5.2: Accuracy of pseudo-labels on the training split.

Bounding Boxes	TPR	TNR
Faster R-CNN Detections	91.6	99.4
2D Annotations	90.6	99.1

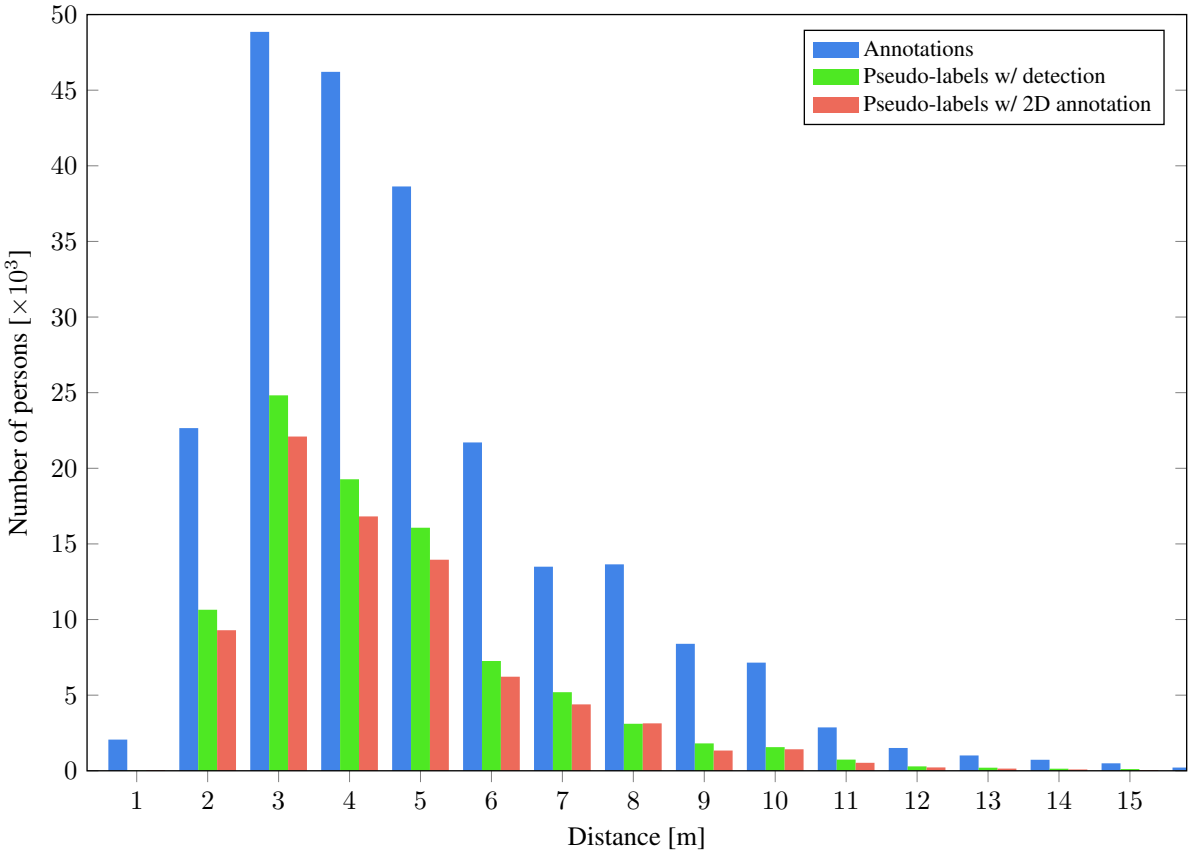


Figure 5.3: Distance distribution of pseudo-labels and annotations. Pseudo-labels have a similar distance distribution to that of the annotations and do not introduce a distance bias.

To ablate the effect of uncertainty in the image-based person detector, we generate pseudo-labels using the annotated 2D bounding boxes. The TPR and TNR of the pseudo-labels generated using detections are higher than those generated using annotations, showing that our proposed method is robust against uncertainty in bounding boxes. Due to the sparsity of LiDAR points, it is difficult to generate pseudo-labels for far away persons contained in 2D annotations. When using a detector, these persons are often missed or detected with low confidence, thus not leading to a (potentially false) pseudo-label.

The distance distribution of our pseudo-labels, as shown in Figure 5.3 shows, is similar to that of the annotations. Pseudo-labels do not introduce any distance bias which may wrongly emphasize samples within a certain range. Due to the high mounting of cameras in the JRDB dataset, the legs of persons close to the robot are outside the camera's field of view. Thus, no pseudo-labels can be generated within 1 m distance.

5.5.3 Training with Pseudo-Labels

We examine the performance of detectors trained using either pseudo-labels or ground truth annotations. We train the DROW detector for 40 epochs, using a batch size of 8 scans and the DR-SPAAM detector for 20 epochs, using a batch size of 4 scans, We use the Adam optimizer (Kingma and Ba, 2015), and a learning rate of 10^{-3}. Starting from 10 epochs for DROW and 5 epochs for DR-SPAAM, we exponentially decay the learning rate until it reaches 10^{-6} at the end of training. Implementation and hyper-parameters are taken from Jia *et al.* (2020) for both detectors, with the only exception being that we enlarge the spatial window size for DR-SPAAM to 17 points to account for denser LiDAR sensor used in collecting the JRDB dataset. This enlarged window matches the effective opening angles, since the scans in JRDB have a higher angular resolution. Due to GPU memory constraints, we randomly crop the scans to 1000 points for DR-SPAAM during training. During inference the full scan is used.

In Table 5.3 we report the average precision ($AP_{0.3}$ and $AP_{0.5}$) on the test split. A detection is considered positive if there is a ground truth within 0.3 m or 0.5 m, and a ground truth can only be matched with a single detection. As baselines, we evaluate the released DROW and DR-SPAAM model from Jia *et al.* (2020), which is trained on the DROW dataset. These pre-trained networks have significantly lower AP compared to the ones trained on JRDB, confirming our initial speculation that networks trained on a single dataset may not generalize well to new environments or different LiDAR models. The pre-trained DR-SPAAM, despite having a higher score on the DROW dataset, performed worse than DROW, showing the effect of overfitting. Networks trained using pseudo-labels, benefiting from a smaller domain gap to the test data, outperform the pre-trained networks, proving the validity of our approach. Starting from a pre-trained model improves the detector performances for pseudo-labels, but gives no clear improvement for 3D annotations.

The performance gap between pseudo-labels and annotations could be caused by two factors: label noise and less training samples. To study the effect of label noise, we additionally train networks with pseudo-labels, while removing falsely labeled points and correcting the regression target using ground truth annotations. As shown in Table 5.4, both false positives and false negatives reduce the detector performance, with the later having a strong influence on DROW. Correcting the regression target improves $AP_{0.3}$ for both networks, especially for the more powerful DR-SPAAM. Cleaning pseudo-labels increases detector performance significantly, showing that label noise is the more dominant cause to the performance gap between pseudo-labels and annotations. Detectors trained using clean pseudo-labels trail the ones trained using annotations by around 2% AP, due to reduced amounts of training samples. Fine-tuning with a pre-trained model further narrows this performance gap.

Table 5.3: Performance of DROW and DR-SPAAM trained using different supervision.

	DROW		DR-SPAAM	
Supervision	$AP_{0.3}$	$AP_{0.5}$	$AP_{0.3}$	$AP_{0.5}$
Pre-trained from Jia *et al.* (2020)	65.5	70.8	62.5	68.3
Pseudo-labels	68.5	77.3	66.9	75.5
+ fine-tuning from Jia *et al.* (2020)	69.0	77.8	69.2	76.7
3D Annotation	76.2	**82.9**	78.5	**84.9**
+ fine-tuning from Jia *et al.* (2020)	**76.4**	82.5	**78.6**	83.8

Table 5.4: Performance of DROW and DR-SPAAM trained with different variants of pseudo-labels.

	DROW		DR-SPAAM	
Supervision	$AP_{0.3}$	$AP_{0.5}$	$AP_{0.3}$	$AP_{0.5}$
Pseudo-labels	68.5	77.3	66.9	75.5
... (remove FP)	71.6	79.0	71.1	77.8
... (remove FN)	68.4	77.8	70.3	78.6
... (remove FP & FN)	73.1	80.8	72.2	78.8
... (remove FP & FN, correct reg.)	74.6	80.5	**76.5**	**81.5**
+ fine-tuning from Jia *et al.* (2020)	**74.7**	**81.2**	76.2	**81.5**

Table 5.5: Performance of DROW and DR-SPAAM trained with pseudo-labels and different robust training methods.

	DROW		DR-SPAAM	
Training scheme	$AP_{0.3}$	$AP_{0.5}$	$AP_{0.3}$	$AP_{0.5}$
Cross-entropy loss	68.5	77.3	66.9	75.5
+ mixup regularization	69.5	78.0	65.8	74.0
Partially Huberized cross-entropy loss	**71.4**	**79.0**	69.4	76.4
+ mixup regularization	71.1	78.5	**70.0**	**78.3**

To mitigate the problem caused by labeling noise, we experiment with two robust training methods: the *partially Huberized cross-entropy loss*, and the *mixup* regularization. Table 5.5 shows that both methods improve the network performance, with the exception of applying *mixup* alone on DR-SPAAM (potentially due to suboptimal hyper-parameters). Combined with robust training methods, detectors trained using pseudo-labels outperform the pre-trained detectors by a large margin, and reach a performance close to training using annotations, without using any labeled data. Pseudo-labels provide an effective way to adjust person detectors to new environments or LiDAR models.

5.5.4 Online Fine-Tuning with Pseudo-Labels

During deployment, it is desirable for a detector to be able to dynamically fine-tune itself in an online fashion. To study the network performance undergoing such fine-tuning, we take a DROW detector, pre-trained on DROW dataset, and fine-tune it on the JRDB train split for one epoch, with the partially Huberized cross-entropy loss. We use a learning rate of 5×10^{-5} and a batch size of 8 scans. In the first set of experiments, we shuffle data only within each sequence (the whole train split is composed of 17 sequences) and pass the in-sequence shuffled data to the network. This mimics the situation where a mobile robot enters into a new environment, curates a small amount of data, and fine-tunes itself. In the second set of experiments, we shuffle data within the whole training split, giving more diversity in each batch.

The detector performance at different stages of fine-tuning is shown in Figure 5.4. When the data is shuffled within the whole training split, the network performance increases significantly, from the pre-trained 70.8 percent $AP_{0.5}$ to more than 74 percent, using less than one hundred updates. This fast performance increase implies that, even in applications with insufficient computation for a full training, it is still possible to adapt the detector and improve its performance, by running a small number of updates using pseudo-labels. However, having curated training samples with enough diversity is a key prerequisite, as fluctuating performance is observed when the data was shuffled only within each sequence. Although for most of the time the detector benefits from fine-tuning (having better performance than that of a pre-trained detector), there are adversarial samples that cause dramatic performance reductions. The same fluctuating behavior exists for fine-tuning using annotations, showing that it is an innate problem of network training, rather than caused by pseudo-labels. Curriculum learning methods (Bengio *et al.*, 2009; Jiang *et al.*, 2018) may help mitigating this problem, and thus relaxing the requirement of data collection. They are interesting directions to be explored in future research.

5.6 Discussion

In this chapter, we proposed a method to automatically generate pseudo-labels for training 2D-LiDAR-based person detectors, using bounding boxes generated from an image-based person detector with a calibrated camera. We analyzed the quality of pseudo-labels by com-

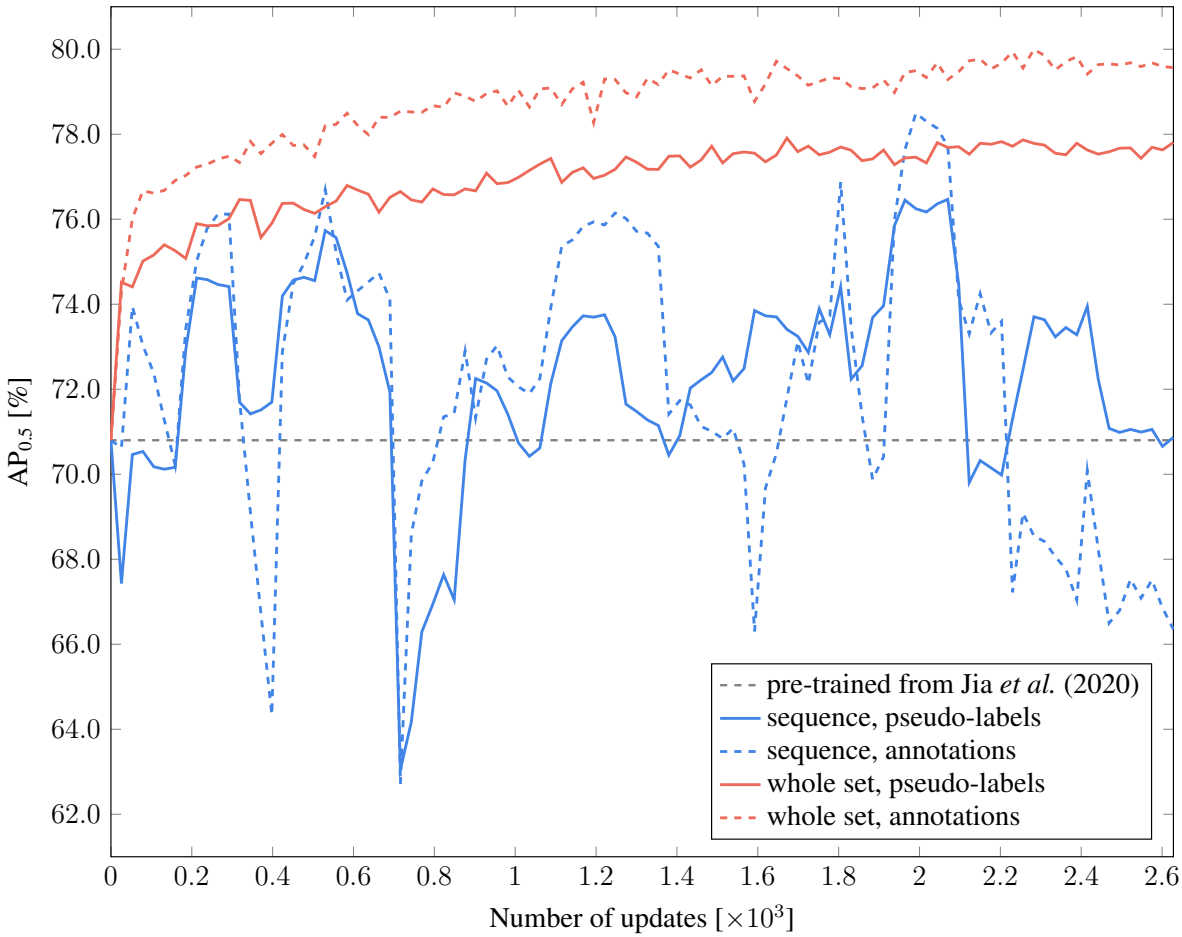


Figure 5.4: Performance at different steps of fine-tuning a pre-trained DROW detector with partially Huberized cross-entropy loss. Data is shuffled either on the whole training set or within each sequence, mimicking different amounts of curated data. With diverse batches, the detector performance improves significantly even with only a small number of updates. Fine-tuning longer using the default training schedule yields 79.2 and 82.3 percent $AP_{0.5}$ for pseudo-labels and annotations, respectively.

paring them against ground truth annotations and proved their validity. Experiments were conducted to train or fine-tune DROW and DR-SPAAM detectors using pseudo-labels, and these self-supervised detectors outperformed the detectors trained on annotations using a different dataset. Even stronger detectors were obtained by combining pseudo-labels with robust training techniques.

The proposed method provides an effective way to bridge the domain gap between data encountered during training and deployment, and can lead to novel robotic applications. For instance, a self-adjusted detector can be implemented during deployment, where a mobile robot generates and stores pseudo-labels while operating and uses them to fine-tune its detector during off hours, such as while charging. Our preliminary experiments in online fine-

tuning demonstrate the feasibility of this approach, but they also emphasize the importance of carefully selecting training examples to ensure sufficient diversity. In environments where deploying a camera is prohibited due to privacy concerns, but detecting persons is still necessary, a mobile robot can undergo a training phase with an attached camera and detach the camera during deployment.

Although demonstrated with 2D range data, the proposed approach in principle can work with point cloud from 3D LiDAR sensors. Further research should examine its capability for generating pseudo-labels for 3D LiDAR sensors, and their potential for training a 3D-LiDAR-based detector. Contrastive learning has been shown to be a powerful paradigm for self-supervised learning (He *et al.*, 2020b; Chen *et al.*, 2020d), where predictions based on variants on underlying samples are contrasted as weak supervision signal. The proposed method, which links signals from different modalities (LiDAR sensors and images), can in principle be paired with a contrastive framework, where the learning signals come from the agreement or disagreement of predictions from individual modalities. Such cross-modal contrastive learning is recently explored by Ding *et al.* (2023) for self-supervised scene flow estimation from radar, LiDAR, and image sensors.

6 How Does a 2D-LiDAR-Based Person Detector Compare Against a 3D-LiDAR-Based One?

6.1 Introduction

Previous chapters have introduced DROW and DR-SPAAM, state-of-the-art person detectors using 2D LiDAR sensors, which are well-suited for deployment on mobile robots with limited compute. Despite the outstanding results on benchmark datasets, one question remains: would the sparse measurement of a 2D LiDAR sensors, whose scans are constrained to one single plane, limits its capability for detecting persons. One alternative is to resort to 3D LiDAR sensors which scan multiple plane, making persons in the scene more recognizable (see Figure 6.1). However, 3D LiDAR sensors often carry a higher price tag, and demand greater power and compute resources, both of which are a scarcity on mobile robots. Properly understanding the performance differences between 2D and 3D-LiDAR-based person detection, such as accuracy and runtime, plays an important role for well-informed robot design.

To this end, this chapter presents a comparative study, where we compare two state-of-the-art detectors, one for each sensor type, on a common benchmark. We focus specifically on scenarios that are encountered by *e.g.* social robots or service robots, which are becoming increasingly relevant in recent years. These scenarios differ significantly from driving scenarios in terms of density and proximity of surrounding persons, sensor height, and encountered objects. Using the state-of-the-art CenterPoint (Yin *et al.*, 2021a) and DR-SPAAM (Jia *et al.*, 2020) as representatives for methods based on 3D and 2D LiDAR sensors, we conduct a series of experiments on the large-scale, publicly available JackRabbot dataset (Martin-Martin *et al.*, 2021), comparing the performance differences between these two sensor types.

This chapter is based on the 2022 IROS conference paper titled *2D vs. 3D-LiDAR-based Person Detection on Mobile Robots* by Jia, Hermans, and Leibe. I was responsible for designing the comparative experiments presented, analyzing the results, and writing. Hermans and Leibe contributed through regular discussions and revisions.

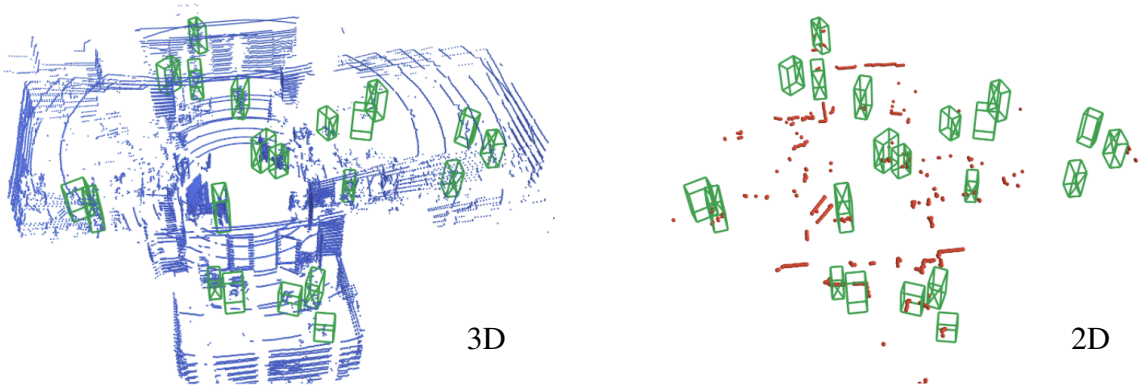


Figure 6.1: Scans from a 3D and a 2D LiDAR sensor with person bounding boxes annotations. The vast difference between the scans are apparent, but how do these two sensors compare, when it comes to person detection? Understanding how these sensors affect this task is crucial for designing robots that are intended to operate around humans.

Our experiments reveal that the 3D-LiDAR-based CenterPoint provides superior detection accuracy compared to the 2D-LiDAR-based DR-SPAAM, but when only considering persons visible in the sensor data, these two methods perform on par. However, when considering only persons visible in the sensor data, both methods perform similarly. For applications where detecting nearby persons is the primary requirement (nearby persons are typically not occluded), 2D LiDAR sensors, which are often available on many mobile robots for mapping and safety purposes, can provide sufficient detection capabilities. Additionally, DR-SPAAM offers higher inference speeds, which is advantageous for mobile robots with limited onboard compute or power. Nevertheless, 3D LiDAR sensors are better suited for general purpose person detection, as they are more robust to occlusion. Our experiments also reveal that both methods can produce well-localized detections and are similarly robust against dense crowds. A closer examination of positive and negative predictions reveals that a non-negligible number of persons are detected by only one of the detectors. Therefore, an ensemble-based approach is recommended for maximum reliability as the primary design objective.

A similar study was recently conducted by Linder *et al.* (2021), in which the authors compared a range of different 2D, 3D, and RGB-D-based person detectors. While they compare a large number of older and more recent detectors, we limit our experiments to the two current state-of-the-art (and open-source) approaches and perform our experiments on the larger, public, and more diverse JackRabbot dataset (Martin-Martin *et al.*, 2021). Their findings with respect to 2D and 3D-LiDAR-based person detection are in line with ours, however, we delve deeper into the differences, yielding interesting additional insights in how these two types of sensors compare for the task of person detection.

6.2 Experimental Setup

6.2.1 Person Detectors

The experiments in this chapter are conducted with CenterPoint (Yin *et al.*, 2021a) and DR-SPAAM (Jia *et al.*, 2020), state-of-the-art detectors based on 3D and 2D LiDAR sensors respectively. For completeness, we briefly summarize the key concepts of both detectors and direct interested readers to the primary publications for a more thorough overview.

CenterPoint (Yin *et al.*, 2021a). The network takes a voxelized 3D point cloud as input and uses either a VoxelNet (Zhou and Tuzel, 2017) or PointPillars (Lang *et al.*, 2019) to extract a 2D feature map on the bird's-eye-view (BEV) plane. From the extracted features, a center head is used to produce heatmaps, corresponding to x, y locations of bounding box centers on the BEV plane. This center head is supervised with 2D Gaussians produced by projecting bounding box centers onto the BEV plane, together with the focal loss (Lin *et al.*, 2017). In addition, regression heads are used to obtain x, y center refinement, center elevation, box dimensions, and orientation encoded as sine and cosine values. These regression heads are supervised at the ground truth centers with an $L1$ loss. In this work, we use the CenterPoint with a VoxelNet backbone for our experiments.

DR-SPAAM (Jia *et al.*, 2020). The input to DR-SPAAM the range component of a 2D LiDAR scan encoded using polar coordinates. A preprocessing step is used to extract points within small angular windows, and each window is normalized and passed into a 1D CNN. The network has two branches: a classification branch, which classifies if the window center is in the proximity of a person, and a regression branch, which regresses an offset from the window center to the x, y center location of the person. The classification branch is supervised with a binary cross-entropy loss, whereas the regression branch is supervised only for positive windows, using an $L2$ loss. Finally, the predictions from all windows are postprocessed with a distance-based non-maximum-suppression to obtain the final detections.

6.2.2 Datasets

We conduct our main experiments with the JRDB dataset (Martin-Martin *et al.*, 2021), and use the nuScenes dataset (Caesar *et al.*, 2020) for settings involve pretraining.

JRDB (Martin-Martin *et al.*, 2021). JRDB contains 54 sequences collected with a mobile robot (the *JackRabbot*) moving in both indoor and outdoor environments on university campuses. These sequences are split into 27 sequences for training and validation, and 27 for testing. The robot is equipped with two 3D LiDAR sensors (Velodyne 16 Puck LITE), on the upper and lower part of the robot respectively, each producing scans with approximately 18,000 points. Persons in the environment are annotated with 3D bounding boxes, with a total of roughly 1.8M annotations across the dataset. At the moment of writing, JRDB is the only large-scale dataset that focuses on mobile robot scenarios, while also featuring LiDAR point cloud with 3D person annotations.

In addition, the JackRabbot is equipped with two 2D LiDAR sensors (SICK LMS500-20000), front and rear facing respectively. They are mounted at the height of the lower legs, and their scans are merged to a single 360° scan, having 1091 points. Jia *et al.* (2021) used these scans to evaluate 2D-LiDAR-based person detectors by synchronizing and aligning them to the 3D annotations. In this work, we use 2D LiDAR scans from JRDB, following the same data preparation procedure. All our reported numbers are based on the JRDB validation set obtained from the standard train-validation split.

nuScenes (Caesar *et al.*, 2020). The nuScenes dataset contains 1,000 short sequences (approximately 20 seconds each) collected from driving vehicles. These sequences are split into a train, validation, and test set, having 700, 150, 150 sequences respectively. The dataset is captured using a 32 beam LiDAR, producing scans with approximately 30,000 points at 20 Hz. Compared to JRDB, these scans cover a larger area and have significantly sparser point clouds. Every tenth frame (0.5 seconds apart) is annotated with 3D bounding boxes, with a total of 10 classes, one of which being pedestrian. A common practice is to combine the unlabeled scans to obtain a denser point cloud Caesar *et al.* (2020); Zhu *et al.* (2019a); Yang *et al.* (2020); Yin *et al.* (2021a) during training and inference.

6.2.3 Training Setup

CenterPoint. To train CenterPoint, we mostly use the same hyperparameters proposed by Yin *et al.* (2021a). We follow the same training procedures and data augmentation scheme, with the AdamW optimizer (Loshchilov and Hutter, 2019), but train the network for 40 epochs with batch size 32, max learning rate 1e-3, weight decay 0.01, and momentum 0.85 to 0.95. Following Yin *et al.* (2021a), we use 0.1×0.1×0.2 m grids to voxelize the input point cloud, and limit detection range to [-51.2m, 51.2m] for the x and y axes, and [-5m, 3m] for the z axis. However, our experiment shows that this original setting, which is intended for nuScenes dataset, is not optimal for JRDB. Thus, we additionally experiment with a more fine-grained voxelization using 0.05×0.05×0.2 meter grids and a limited detection range of [-25.6m, 25.6m] for the x and y axes, and [-2m, 8m] for the z axis. This reduction in voxel size and detection range is motivated by the fact that scenes in JRDB are typically of a smaller scale compared to those in nuScenes.

When fine-tuning a network pre-trained on nuScenes, we reduce the training duration to 10 epochs. This resulted in the same performance as a complete 40 epoch training schedule on top of the pre-trained network. Since nuScenes provides additional time increments and reflectance features for every point, we cannot directly fine-tune checkpoints provided by the authors. Instead we retrain the network on nuScenes, using only x, y, z coordinate of the points as input features and fine-tune this network.

DR-SPAAM. For DR-SPAAM, we follow the procedure proposed by Jia *et al.* (2021) and train the network for 20 epochs with a batch size of 6, using the same pre and postprocessing hyperparameters. The original DR-SPAAM only predicts x, y locations of the person, providing no information related to bounding boxes. In order to compare with 3D-LiDAR-based

Table 6.1: Performance of different CenterPoint and DR-SPAAM variants on the JRDB validation set, evaluated under Default (De.) and 2D-visible (Vi.) setting. *:3D Bounding boxes are obtained using predicted BEV boxes and the average box height from the training set.

		AP_{box}		AP_{BEV}		$AP_{centroid}$	
		De.	Vi.	De.	Vi.	De.	Vi.
CenterPoint	trained on nuScenes	26.3	26.0	30.9	28.5	35.3	32.1
	+ JRDB fine-tuning	58.2	64.2	61.2	67.1	69.5	75.0
	trained on JRDB	60.0	68.2	62.6	69.9	67.9	75.1
	+ fine-grained voxelization	66.0	75.0	67.1	76.5	70.1	78.1
	+ nuScenes pretraining	70.0	78.6	71.4	80.7	74.9	82.7
DR-SPAAM	BEV	34.1*	58.2*	40.8	67.3	46.6	74.8
	Centroid only	–	–	–	–	47.6	77.2

methods, we experiment with modifying the regression branch, additionally predicting the bounding box width, length, and orientation. The width and length are parameterized as

$$t_w = \log(w/\bar{w}) \quad t_l = \log(l/\bar{l}) \ , \tag{6.1}$$

where $\bar{w} = 0.5$ and $\bar{l} = 0.9$ represent the average box width and length in the JRDB training set. The orientation is parameterized by its sine and cosine as done by Yin *et al.* (2021a). Both the size and orientation regression are supervised with an $L2$ loss, with a weighting factor of 0.2 for the orientation. In addition, assuming persons do not vary significantly in height, we experiment with generating 3D bounding boxes, using the predicted BEV boxes and the average height from the training set. We did not experiment with regressing box height from the 2D LiDAR data.

6.2.4 Metrics

Following the standard for object detection, we use the Average Precision (AP) as the main evaluation metric. We use three variants, AP_{box}, AP_{BEV}, and $AP_{centroid}$, each having different criteria for assigning detections to ground truth boxes. For AP_{box}, a detection can be assigned to a ground truth if their 3D box IoU is above 0.3 (default threshold used by JRDB). AP_{BEV} relaxes the requirement to a 2D box IoU criterion on the bird's-eye-view boxes, discarding the requirement related to box height and elevation. $AP_{centroid}$ focuses on the center localization only, assigning a detection to a ground truth if their x, y location difference is smaller than 0.5m, dropping the requirements on box size and orientation. In all these three variants, a ground truth can only be matched with one detection.

Since ground truth boxes have different visibility to 3D and 2D LiDAR sensors, we compute all three APs under two settings. In the first setting, we evaluate detections against ground truth boxes that contain at least 10 points from the 3D LiDAR sensor. This corresponds to the

default evaluation procedure of JRDB and we thus refer to it as the *default* evaluation. In the second setting, which we term *2D-visible*, we evaluate detections against ground truth boxes which have at least 5 points from the 2D LiDAR sensor within a 0.5m radius from the box x, y centroid. These two evaluation settings enable us to examine the detector performance from both types of sensors, independent of factors that cause persons to be completely invisible to the sensor (thus being impossible to detect).

6.3 Results

6.3.1 2D vs. 3D-LiDAR-based Detection Performance

The performance of CenterPoint and DR-SPAAM are compared in Table 6.1. Under the default evaluation setting, DR-SPAAM trails CenterPoint trained on JRDB with fine-grained voxelization by a significant 31.9% AP_{box} and 26.3% AP_{BEV}. The gap on AP_{box} is greater since DR-SPAAM can not estimate the bounding box height. However, when evaluated against ground truth that is visible to 2D LiDAR sensors, this gap shrinks to 9.2% AP_{BEV}. For the $AP_{centroid}$, DR-SPAAM and CenterPoint differ by 3.3%. A DR-SPAAM that regresses centroid only, as the original one from Jia *et al.* (2020), has a further improved performance, leaving a small performance gap of 0.9% $AP_{centroid}$ to CenterPoint trained only with JRDB.

These numbers show that, when only the centroid locations of visible persons are concerned, detectors using 2D or 3D LiDAR sensors have a similar performance, despite the vast information gap between the detector input (a simple planar scan of the legs *vs.* a full-body surface scan). Thus, for tasks like person avoidance or following, a 2D LiDAR sensor is sufficient. General purpose detection, however, is better carried out with 3D-LiDAR-based detectors, since it is easy for persons to be fully occluded, and thus impossible to detect, from the scan plane of 2D LiDAR sensors. When designing robots, the choice between 2D and 3D LiDAR sensors should be made with task requirements in view.

The first row of Table 6.1 furthermore shows that a quite big domain gap exists between autonomous driving and mobile robot scenarios. There are many pre-trained 3D-LiDAR-based detectors for driving scenarios, but it is unlikely that they will perform well on robotic tasks involving close contact with persons. Fortunately, this domain gap can be bridged by fine-tuning and adapting suited voxel sizes. These numbers highlight the importance of training on data similar to that will be encountered during deployment. For 2D LiDAR sensors, estimates about this domain gap are not directly possible, since they are rarely used in outdoor driving scenarios.

6.3.2 Detailed Detector Comparisons

The remainder of this section focuses on comparison of CenterPoint and DR-SPAAM in specific aspects. The best variants of each detector—nuScenes pretrained and JRDB fine-tuned

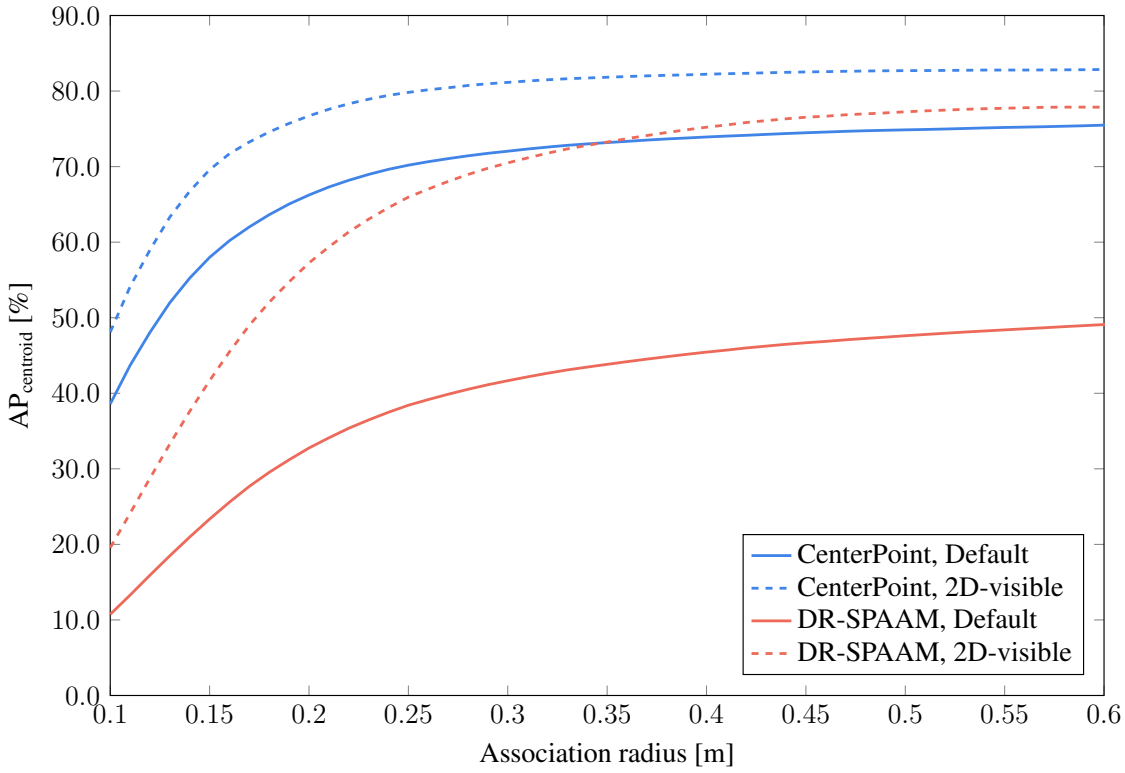


Figure 6.2: $AP_{centroid}$ evaluated at different ground truth association radii. The CenterPoint curves are slightly steeper at the start, suggesting a better localization of the detections.

CenterPoint with fine-grained voxelization and centroid-only DR-SPAAM—are used for these comparisons with $AP_{centroid}$ as the evaluation metric.

Localization Accuracy. The default ground truth association radius of 0.5m used by $AP_{centroid}$ does not capture how well localized the different detections are. To further evaluate the localization accuracy, we run the evaluation with a varying association radius and plot the development of the performance in Figure 6.2. Note that for this evaluation the gradient of a curve is interesting and not so much the absolute performance.

After a quick increase, the performances slowly saturate, suggesting that most detections are made with a reasonable accuracy. The main difference between CenterPoint and DR-SPAAM is the slope of the curves for small association radii. Here the performance for CenterPoint increases faster, suggesting a slightly better localization of its detections. Furthermore, the CenterPoint curves are almost saturated after an association radius of $\sim$0.3m, indicating an upper bound for its localization inaccuracy. Here the DR-SPAAM curves are still slightly increasing, meaning that some of the detections are less well localized. Note that small inaccuracies exist in the annotations. Thus, detection scores obtained with an overly small association radius such as 0.1m or less do not provide meaningful information.

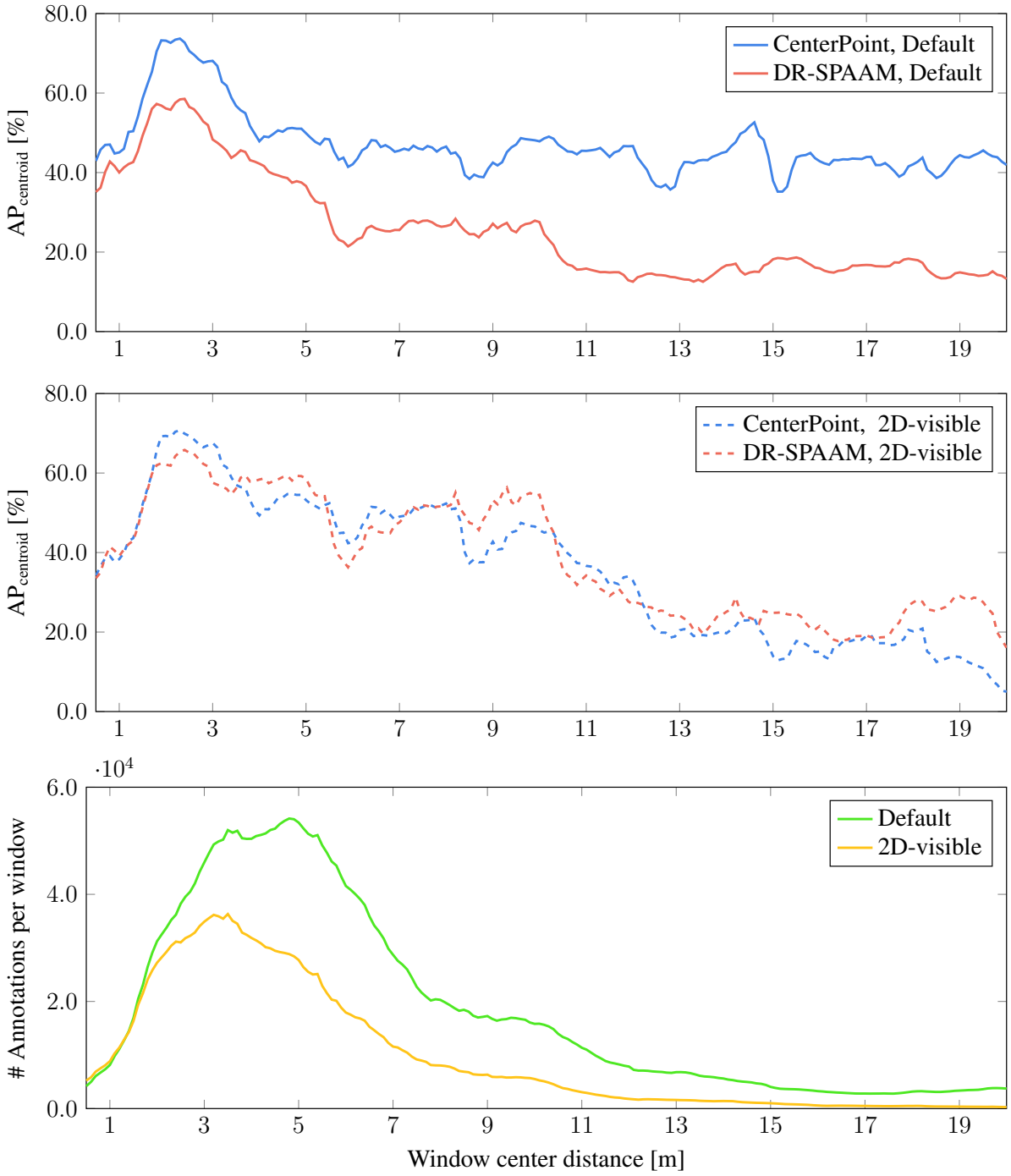


Figure 6.3: $AP_{centroid}$ evaluated within a 2m window at different distances from the sensor. The top and middle plot show how the performance develops across the distance for the default and 2D-visible evaluation respectively. The bottom plot shows the number of annotations within the windows for both evaluations.

Distance-based Evaluation. The density of LiDAR data varies significantly with the distance to the sensor, which can affect the performance of detectors. Because of that we run an additional sliding-window evaluation, where we only consider ground truth annotations within a 2m window at varying distances from the sensor. For a meaningful evaluation we also constrain the detections to be in or near the evaluation window, as to not report unnecessary false positives. We do this by extending the window by 0.5m (the association radius) to the front and back and only consider detections within this larger window. This will still introduce additional false positives but also increase the true positive count, effectively being a trade-off between precision and recall. For this reason the overall performance is lower than reported in Table 6.1, but here the relative performance between the two detectors is the interesting part.

The top and middle plots in Figure 6.3 shows how the detector performances behave for this experiment. In all evaluations the performance is best within the first few meters and drops at higher distances. For the default evaluation, CenterPoint performs significantly more stable across the whole range, whereas the DR-SPAAM performance decreases more at higher ranges. This means that CenterPoint is better equipped to deal with the large variation of point densities, potentially due to the fact that objects are still visible to multiple LiDAR beams, whereas the 2D LiDAR only uses a single beam. For the 2D-visible evaluation, the performance of both detectors is surprisingly similar across the whole range, suggesting that for visible persons, both detectors have similar range-robustness.

The bottom plot in Figure 6.3 shows how many ground truth annotations are present within the windows for both types of evaluations. Here it becomes apparent that, from 2m and onward, a significant number of the persons are invisible to 2D LiDAR scans, and thus impossible to be detected. The largest chunk of annotations is found within the first 12m. After that only few annotations remain, for both the default and 2D-visible evaluation, and the evaluation becomes less meaningful. This highlights another difference between JRDB and autonomous driving datasets, where persons are regularly perceived at much greater distances.

The Influence of Crowds. People are often perceived while interacting amongst each other, typically resulting in them being quite close together. This results in many occlusions and could potentially lead to a more difficult task of detecting the separate persons properly. To investigate how the detection performance changes with the object density of a scene, we look at a radius of 1.5m around every ground truth annotation and count how many other annotations can be found. We categorize all annotations by this "group size" and perform a separate evaluation for all the different sizes. Figure 6.4 shows the result of this experiment. Even though the overall detection performance seems to be reducing with larger group sizes, the detectors do not completely fail for larger group sizes and the effect seems to be very similar for both detectors. What Figure 6.4 also shows, is that a significant amount of people in the JRDB dataset are observed in the vicinity of at least one other person making it significantly more realistic than some older datasets where one would often only observe a single person in an empty scene.

Runtime. The runtime is a typical practical constraint when deploying detectors. Especially in robotic applications, computational resources are often limited and we cannot rely

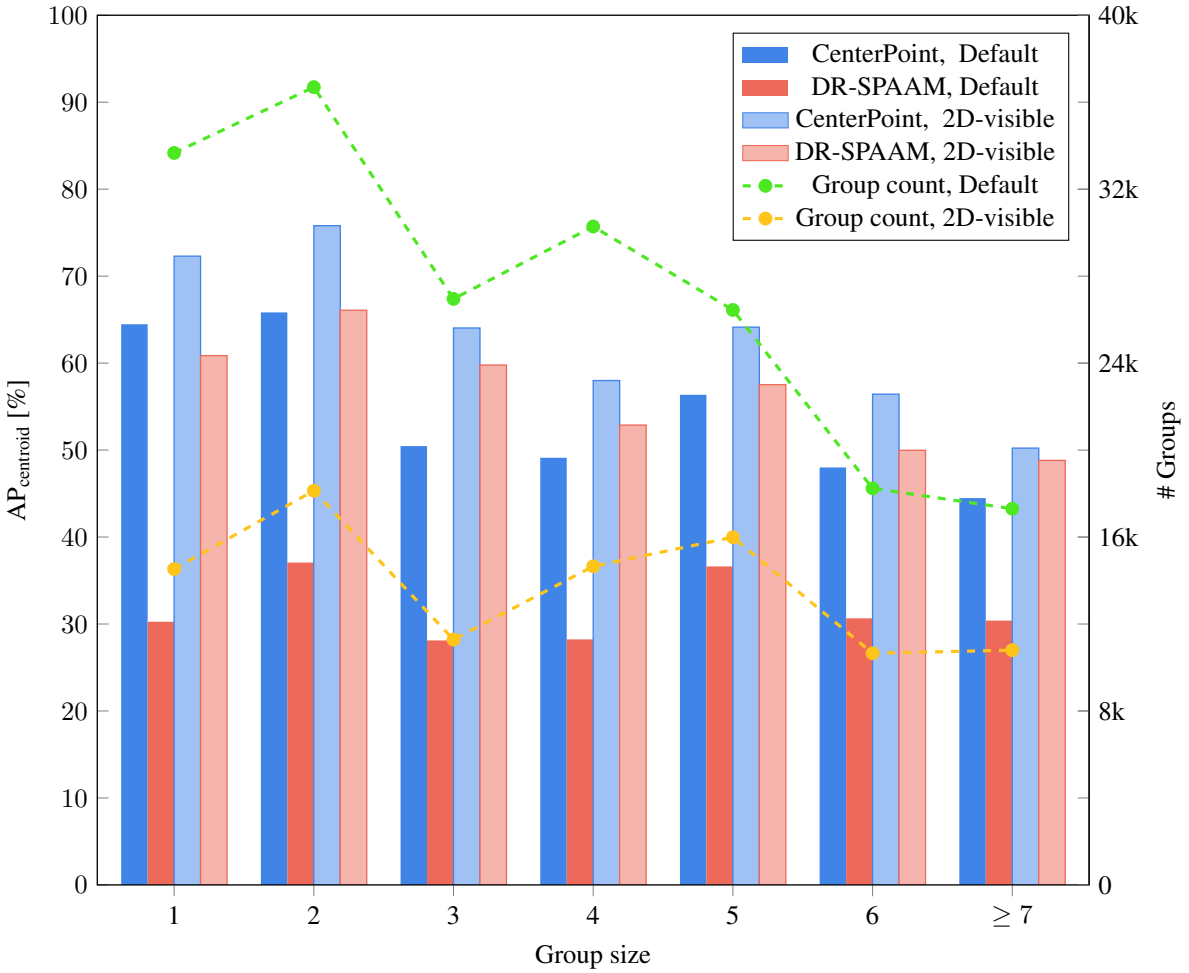


Figure 6.4: Evaluation split up by how crowded it is around annotations. While the performance for larger group sizes decreases a little, the effect is less pronounced than might be expected. We additionally plot how many groups of a specific size are present in the data, showing that JRDB is indeed fairly crowded.

on powerful desktop GPUs. Table 6.2 shows the runtime of the evaluated detectors on three different platforms: a powerful desktop machine, a laptop equipped with a decent GPU, and the lower-powered Jetson AGX Xavier. Preprocessing steps such as computing cutouts and voxelization, as well as postprocessing steps such as non-maximum-suppression, are included in the measurement, and no batching is used for inference. In other words, these numbers reflect the end-to-end runtime when the detector is deployed. Both DR-SPAAM and CenterPoint achieve real-time performance on strong desktop or laptop GPUs, but not on embedded GPUs. In general, DR-SPAAM is roughly twice as fast as CenterPoint, with the margin being smaller on the Jetson AGX. We additionally measured the memory usage on the Jetson

Table 6.2: Detector inference speed (frames per second) on three different platforms.

	Desktop (TITAN RTX)	Laptop (RTX 2080)	Jetson AGX Xavier
CenterPoint	32.4	19.8	6.0
DR-SPAAM	59.1	37.1	8.9

Table 6.3: DR-SPAAM performance and inference speed (frames per second) on Jetson AGX Xavier with different spatial subsampling.

Subsampling factor	$AP_{centroid}$ (Default)	$AP_{centroid}$ (2D-visible)	FPS
1	47.6	77.3	8.9
2	46.5	76.5	18.2 ($\times$2.0)
3	45.2	75.9	26.6 ($\times$3.0)
4	43.7	74.9	32.5 ($\times$3.7)
5	42.0	73.3	36.4 ($\times$4.1)

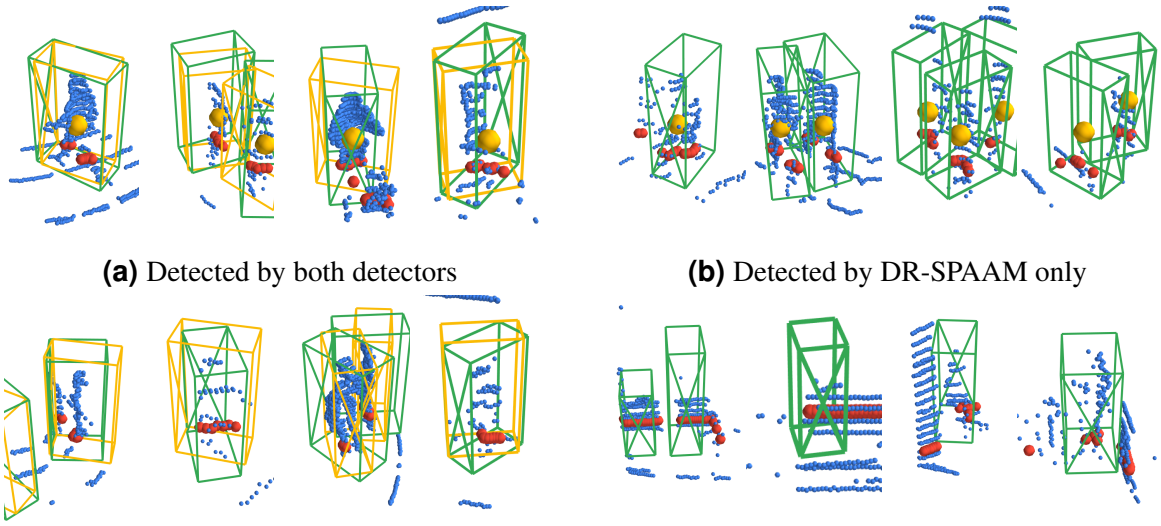
AGX, 5311MB for DR-SPAAM, 6318MB for CenterPoint. These numbers are likely to vary depending on specific system setups (we used L4T 32.4.4 and PyTorch 1.6).

DR-SPAAM can, thanks to its *cutout*-based design, obtain higher runtime by subsampling the scan, without significantly lowering the detection performance (Jia *et al.*, 2020). Table 6.3 shows the performance and runtime of DR-SPAAM with different subsampling factors on the Jetson AGX. With 3 times subsampling, DR-SPAAM can run at 26.6 FPS, while losing only 1.4% AP. In applications where the onboard computation is limited, DR-SPAAM presents a more favorable trade-off between performance and runtime, compared to the more accurate, yet slower CenterPoint.

6.3.3 Qualitative Results

Figure 6.5 shows several qualitative results of both detectors. These results are obtained by using a detection threshold resulting in an equal error rate (EER), meaning the precision is equal to the recall for that threshold. We show cases where both detectors, one of the two, or neither detected a person. We specifically focus on the true positives and false negatives as these cases should contain a person. The predicted centroids, either coming from CenterPoint or DR-SPAAM, are well-localized to the actual person. The error in orientation estimation sometimes leads to misaligned boxes for CenterPoint, but the overlap with ground truth boxes is often sufficiently high.

Even though the numbers so far suggested that CenterPoint typically outperforms DR-SPAAM, we can here see that there are also cases where only one of the two detectors is able to detect a person. While Figure 6.5 shows some cases where it clearly makes sense



(a) Detected by both detectors

(b) Detected by DR-SPAAM only

(c) Detected by CenterPoint only

(d) Missed by both detectors

Figure 6.5: Qualitative results from both detectors evaluated at equal error rates. Green boxes represent the ground truth, yellow boxes/spheres are detections by CenterPoint and DR-SPAAM respectively. The blue and red spheres are the 3D and 2D LiDAR points. These examples are picked from the less crowded areas to avoid visual clutter. Note that in many cases where one of the detectors fails, the data seems to be sufficient for a detection to be possible.

Table 6.4: Detection statistics obtained using an equal error rate threshold for both CenterPoint and DR-SPAAM. The majority of persons is detected by both detectors, whereas a significant amount is only detected by one of the two.

			Detected by			
	Total GT		Both	CenterPoint	DR-SPAAM	None
Default	189 579	(50.7%)	95 711 (50.5%)	53 136 (28.0%)	7 212 (3.8%)	33 520 (17.7%)
2D-visible	96 039		71 619 (74.6%)	12 432 (12.9%)	5 263 (5.5%)	6 725 (7.0%)

that a detector fails, *e.g.* due to missing points or partial occlusions. Most of the cases where only one of the detectors fails clearly show a person though and it is unclear why it was not detected. Interestingly, many ground truth bounding boxes missed by both detectors indeed do not look like they contain a person, potentially suggesting annotation errors.

To show these are not only a small set of cherry picked cases, Table 6.4 shows exactly how often these four different cases happen. While CenterPoint is more often the only detector to detect a person, there is a small but significant number of people only detected by DR-SPAAM as well. This suggests that an ensemble of both methods could be an interesting approach when both sensors are available.

6.4 Discussion

In this chapter we investigated differences between 2D and 3D-LiDAR-based person detection for mobile robots. For this we perform direct comparisons between the state-of-the-art CenterPoint (Yin *et al.*, 2021a) and DR-SPAAM (Jia *et al.*, 2020) detectors on the JackRabbot dataset (Martin-Martin *et al.*, 2021). We found that, when only visible persons are considered, detectors from both 2D and 3D LiDAR sensors perform on a similar level, but in 2D LiDAR sensors persons are more prone to being invisible to the sensor (and thus impossible to detect). While 3D LiDAR sensors overall provide a more robust solution to person detection, 2D LiDAR sensors provide better a trade-off between performance and runtime, a favorable trait for mobile robots with limited onboard computations.

Further analysis of the detectors showed that overall both their detections are well-localized and they are not significantly affected by crowds. Since a significant number of people are only detected by one of the two detectors, if computational resources allow it, an ensemble of the detectors could be an interesting possibility to further boost the detection performance. While one might have assumed the superiority of 3D-LiDAR-based person detection, the similar performance and robustness on visible persons for these two sensor types comes as somewhat of a surprise.

We performed all experiments with two state-of-the-art LiDAR-based detectors, while we could have tried to adapt CenterPoint to run on 2D LiDAR data, allowing for a more direct comparison. However, this would require a significant amount of tuning to make sure CenterPoint yields a representative performance on 2D LiDAR and our goal is not to develop a new 2D-LiDAR-based detector. Instead, we here rely on an existing and well-tuned state-of-the-art method. With our detector evaluations, we have set a lower-bound for the person detection performance on JRDB using 2D and 3D LiDAR sensors, which will be further improved by future developments.

Apart from LiDAR-based person detection, one could consider other sensor modalities. In particular, image-based person detectors have a long history. While the person class is now typically seen as one of many classes by most deep-learning-based detectors (Ren *et al.*, 2015; Carion *et al.*, 2020), a whole range of person-specific detectors existed before (Dalal and Triggs, 2005; Benenson *et al.*, 2012). Furthermore, RGB-D based person detectors are frequently used in robotics, which have the additional advantage that a person's position can be estimated (Jafari *et al.*, 2014; Lewandowski *et al.*, 2019; Linder *et al.*, 2020). While both types of image-based detectors can perform robust person detection, they have the drawback that the field-of-view of most cameras is fairly limited, as also shown by Linder *et al.* (2021). Image-based detectors will likely remain a viable source for person detections, however, a single LiDAR sensor can be sufficient to cover the complete surrounding, which would typically require multiple cameras, and thus significantly reduces compute requirements.

7 GHA: Global Hierarchical Attention for 3D Point Cloud Analysis

7.1 Introduction

Processing point cloud is fundamental in many robotic applications, including person detection. Recent years have witnessed a significant surge in interest for point cloud analysis (Qi *et al.*, 2017b; Choy *et al.*, 2019a; Thomas *et al.*, 2019). Due to the lack of inherent order in point cloud representation, methods aiming to directly process point clouds need to exhibit permutation invariance (Qi *et al.*, 2017b), as discussed in Chapter 2. The attention mechanism, being a set-to-set operation that is invariant to permutation, is a promising candidate for this task. Originally introduced for language tasks (Vaswani *et al.*, 2017), transformers and the attention mechanism have since been successfully applied to a range of computer vision tasks involving images as input (Dosovitskiy *et al.*, 2021b; Carion *et al.*, 2020; Liu *et al.*, 2021b; Caron *et al.*, 2021). Recently, several studies have applied transformer models to 3D tasks, including classification (Guo *et al.*, 2021), detection (Pan *et al.*, 2021; Liu *et al.*, 2021a; Misra *et al.*, 2021; Mao *et al.*, 2021b), and segmentation (Zhao *et al.*, 2021; Park *et al.*, 2022), achieving promising results.

There are two major challenges when applying the attention mechanism to point clouds. Firstly, the required memory quickly grows out of bounds for large point clouds, due to the quadratic space complexity of the attention matrix. Secondly, the global receptive field of an attention layer allows the network to easily overfit to distracting long range information, instead of focusing on local neighborhoods, which provide important geometric information (Raghu *et al.*, 2021; Choy *et al.*, 2019b; Zeng *et al.*, 2017). To solve these two problems, existing point cloud methods often resort to computing attention only within a local neighborhood, known as *local attention*, and channel global information either through customized

This chapter is based on the 2022 GCPR conference paper titled *GHA: Global Hierarchical Attention for 3D Point Cloud Analysis* by Jia, Hermans, and Leibe. I was responsible for the method design, experiment evaluation, and writing. Hermans and Leibe contributed through regular discussions and revisions.

multi-scale architectures (Zhao *et al.*, 2021; Pan *et al.*, 2021) or via modeling interactions between point clusters (Pan *et al.*, 2021).

In this chapter we present a novel attention mechanism, called *Global Hierarchical Attention* (GHA), which natively addresses the aforementioned two challenges. The memory consumption of GHA scales linearly with respect to the number of points, a significant reduction to the regular quadratic attention. In addition, GHA by design embodies the inductive bias that encourages the network to focus more on local neighbors, while retaining a global receptive field.

The core of GHA is a series of hierarchical levels, connected using *coarsening* and *interpolation* operations. Given an input point cloud, the coarsening operation is used to construct hierarchies with different levels of details. Within each hierarchy level, local attention is computed, and the results from all levels are collated via the interpolation operation. The output of GHA can be computed without needing to reconstruct the memory-intensive attention matrix, and it approximates the output of regular attention, while giving more emphasis to the nearby points. Our method is inspired by the recent work from Zhu and Soricut (2021), which produces an efficient hierarchical approximation of the attention matrix for 1D (language) sequences, but due to its reliance on the existence of a 1D order, it cannot be directly applied to point clouds.

We propose two realizations of GHA, compatible with either raw or voxelized point clouds. We design an add-on module based on GHA that can be easily inserted into an existing network. We experiment with various baseline networks on both 3D object detection and segmentation tasks. Our experiments show that GHA brings a consistent improvement to a range of point cloud networks and tasks, including +1.7% mIoU on ScanNet segmentation for the MinkowskiEngine (Choy *et al.*, 2019a), +0.5% mAP on nuScenes detection for CenterPoint (Yin *et al.*, 2021a), and +2.1% mAP_{25} and +1.5% mAP_{50} on ScanNet detection for 3DETR.

7.2 Related Work

7.2.1 Point Cloud Analysis

Existing architectures for analyzing point clouds fall into several categories. Projection-based methods (Kanezaki *et al.*, 2021; Lang *et al.*, 2019; Su *et al.*, 2018; Tatarchenko *et al.*, 2018) process the point clouds by projecting them onto 2D grids and applying 2D CNNs to the obtained image. Voxel-based approaches (Graham *et al.*, 2018; Li *et al.*, 2018; Zhou and Tuzel, 2017; Riegler *et al.*, 2017; Choy *et al.*, 2019a) use 3D CNNs on the voxelized point cloud. In addition, there are methods that do not rely on discretization and operate directly on point clouds, via PointNet (Qi *et al.*, 2017a), continuous convolutions (Xu *et al.*, 2018b; Thomas *et al.*, 2019; Wu *et al.*, 2019b; Mao *et al.*, 2019), or graph neural networks (Li *et al.*, 2021a; Wang *et al.*, 2019b,c; Zhao *et al.*, 2019; Landrieu and Simonovsky, 2018; Landrieu and Boussaha, 2019; Jiang *et al.*, 2019). Recent works (Liu *et al.*, 2021a; Pan *et al.*, 2021; Guo

et al., 2021; Zhao *et al.*, 2021; Fan *et al.*, 2021a; Park *et al.*, 2022; Misra *et al.*, 2021) have attempted to adapt transformers and attention mechanisms (Vaswani *et al.*, 2017) for point cloud processing, encouraged by their recent success in language and vision tasks.

7.2.2 Attention for Point Clouds

Several works have used attention mechanisms for point clouds. Group-free Transformer (Liu *et al.*, 2021a) is an object detection framework that uses a PointNet++ (Qi *et al.*, 2017a) backbone to generate initial bounding box proposals, and a stack of self and cross-attention layers for proposal refinement. Pointformer (Pan *et al.*, 2021) is another detection network which builds a multi-scale backbone network via a series of customized local and global attention mechanisms. VoTr (Mao *et al.*, 2021b) proposed to replace the sparse voxel convolution in the SECOND detector (Yan *et al.*, 2018) with local and dilated attention and obtained good detection results. These works rely on specially crafted attention mechanisms and network architectures. 3DETR (Misra *et al.*, 2021) is the first work that uses a standard end-to-end transformer architecture for object detection, following the framework laid out by the image-based DETR object detector (Carion *et al.*, 2020). In addition to detection, attention mechanisms have also been used on tasks like analyzing point cloud videos (Fan *et al.*, 2021a), registration (Wang and Solomon, 2019), or normal estimation (Guo *et al.*, 2021).

Two works have specifically aimed at designing attention-based networks for different point cloud tasks. Zhao *et al.* (2021) proposed the Point Transformer, a multi-scale U-Net architecture with local attention computed among k-nearest neighbors. This model has demonstrated strong performance in shape classification on ModelNet40 (Wu *et al.*, 2015), object part segmentation on ShapeNet (Yi *et al.*, 2016), and semantic segmentation on S3DIS (Armeni *et al.*, 2016). Fast Point Transformer (Park *et al.*, 2022) enhances the Point Transformer architecture by operating on voxels, resulting in improved speed and performance.

Similar to these, GHA is intended as a general-purpose method that can be applied to a wide range of tasks. However, our focus is on the attention mechanism itself, rather than the network architecture. We conduct experiments by augmenting existing well-known networks with several attention layers, and compare the performance between networks using different types of attention (*e.g.* GHA, local, or regular). By excluding architecture design variables, we can draw more accurate conclusions about the effectiveness of the attention mechanism itself from our comparative experiments.

7.2.3 Efficient Transformers

A large body of works exists in the NLP community that aims to reduce the memory consumption of the attention mechanism (Zaheer *et al.*, 2020; Choromanski *et al.*, 2020; Zhu and Soricut, 2021; Wang *et al.*, 2020a; Tay *et al.*, 2020). The most relevant to our work is that of Zhu and Soricut (2021), which proposed a hierarchical attention mechanism to approximate the full attention matrix with linear space complexity. As they exploit the fixed order of a 1D sequence to represent the attention matrix in a matrix block hierarchy, their exact formulation

does not readily generalize to higher dimensions. In this work, we extend this idea to point cloud analysis and propose the GHA mechanism that can work with both 3D point clouds and voxels.

In the vision community, most methods resort to computing attention only within local windows (Liu *et al.*, 2021b; Zhao *et al.*, 2021) to reduce the memory consumption and rely on architectural level design to channel long-range information. Compared to these methods, GHA has the advantage of having a global field of view, without imposing additional constraints on the network architecture. As such, GHA can easily be integrated into a whole range of existing networks.

7.3 Method

7.3.1 Global Hierarchical Attention

The regular dot-product attention by Vaswani *et al.* (2017) is defined as

$$Z = \text{softmax}(\frac{QK^T}{\sqrt{d}})V \ , \tag{7.1}$$

where $Q, K, V \in \mathbb{R}^{N \times d}$ represents the token queries, keys, and values stacked as rows of the matrices. Here N is the number of tokens, and d is the dimension of the embedding space. This equation can be expressed in a more compact matrix form

$$\begin{aligned} A &= \exp[\frac{QK^T}{\sqrt{d}}] \\ D &= \text{diag}(A \cdot \mathbf{1}_N) \\ Z &= D^{-1}AV \end{aligned} \tag{7.2}$$

where $\mathbf{1}_N = [1, 1, \ldots, 1]^T$ represents a vector of 1 and $\exp[\cdot]$ denotes the element-wise exponential. The normalized attention matrix $D^{-1}A$ has $O(n^2)$ space complexity, making it memory intensive for large numbers of tokens.

Using Equation 7.2 to compute full attention on a LiDAR point cloud is in general not feasible, due to the large number of points in a scene. We propose a global hierarchical attention mechanism, which approximates the full attention while significantly reducing the memory consumption. The core of GHA is a series of hierarchical levels, augmented with three key ingredients: the *coarsening* operation and the *interpolation* operation that connect tokens between two hierarchies, and the *neighborhood topology* that connects tokens within a single hierarchy. These ingredients are illustrated in Figure 7.1.

Starting from $\tilde{Q}^{(0)} = Q$ (and identically for K and V), the coarsening operation

$$\tilde{Q}^{(h+1)} = \text{Coarsen}(\tilde{Q}^{(h)}, \mathcal{T}^{(h)}) \tag{7.3}$$

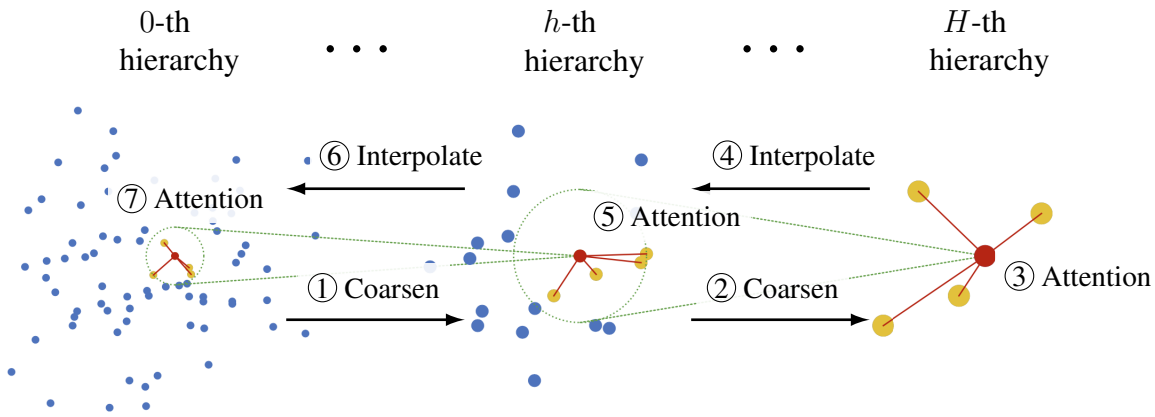


Figure 7.1: The operations of **one GHA layer** (not a complete network). The operation order is shown with circled numbers. See Section 7.3.1 for a detailed explanation.

recursively computes a down-sampled representation for each hierarchy level by averaging points within a defined neighborhood topology $\mathcal{T}^{(h)}$ until all remaining points belong to a single neighborhood, and the interpolation operation

$$\tilde{Y}^{(h)} = \text{Interp}(\tilde{Y}^{(h+1)}) \tag{7.4}$$

expands the coarsened representation to the spatial resolution of the level above (here Y represents any arbitrary embedding). With these two operation, one can first compute the coarsened $\tilde{Q}^{(h)}$, $\tilde{K}^{(h)}$, $\tilde{V}^{(h)}$ for all hierarchy levels h and approximate the attention weighted output Z via the recursion

$$\begin{aligned}
\tilde{A}_{ij}^{(h)} &= e^{\tilde{S}_{ij}^{(h)}} = e^{\frac{\tilde{Q}_i^{(h)} (\tilde{K}_i^{(h)})^T}{\sqrt{d}}} \\
\tilde{Y}_i^{(h)} &= \sum_{j \in \mathcal{T}_i^{(h)}} \tilde{A}_{ij}^{(h)} \tilde{V}_j^{(h)} + \text{Interp}(\tilde{Y}^{(h+1)})_i \\
\tilde{D}_i^{(h)} &= \sum_{j \in \mathcal{T}_i^{(h)}} \tilde{A}_{ij}^{(h)} + \text{Interp}(\tilde{D}^{(h+1)})_i \\
Z \approx \tilde{Z} &= \text{diag}(\tilde{D}^{(0)})^{-1} \tilde{Y}^{(0)}.
\end{aligned} \tag{7.5}$$

This procedure is summarized in Algorithm 2. Note that here, the attention at each level is only computed within the local neighborhood $\mathcal{T}$. For N points, the memory cost is proportional to

$$\begin{aligned}
\sum_{h=0}^{H} \frac{1}{k^h} Nk &= \frac{1 - k^{-(H+1)}}{1 - k^{-1}} Nk \\
&= \frac{k^{H+1} - 1}{k^{H-1}(k-1)} N
\end{aligned} \tag{7.6}$$

with $H = log_k N$ level of hierarchies. The sum of this series approaches $\frac{k^2}{k-1}N$ with $H \to \infty$, showing the complexity of GHA is upper bounded by a linear function. This linear complexity makes it possible to process large-scale point clouds.

The procedure outlined in Algorithm 2 assigns higher attention weights for closer points. Thus, GHA natively embodies the inductive bias to focus more on local neighborhoods while retaining the global connectivity. In this light, local attention (Zhao *et al.*, 2021; Park *et al.*, 2022) can be viewed as a special case of GHA which truncates all hierarchy below the initial input level.

Algorithm 2 GHA for Point Cloud Analysis

Input: $Q, K, V \in \mathbb{R}^{N \times d}$
Output: $\tilde{Z} \approx \text{softmax}(\frac{QK^T}{\sqrt{d}})V$
$\tilde{Q}^{(0)}, \tilde{K}^{(0)}, \tilde{V}^{(0)} \leftarrow Q, K, V$ ▷ Coarsening
for h in $(0, \ldots, H-1)$ **do**
  compute $\tilde{Q}^{(h+1)}$, $\tilde{K}^{(h+1)}$, $\tilde{V}^{(h+1)}$ via Equation 7.3
end for
$\tilde{Y}^{(H)}, \tilde{D}^{(H)} \leftarrow 0, 0$ ▷ Interpolation
for h in $(H-1, \ldots, 0)$ **do**
  compute $\tilde{Y}^{(h)}, \tilde{D}^{(h)}$ via Equation 7.5
end for
$\tilde{Z} \leftarrow \text{diag}(\tilde{D}^{(0)})^{-1}\tilde{Y}^{(0)}$

In order to apply GHA to both point-based and voxel-based methods, we propose two flavors of GHA. Taking a voxelized point cloud as input, **Voxel-GHA** uses average pooling as the coarsen operation, unpooling as the interpolation operation, and the local kernel window as the neighborhood topology $\mathcal{T}$. These operations can be efficiently accomplished using existing sparse convolution libraries (Choy *et al.*, 2019a; Yan, 2021). **Point-GHA** operates directly on the raw point clouds instead. It uses *kNN* as the neighborhood topology, and conducts coarsening via farthest point sampling

$$\tilde{Q}^{(h+1)} = \text{Sample}_{(h)}\left(\frac{1}{K}\sum_{j \in \mathcal{T}_i^{(h)}} \tilde{Q}^{(h)}\right) , \tag{7.7}$$

and interpolation based on nearest neighbor interpolation

$$\tilde{Y}^{(h)} = \text{NN}(\tilde{Y}^{(h+1)}) . \tag{7.8}$$

Having to compute *kNN* neighborhoods and perform sampling means Point-GHA has some computational overhead compared to its voxel-based counterpart. In a block composed of multiple GHA layers, the sampling and *kNN* neighborhood computation only needs to be done once and can be shared between all layers, amortizing this overhead.

Figure 7.2: An example of four points with two hierarchy levels. The *kNN* neighborhood ($k = 2$) for each point is marked with colored edges.

7.3.2 GHA with An Example

The example case in Figure 7.2 illustrates the effect of GHA. Given a point cloud with four points, we use *kNN* neighborhood with $k = 2$, and farthest point sampling with a factor of two, resulting in two hierarchies in total. For this configuration, the GHA output for the point P_0 is given by

$$\begin{aligned}\tilde{Z}_0 &= \frac{1}{\tilde{D}_0}(\underbrace{w_0^{(0)}V_0 + w_1^{(0)}V_1}_{\text{from 0th hierarchy}} + \underbrace{w_0^{(1)}\frac{V_0+V_1}{2} + w_2^{(1)}\frac{V_2+V_3}{2}}_{\text{from 1st hierarchy}}) \\ &= \frac{1}{\tilde{D}_0}((w_0^{(0)} + \frac{w_0^{(1)}}{2})V_0 + (w_1^{(0)} + \frac{w_0^{(1)}}{2})V_1 + \frac{w_2^{(1)}}{2}V_2 + \frac{w_2^{(1)}}{2}V_3)\end{aligned} \tag{7.9}$$

with attention weights w and normalization factor $\tilde{D}_1$

$$\begin{aligned}\tilde{D}_0 &= w_0^{(0)} + w_1^{(0)} + w_0^{(1)} + w_2^{(1)} \\ w_0^{(0)} &= e^{\frac{Q_0 K_0^T}{\sqrt{d}}}, \quad w_1^{(0)} = e^{\frac{Q_0 K_1^T}{\sqrt{d}}} \\ w_0^{(1)} &= e^{\frac{(Q_0+Q_1)(K_0+K_1)^T}{4\sqrt{d}}} \\ w_2^{(1)} &= e^{\frac{(Q_0+Q_1)(K_2+K_3)^T}{4\sqrt{d}}}.\end{aligned} \tag{7.10}$$

The effect of GHA is clearly illustrated: neighboring points (P_0, P_1) are assigned with higher weights, while the interaction with far points (P_2, P_3) are only approximated with a single attention weight, reducing the overall memory footprint.

7.3.3 GHA Block

Following Vaswani *et al.* (2017), we combine GHA with a feedforward network (FFN) into a GHA layer and stack a total of L layers within a GHA block as shown in Figure 7.3. The input to a block consists of n tokens with a dimensionality of c, which we keep throughout the block, apart from the dimensionality c_f that is used between the two linear layers of the FFN. Similar

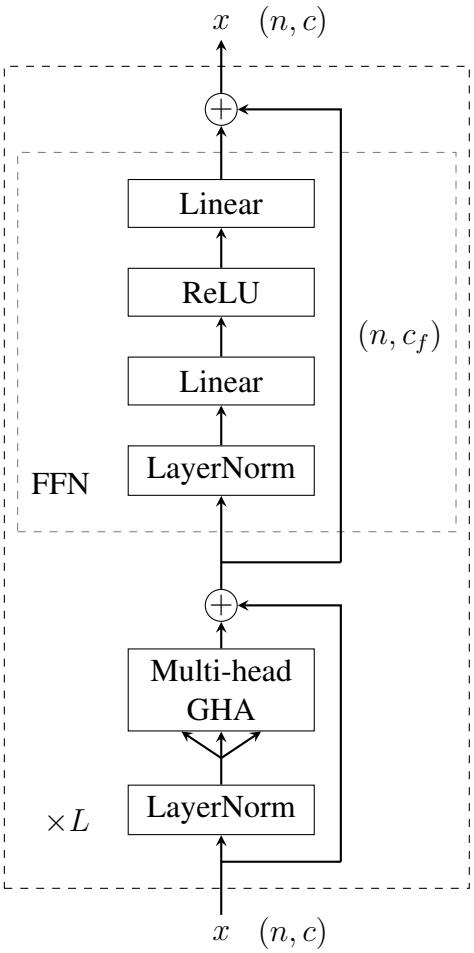


Figure 7.3: In our GHA block the attention and FFN are repeated L times. n is the number of points/voxels and c/c_f are the feature dimensionalities. Positional embeddings are omitted in the diagram.

to normal dot-product attention, GHA can also be extended to a multi-head variant which we use in all of our experiments. We apply dropout during training, with probability 0.1 in the attention layer, and 0.3 in the FFN layers.

In order to give the network spatial information, we compute random Fourier mappings (Tancik *et al.*, 2020)

$$\gamma(\mathbf{p}) = [\cos(2\pi \mathbf{b}_1^T \mathbf{p}), \sin(2\pi \mathbf{b}_1^T \mathbf{p}), \cdots, \cos(2\pi \mathbf{b}_m^T \mathbf{p}), \sin(2\pi \mathbf{b}_m^T \mathbf{p})]^T \tag{7.11}$$

and use them as positional embedding when computing attention. Here $\mathbf{p} \in \mathbb{R}^3$ and $\mathbf{b}_i \sim \mathcal{N}(0, 1)$ are sampled at initialization. Our method uses positional embeddings based on relative positions

$$\mathbf{y}_i = \sum_j \mathbf{q}_i^T (\mathbf{k}_j + \gamma(\mathbf{p}_i - \mathbf{p}_j)) \mathbf{v}_j \ , \tag{7.12}$$

but we also conduct ablation on using absolute positions

$$\mathbf{y}_i = \sum_j (\mathbf{q}_i + \gamma(\mathbf{p}_i))^T (\mathbf{k}_j + \gamma(\mathbf{p}_j))\mathbf{v}_j \ . \tag{7.13}$$

7.4 Evaluation

Our evaluation can be split into three main parts. We first evaluate Voxel-GHA for the task of 3D semantic segmentation and secondly for 3D object detection. In both cases we extend different sparse CNN based approaches with a GHA block and evaluate how this affects their performance. In the third part we evaluate how Point-GHA can serve as a replacement for vanilla global attention in the recent 3DETR object detection approach (Misra *et al.*, 2021).

7.4.1 Voxel-GHA: 3D Semantic Segmentation

Dataset. We conduct our experiment on the challenging ScanNet dataset (Dai *et al.*, 2017). It contains 3D reconstructions of indoor rooms collected using RGB-D scanners and points are labeled with 20 semantic classes. We follow the official training, validation, and test split of 1201, 312, and 100 scans.

Setting. We use two MinkUNet (Choy *et al.*, 2019a) architectures as baselines, the smaller ME-UNet18 and the bigger ME-UNet34, and test them on two voxel sizes, 2 cm and 5 cm. Both networks are based on ResNet encoders (18 and 34 layers, respectively) and a symmetric decoder. We insert $L = 6$ GHA layers ($c = 256, c_f = 128$, 8 heads) to the decoder branch after the convolution block at $2\times$ voxel stride (at $4\times$ voxel stride for 2 cm experiments, in order to roughly match the number of voxels in the attention layer). On average, this resulted in five to twelve thousand voxels per frame as input for the attention, a number for which computing multiple layers of full global attention is generally infeasible. We use a $3 \times 3 \times 3$ local window for the neighborhood in GHA, which we apply for all of our experiments.

We train all networks using the AdamW optimizer (Loshchilov and Hutter, 2019) and a one-cycle policy (Smith and Topin, 2019) for 600 epochs with batch size 8. For the baseline, we use a learning rate of 1e-2 with 0.01 weight decay, and for the GHA-augmented network, we use a learning rate of 1e-3 with 0.05 weight decay, which performed better in the respective settings. For augmentation, we use random rotation, scaling, and random cropping (Zhang *et al.*, 2021).

Main results. In Table 7.1 we present the mean intersection-over-union (mIoU) between the predicted and labeled masks on the ScanNet validation set. In addition to the official annotation, we also report an evaluation done using the cleaned validation labels from (Ye *et al.*, 2021). Compared to the CNN only baseline, GHA gives a clear improvement across all network models and voxel sizes. The performance gap is especially visible for the coarser 5 cm voxels, where GHA improves the baseline by 1.7% mIoU. At finer voxel sizes, however, we see that the improvement of GHA is small, only 0.2% mIoU, for the weaker ME-UNet18

Table 7.1: 3D semantic segmentation performance (mIoU) on ScanNet validation set using original or cleaned labels from Ye *et al.* (2021).

	Original		Clean	
Method	5 cm	2 cm	5 cm	2 cm
ME-UNet18	67.8	72.6	68.4	73.2
+ local attention	69.0	**73.3**	69.6	**73.8**
+ local attention ($L = 8$)	68.8	–	69.3	–
+ GHA	**69.6**	72.8	**70.2**	73.4
ME-UNet34	68.3	72.9	69.0	73.7
+ local attention	68.6	**73.6**	69.0	**74.6**
+ local attention ($L = 8$)	69.1	–	69.7	–
+ GHA	**70.0**	73.5	**70.3**	74.1

backbone. We speculate that, at this fine scale, a smaller backbone may not be strong enough to extract rich features that can support the global fitting of GHA. The results using cleaned validation labels are in general higher, but the trend matches that observed using original labels, suggesting that the networks do not have special behavior with respect to labeling noise.

In addition, we conduct ablation studies by replacing GHA with local attention. Since local attention consumes less memory than GHA, to have a fair comparison, we also experiment with a bigger model composed of eight layers, which surpasses memory consumption of GHA. With 5 cm voxels, using the same memory, GHA outperforms local attention by 0.8% mIoU. This shows the benefit of having a global field of view. However, with 2 cm voxels, local attention outperforms GHA. This may suggest that, depending on the resolution and the capacity of the feature extractor, limiting field of view can be beneficial to the overall performance, and prompts further architecture level investigation.

Table 7.2 shows the per-class performance of the 2 cm experiments, along with several state-of-the-art methods. Compared to the baseline, GHA performs particularly well on the "fridge" class, which has very few training samples, but under-performs on the "shower" class. This behavior may be caused by the global field of view of GHA, which brought in supportive (in the "fridge" case) or distracting (in the "shower" case) long range information. Figure 7.4 shows some qualitative results. The predictions of GHA are significantly better on the regions where the semantic labels are ambiguous based on local geometry and color information alone, showing the advantage of including non-local information for inference.

Table 7.2: Per-class semantic segmentation score on ScanNet validation set. †: numbers taken from Nekrasov *et al.* (2021).

Method	mIoU	wall	floor	cabinet	bed	chair	sofa	table	door	wndw	booksh.
KPConv† (Thomas *et al.*, 2019)	69.3	82.4	94.4	64.5	79.2	88.5	77.2	73.0	60.5	59.1	**79.8**
MinkNet† (Choy *et al.*, 2019a)	72.4	**85.6**	**96.5**	**64.7**	**82.1**	**91.0**	**84.4**	**74.5**	**65.0**	**62.8**	79.5
BPNet (Hu *et al.*, 2021a)	**73.9**	–	–	–	–	–	–	–	–	–	–
VMNet (Hu *et al.*, 2021b)	73.3	–	–	–	–	–	–	–	–	–	–
ME-UNet18	72.6	85.1	95.2	65.2	79.0	**91.7**	84.1	73.8	63.2	64.6	**80.4**
+ local	**73.3**	85.6	**95.4**	66.7	79.6	91.2	**83.7**	74.8	**66.1**	**65.4**	80.2
+ GHA	72.8	**85.9**	95.1	**69.4**	**79.9**	91.0	83.6	75.3	63.6	65.3	**80.4**
ME-UNet34	72.9	85.1	95.2	65.6	78.6	91.5	84.1	73.9	62.2	64.7	79.4
+ local	**73.6**	**86.1**	**95.4**	66.4	79.1	**91.7**	83.7	74.3	**64.5**	**64.9**	80.3
+ GHA	73.5	85.7	95.3	**67.0**	**79.9**	91.0	**85.1**	**77.5**	61.7	**64.9**	**81.8**

Method	mIoU	pic	counter	desk	curtain	fridge	shower	toilet	sink	bathtub	otherf.
KPConv† (Thomas *et al.*, 2019)	69.3	28.4	59.9	**63.7**	71.6	**53.1**	54.1	91.5	63.3	86.0	**56.4**
MinkNet† (Choy *et al.*, 2019a)	72.4	**32.4**	**64.4**	**63.7**	**75.5**	51.6	**69.0**	**93.0**	**67.6**	**87.6**	56.3
BPNet (Hu *et al.*, 2021a)	**73.9**	–	–	–	–	–	–	–	–	–	–
VMNet (Hu *et al.*, 2021b)	73.3	–	–	–	–	–	–	–	–	–	–
ME-UNet18	72.6	32.2	62.1	63.9	75.2	55.9	**70.8**	92.8	67.3	86.4	62.3
+ local	**73.3**	**33.5**	**65.8**	66.3	**77.0**	55.4	69.3	**93.6**	**66.9**	86.1	**62.8**
+ GHA	72.8	30.8	61.1	**67.6**	76.0	**62.0**	60.6	92.6	**66.9**	**87.0**	61.1
ME-UNet34	72.9	33.0	**64.9**	65.1	76.3	60.0	71.7	93.3	65.7	86.4	61.1
+ local	**73.6**	**33.2**	63.6	64.2	**77.4**	62.6	**73.8**	**93.7**	68.3	86.1	**63.7**
+ GHA	73.5	31.0	63.7	**68.9**	76.0	**65.3**	63.2	93.0	**68.7**	**87.6**	61.9

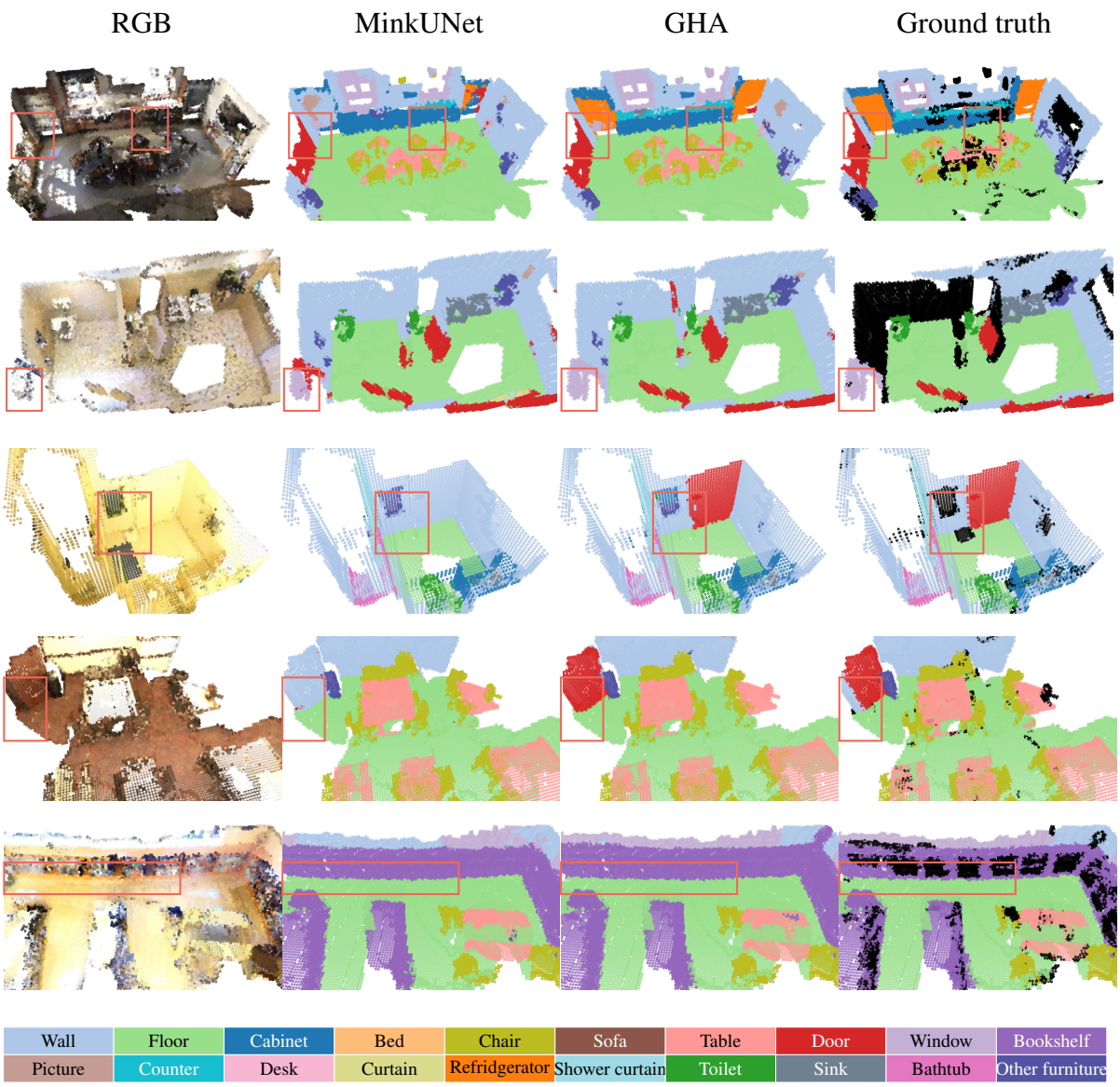


Figure 7.4: Qualitative semantic segmentation results on the ScanNet validation set. Predictions with GHA show clear improvement in the marked regions, where semantic labels are ambiguous judging from local geometry and color information alone. This improvement highlights the benefit of long-range information.

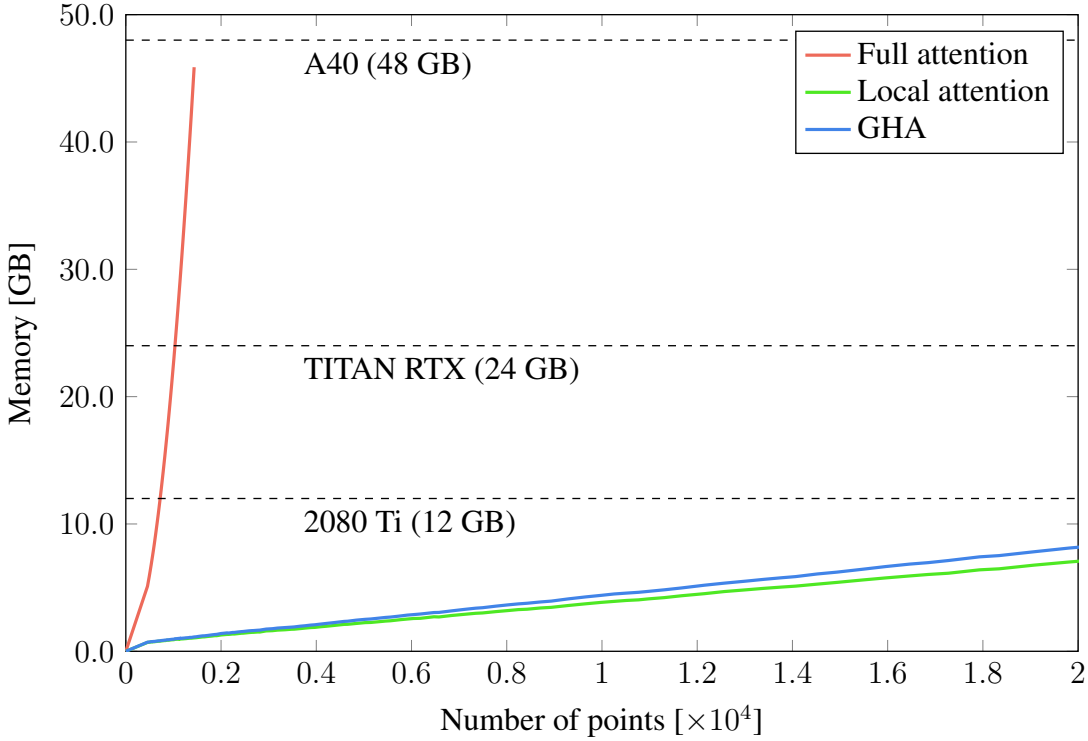


Figure 7.5: Training memory measured for 6 attention layers.

Figure 7.5 shows the memory consumption of one forward and backward pass for different attention types. GHA has linear space complexity, similar to that of local attention, while having a global field of view. The memory consumption of full attention is prohibitively expensive. Training the ME-18 variants (5 cm) takes 34 hours with GHA and 27.5 hours with local attention, both using a single A40 GPU. For inference on ScanNet, GHA averages 5.6 and 3.7 FPS for 5 and 2 cm voxels, whereas local attention has 12.9 and 6.5 FPS. The inference speed of GHA can be further improved with a cuda-optimized indexing implementation and in-place aggregation.

Figure 7.6 shows the distribution of attention at different distances. While local attention only attends within a local neighborhood, GHA has a global receptive field, and can attend to all regions in the scene. In addition, GHA has the inductive bias to place an emphasis on nearby points, leading to the skewed pattern in the attention histogram. Figure 7.7 shows the attention of a randomly initialized GHA layer as a heatmap. Its inductive bias can clearly be seen, as close points tend to receive higher weights, even without being trained.

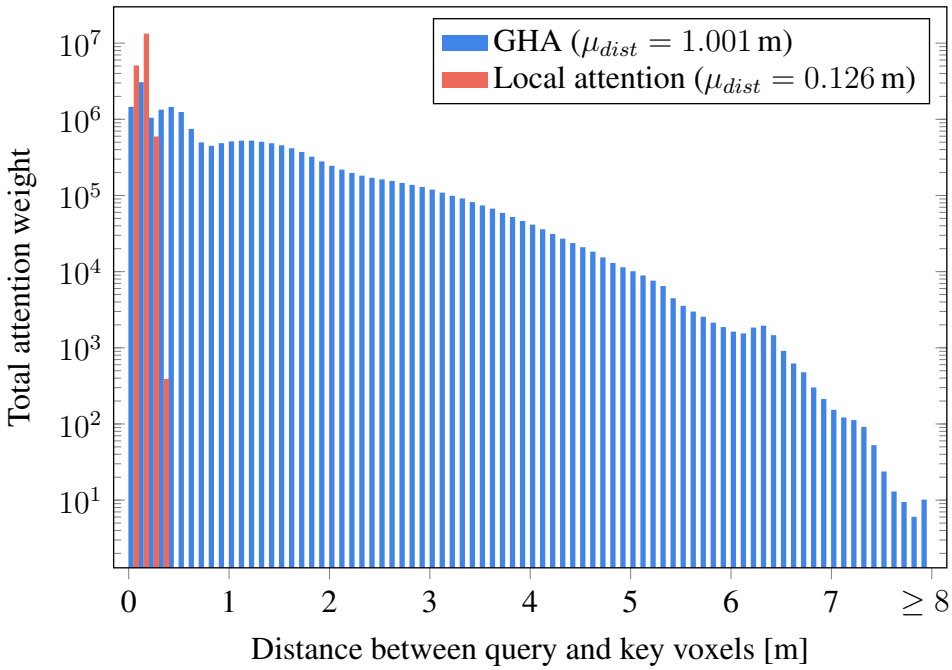


Figure 7.6: The distribution of attention distances for both the GHA and local attention shown in logarithmic scale. Local attention is incapable of attending to voxels beyond a fixed threshold, whereas GHA can and does indeed attend globally.

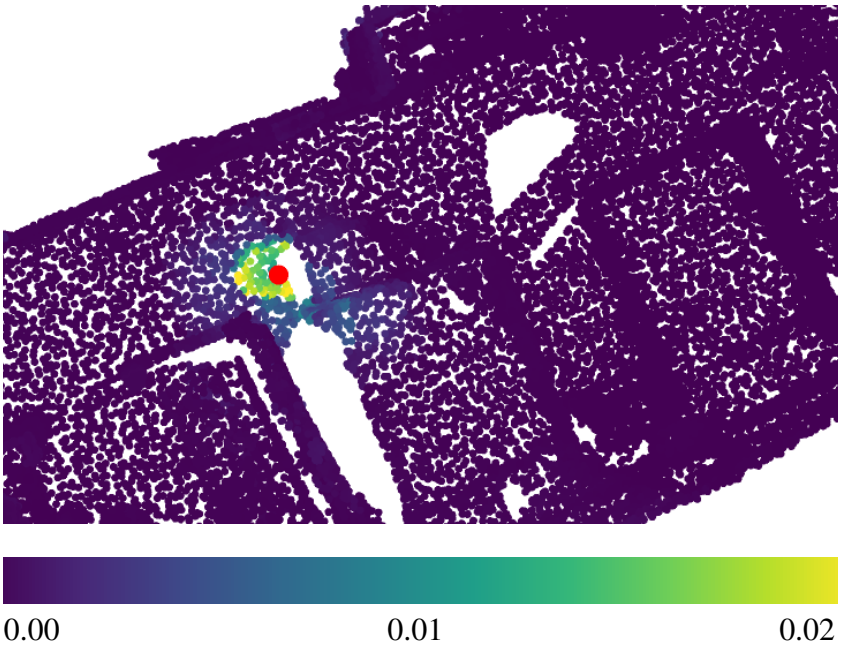


Figure 7.7: The attention heatmap of a **randomly initialized** GHA layer (source point marked with •). Closer points tend to have higher weights, showing the locality bias of GHA.

Table 7.3: ScanNet validation set performance with different number of GHA layers.

# Layers	mIoU
2	68.3
4	68.6
6	70.0
8	69.5

Table 7.4: ScanNet validation set performance with different positional embeddings.

Embedding	Relative	mIoU
None		69.0
Learnable	✗	69.3
Fourier	✗	69.4
Fourier	✓	70.0

Table 7.5: ScanNet validation set performance with different attention mechanism.

Attention type	mIoU
Cosine (Park *et al.*, 2022)	68.8
Vector (Zhao *et al.*, 2021)	71.2
Dot-product (Vaswani *et al.*, 2017)	70.0

Additional experiments. We conduct experiments evaluating several design choices. These experiments are done on ScanNet using the ME-UNet34 model with 5 cm voxels.

Number of layers. Experiments on different numbers of layers are reported in Table 7.3. The result shows that, with sufficient network depth, GHA outperforms the baseline or the local attention variants. We attribute this superior performance to its global receptive field, which channels information beyond a local neighborhood.

Positional embedding. We experiment with different types of positional embedding, including the learned embedding via a two-layer MLP, and report the results in Table 7.4. Results show that random Fourier mapping with relative positions gives the best performance while requiring no additional parameters.

Attention type. In addition to the standard scaled dot-product attention (Vaswani *et al.*, 2017), cosine-similarity-based attention (Park *et al.*, 2022) and full vector attention (Zhao *et al.*, 2020) have also been used in transformer architectures for point clouds. As shown in Table 7.5, the full vector attention gives a 1.2% mIoU performance increase, at the cost of increased memory requirements and parameter counts (training batch size needs to be halved). We use the regular dot-product attention to stay consistent with most existing research.

Local window vs. kNN neighbors. Our method uses $3 \times 3 \times 3$ local windows as the neighborhood $\mathcal{T}$ in Equation 7.5. As an alternative, we can use a *kNN*-neighborhood ($k = 27$), resulting in on-par performance (69.9% mIoU). We opt for the faster local window-based neighborhood, which in addition has the advantage of being robust against varying point densities.

Table 7.6: 3D object detection performance (mAP) on the KITTI validation set for (E)asy, (M)oderate, and (H)ard objects. †: numbers taken from MMDetection3D (2020).

Method	Car			Cyclist			Pedestrian		
	E	M	H	E	M	H	E	M	H
VoxelNet (Zhou and Tuzel, 2017)	82.0	65.5	62.9	67.2	47.7	45.1	57.9	53.4	48.9
SECOND (Yan *et al.*, 2018)	87.4	76.5	69.1	–	–	–	–	–	–
F-PointNet (Qi *et al.*, 2018)	83.8	70.9	63.7	–	–	–	–	–	–
PointRCNN (Shi *et al.*, 2019)	89.1	78.7	78.2	**93.5**	**74.2**	**70.7**	65.8	59.6	52.8
MVF† (Zhou *et al.*, 2019b)	88.6	77.7	75.1	80.9	61.9	59.5	61.3	55.7	50.9
Part-A^2 (Shi *et al.*, 2020a)	89.5	79.5	78.5	88.3	73.1	70.2	**70.4**	**63.9**	**57.5**
PV-RCNN (Shi *et al.*, 2020b)	**92.6**	**84.8**	**82.7**	–	–	–	–	–	–
3DSSD (Yang *et al.*, 2020)	89.7	79.5	78.7	–	–	–	–	–	–
SECOND	88.6	77.6	**74.8**	80.4	66.7	63.1	61.5	55.5	50.1
+ LA	**89.0**	**77.8**	74.5	76.8	61.7	59.7	60.2	53.2	49.2
+ LA ($L = 6$)	88.0	77.5	74.4	77.7	64.2	60.1	63.5	55.8	51.2
+ GHA	88.8	**77.8**	74.6	**81.7**	**67.8**	**63.9**	**64.0**	**56.1**	**51.2**

7.4.2 Voxel-GHA: 3D Object Detection

Dataset. We conduct our experiments using the KITTI (Geiger *et al.*, 2013) and the nuScenes dataset (Caesar *et al.*, 2020), both are collected using cars equipped with LiDAR sensors driving on the city streets. The KITTI dataset contains 7,481 LiDAR scans for training and 7,518 for testing. We follow prior work by Zhou and Tuzel (2017) and divide the training split into 3,712 samples for training and 3,769 for validation. The nuScenes dataset is bigger and more challenging, containing 1,000 sequences (380,000 scans), divided into 700/150/150 sequences for training, validation, and testing. The KITTI dataset is annotated with bounding boxes for three classes, whereas the nuScenes dataset is annotated with ten classes.

Setting. We use implementations from MMDetection3D (2020) library for the next set of our experiments. For the KITTI dataset, we experiment with augmenting the SECOND detector (Yan *et al.*, 2018) with GHA. The SECOND detector uses a sparse voxel-based backbone to encode point cloud features, which are then flattened into a dense 2D feature map and passed into a standard detection head for bounding box regression. We add $L = 4$ GHA layers ($c = 64, c_f = 128$, 4 heads) at the end of the sparse voxel encoder and leave the rest of the network unchanged (the dimension of GHA layers is chosen to match the encoder output). We use the original setting from MMDetection3D (2020), namely, we train the network for 40 epochs, with batch size 6, AdamW optimizer (Loshchilov and Hutter, 2019), 0.018 learning rate, and a one cycle learning rate policy (Smith and Topin, 2019).

For the nuScenes dataset, we experiment with the state-of-the-art CenterPoint detector (Yin *et al.*, 2021a) which combines a VoxelNet (Zhou and Tuzel, 2017) backbone and a center-

Table 7.7: 3D object detection performance on nuScenes validation set.

Method	mAP	NDS
PointPillars (Lang *et al.*, 2019)	28.2	46.8
3DSSD (Yang *et al.*, 2020)	42.6	56.4
CenterPoint (Yin *et al.*, 2021a)	**56.4**	**64.8**
SASA (Chen *et al.*, 2022a)	45.0	61.0
CenterPoint (10 cm)	56.2	64.4
+ LA	**56.8**	**64.8**
+ LA ($L = 6$)	**56.8**	64.7
+ GHA	56.7	64.7
CenterPoint (7.5 cm)	57.3	65.2
+ LA	57.7	65.6
+ LA ($L = 6$)	OOM	
+ GHA	**57.9**	**65.7**

based object detection head. Similar to the KITTI experiment, we add $L = 4$ GHA layers ($c = 128, c_f = 256$, 8 heads) at the end of the backbone. We experiment with two voxel sizes, 10 cm and 7.5 cm. To reduce the training time, we load a pre-trained model from MMDetection3D (2020), and fine-tune the network end-to-end for three epochs. The pre-trained model is trained for 20 epochs. To fine-tune the CenterPoint variants, we use a batch size of 8 for 10 cm voxels, and 6 for 7.5 cm voxels, the AdamW optimizer (Loshchilov and Hutter, 2019), and a learning rate of 1e-5 with a one-cycle policy (Smith and Topin, 2019) and 0.05 weight decay.

Results. We report in Table 7.6 the performance of the retrained SECOND detector with and without GHA, and in Table 7.7 we report detection mAP and NDS (nuScenes detection score) on nuScene dataset. In all experiment cases, GHA matches or outperforms the baseline results. On the KITTI dataset, the benefit is especially visible on the smaller cyclist and pedestrian classes, increasing 1.1% and 0.6% mAP respectively in the moderate setting. On the nuScenes dataset, fine-tuning with GHA for only 3 epochs consistently brings a 0.5% mAP and 0.3% NDS improvement for both voxel sizes. The qualitative results on KITTI and nuScenes dataset are shown in Figure 7.8 and 7.9 respectively. These results show good detection performance even for small object classes.

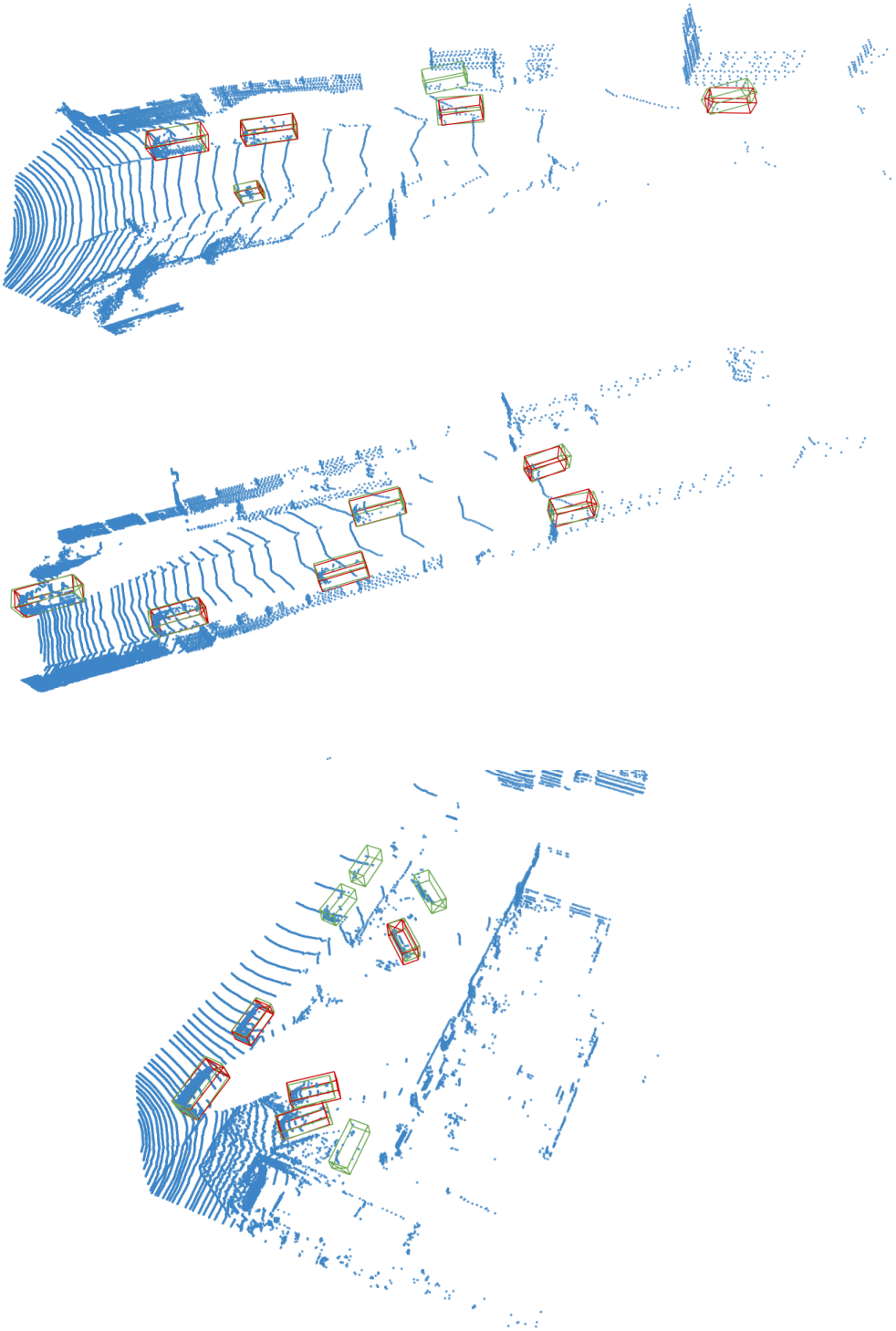


Figure 7.8: Qualitative detection results of SECOND detector (Yan *et al.*, 2018) with GHA on the KITTI (Geiger *et al.*, 2012) validation set. Predictions and groundtruth are shown as green and red bounding boxes.

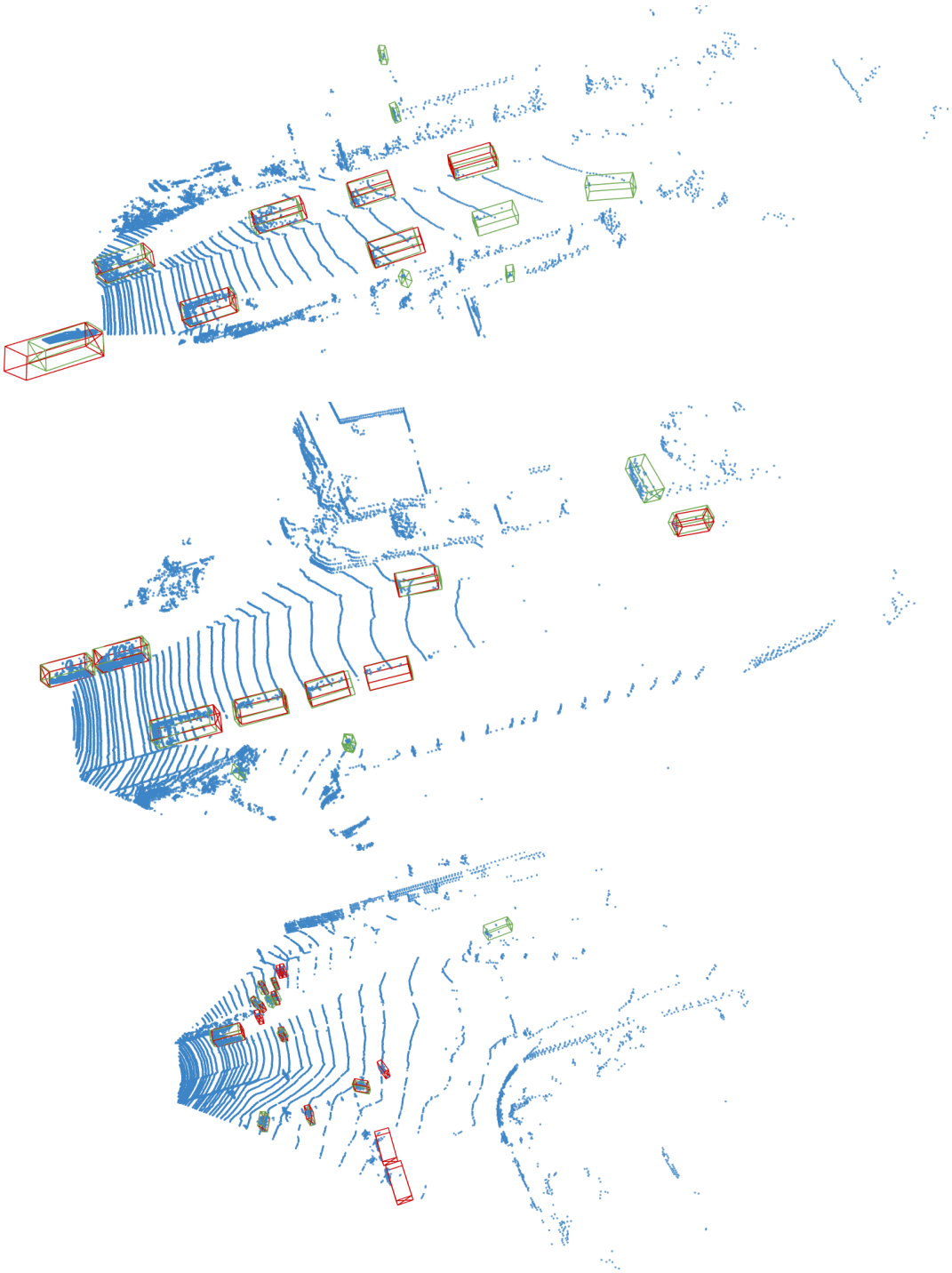


Figure 7.8: Qualitative detection results of SECOND detector (Yan *et al.*, 2018) with GHA on the KITTI (Geiger *et al.*, 2012) validation set. Predictions and groundtruth are shown as green and red bounding boxes.

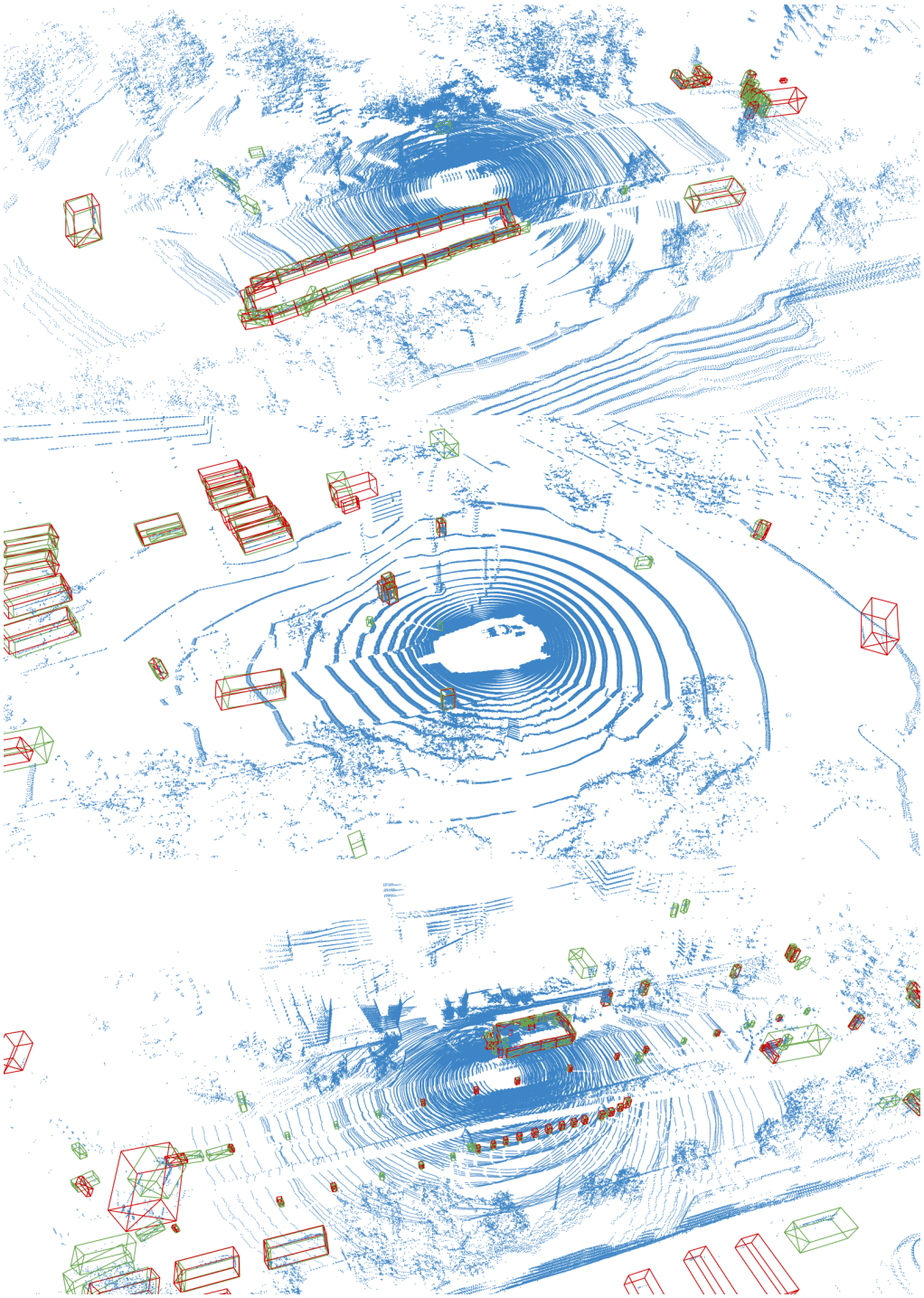


Figure 7.9: Qualitative detection results of CenterPoint (Yin *et al.*, 2021a) with GHA (7.5 cm voxels) on the nuScenes (Caesar *et al.*, 2020) validation set. Predictions and groundtruth are shown as green and red bounding boxes. Per nuScenes convention, each LiDAR scan is an accumulation of five consecutive sweeps.

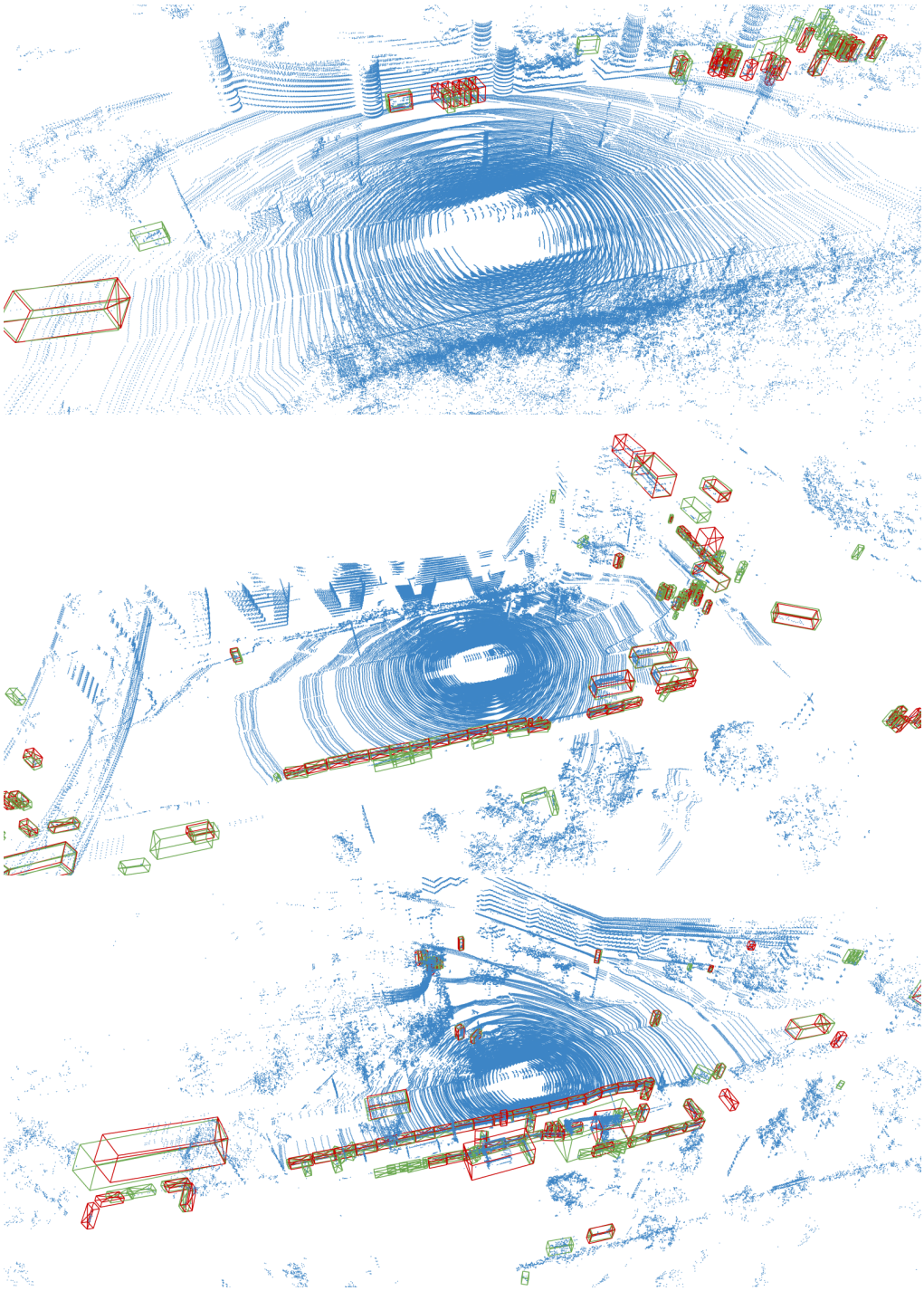


Figure 7.9: Qualitative detection results of CenterPoint (Yin *et al.*, 2021a) with GHA (7.5 cm voxels) on the nuScenes (Caesar *et al.*, 2020) validation set. Predictions and groundtruth are shown as green and red bounding boxes. Per nuScenes convention, each LiDAR scan is an accumulation of five consecutive sweeps.

7.4.3 Point-GHA: 3D Object Detection

Dataset. To evaluate point-based GHA for object detection, we experiment on ScanNet detection dataset (Dai *et al.*, 2017), which contains 1,513 indoor scans with annotated bounding boxes, split into 1,201 scenes for training and 312 for validation. Following prior works (Qi *et al.*, 2019; Cheng *et al.*, 2021; Zhang *et al.*, 2020b; Xie *et al.*, 2021), we report the mean average precision, specifically mAP_{25} and mAP_{50}, computed at a 0.25 and 0.5 IoU threshold respectively.

Setting. We experiment with the 3DETR network proposed by Misra *et al.* (2021). 3DETR follows a transformer encoder-decoder architecture. From the input point cloud, 3DETR first samples 2048 points and extracts their neighboring features via a single set aggregation operation, as introduced in PointNet++ (Qi *et al.*, 2017a). These points are then processed with a transformer encoder composed of multiple self-attention layers. During the decoding stage, 3DETR samples a set of query points and projects them into the embedding space (*object queries*), and through a stack of self and cross-attention, decodes each of the object queries into bounding boxes.

To evaluate Point-GHA, we replace 3DETR's encoder self-attention with Point-GHA using $k = 6$ for the *kNN* neighborhood, while leaving all other network components unchanged. We refer to this network as 3DETR-GHA. In addition, Misra *et al.* (2021) experimented with masking of attention between points further than a threshold, and obtained improved results. This version, called 3DETR-m, also includes an intermediate pooling layer. Since GHA inherently has the inductive bias to focus on local neighborhoods, we do not need to apply distance-based masking. Rather, we experiment with adding the intermediate pooling layer, and refer to this version as 3DETR-GHA-p. Finally, with the benefit of the reduced memory consumption of GHA, we experiment with a bigger model by including more points (4096) into the encoder.

Following Misra *et al.* (2021), we train our network for 1080 epochs, with the AdamW optimizer (Loshchilov and Hutter, 2019), a cosine learning rate schedule, with a final learning rate of 10^{-6}, weight decay of 0.1, and $L2$-norm gradient clipping at 0.1.

Results. In Table 7.8 we present the main experiment result. Compared to the 3DETR baseline, our GHA-3DETR obtains an improvement of 1.6% on mAP_{25} and 8.1% on mAP_{50}. Compared to the masking 3DETR-m, our GHA-3DETR-p achieves similar performance. When including more points, GHA-3DETR-p gives an mAP_{25} and mAP_{50} improvement of 2.1% and 1.5% respectively, coming closer to state-of-the-art performance.

In Table 7.9 we present an ablation study, where we replace the self-attention in 3DETR with different attention mechanisms. Compared to the regular attention used in 3DETR, local attention achieved improved results, especially for mAP_{50}, despite not having a global field of view. This demonstrates the value of focusing on local neighbors. GHA outperforms local attention, thanks to the additional access to global information acquired from its hierarchical design. We additionally implement a variant of BigBird (Zaheer *et al.*, 2020) to work with

Table 7.8: 3D object detection performance on the ScanNet validation set.

Method	mAP_{25}	mAP_{50}
VoteNet (Qi *et al.*, 2019)	60.4	37.5
MLCVNet (Qian *et al.*, 2020)	64.7	42.1
BRNet (Cheng *et al.*, 2021)	66.1	50.9
H3DNet (Zhang *et al.*, 2020b)	67.2	48.1
VENet (Xie *et al.*, 2021)	67.7	–
GroupFree3D (Liu *et al.*, 2021b)	**69.1**	**52.8**
3DETR (Misra *et al.*, 2021)	62.7	37.5
3DETR-m (Misra *et al.*, 2021)	**65.0**	**47.0**
GHA-3DETR	64.3	45.6
GHA-3DETR-p	65.4	46.1
GHA-3DETR-p (4096)	**67.1**	**48.5**

Table 7.9: 3DETR performance with different encoder self-attention.

Attention	mAP_{25}	mAP_{50}
Regular	62.7	37.5
Local	63.1	43.9
BigBird (Zaheer *et al.*, 2020)	63.4	43.4
GHA	**64.3**	**45.6**

Table 7.10: 3DETR performance with different *kNN* neighborhood sizes.

kNN size	mAP_{25}	mAP_{50}
$k = 2$	64.1	42.9
$k = 4$	64.1	43.6
$k = 6$	**64.3**	**45.6**
$k = 8$	63.4	43.2
$k = 10$	62.6	44.3
$k = 20$	61.7	41.1

point clouds (it was originally proposed for language tasks), which augments local attention with several tokens that have global connectivity (*global attention*), and randomly activate a small number of non-local attentions (*random sparse attention*). GHA outperforms our BigBird implementation. Even though both methods have global connectivity and have linear memory complexity, GHA additionally embodies the inductive bias to focus on local information, to which we attribute its better performance. In Table 7.10 we also present a study on the effect of the *kNN* neighborhood size, and the result shows that GHA is robust to a wide range k, consistently outperforming its counterpart using regular attention.

7.5 Discussion

In this chapter we propose Global Hierarchical Attention (GHA) for point cloud analysis. Its inductive bias towards nearby points, its global field of view, as well as its linear space complexity, make it a promising mechanism for processing large point clouds. Extensive experiments on both detection and segmentation show that adding GHA to an existing network give consistent improvements over the baseline network. Moreover, ablation studies replacing GHA with other types of attention have demonstrated its advantage. The experiments in this chapter use GHA as an add-on component to an existing network, in order to limit the number of design choices involved in designing a network from scratch. The overall positive results prompt to further exploration on new architectures built with GHA as the main operation.

The design of GHA is motivated by the need of incorporating global information. Indeed, fast information exchange between long-range tokens is a hallmark of transformer networks (Vaswani *et al.*, 2017). However, the role of global information in processing point cloud remains unclear and should be further investigated. On the one hand, the work by Lai *et al.* (2022) and the qualitative results in Figure 7.4 seem to suggest the benefit of incorporating global information, but the evidence is short from being conclusive. On the other hand, local attention, despite lacking a global field of view, does demonstrate good results, as shown by the experiments in this chapter as well as several existing researches (Zhao *et al.*, 2021; Park *et al.*, 2022; Wu *et al.*, 2022). Furthermore, existing approaches focus on indoor dataset like ScanNet (Dai *et al.*, 2017) and S3DIS (Armeni *et al.*, 2016) where the point clouds are collected by RGB-D scanners. The benefit of a transformer architecture for processing LiDAR data should be further examined on dataset like SemanticKITTI (Behley *et al.*, 2019) and nuScenes (Caesar *et al.*, 2020).

8 Conclusion

In this thesis, we have introduced several novel approaches advancing the state of the art in LiDAR-based person detection. Our focus has been on mobile robotic applications, where robots operate in close proximity to surrounding humans. To bring the thesis to a conclusion, we will now summarize the contributions presented in the technical chapters and discuss exciting open challenges that pave the way for future research directions.

8.1 Summary and Contributions

After preliminaries and related work, Chapter 4 introduced DR-SPAAM, a person detector based on range data collected using 2D LiDAR sensors. It takes the distance-robust architecture from the DROW (Beyer *et al.*, 2018) detector and augments it with a powerful spatial attention and auto-regressive temporal integration model. The spatial attention mechanism associates features from different frames, even in the presence of misalignment, based on a learned similarity, whereas the auto-regressive model aggregates temporal information progressively through time. The temporal integration model effectively alleviates the challenge caused by sparse LiDAR scans from a single frame, thus producing more accurate detections, with little added computation. Experiments show that the spatial attention and temporal integration model is robust against changes in LiDAR sampling rate or point density, thanks to the learned similarity, allowing the trained detector to be deployed on LiDAR sensors with different specifications. In addition, DR-SPAAM has a high inference rate, reaching 87.2 FPS on a laptop with a dedicated GPU and 22.6 FPS on an NVIDIA Jetson AGX embedded GPU, and is well-suited for real-time applications on mobile platforms.

Only a few public datasets exist for 2D LiDAR sensors with annotated persons. Despite the architecture design of DR-SPAAM, which facilitates generalization to different sensor specifications, the limited diversity in the training data may still negatively impact the detector performance on new scenarios. To improve generalizability during deployment, in Chapter 5 we propose a method to automatically generate pseudo training labels for a 2D-LiDAR-based

detector, using a calibrated camera and an off-the-shelf image-based detector. Our experiments show that fine-tuning with pseudo-labels during deployment can boost the detector performance by up to +8.3% AP. The generated pseudo-labels inevitably contain noise. Using a robust training scheme, such as mixup regularization (Zhang *et al.*, 2018) or partially Huberized cross-entropy loss (Menon *et al.*, 2020), can further increase the detector performance by up to +2.9% AP. The development in this chapter opens up the way towards continual learning, in which an autonomous agent continues to improve its perceptual capabilities during deployment with newly collected training samples. In such applications, it is important to plan the collection and training schedule. As highlighted in our preliminary experiment, the training batches need to contain samples from sufficiently diverse scenarios, rather than from one specific scenario over a short time span, to avoid undesired overfitting.

2D LiDAR sensors are widely used on robotic platforms for mapping and navigation purposes, thanks to their affordable price, large field of view, and accurate range measurement. However, the measurement from a 2D LiDAR sensor is restricted to only a single plane. An alternative, the 3D LiDAR sensor, can provide scans to a 3D volume using multiple laser beams, but carries a higher price tag. In Chapter 6 we conducted experiments to analyze the performance gap between 2D and 3D LiDAR sensors for person detection. Our experiment shows that using a 2D or 3D LiDAR sensor gives equally accurate detection when persons are visible to the sensor. However, persons are more prone to being fully occluded and invisible from the 2D LiDAR sensor due to its constrained scanning angles, thus making detection impossible for the network. In addition, the detections from 3D LiDAR sensors are better localized. The 2D-LiDAR-based detector has a higher inference speed, reaching 37.1 FPS on a laptop with an NVIDIA RTX 2080 graphics card, compared to 19.8 FPS from the 3D-LiDAR-based detector. While 3D LiDAR sensors offer a generally more robust solution for person detection, 2D LiDAR sensors strike a favorable balance between performance and runtime, making them well-suited for mobile robots with limited onboard computational resources.

The attention mechanism (Vaswani *et al.*, 2017), being equivariant towards permutation and effective at channeling long-range information, is a promising candidate for processing point clouds. However, the quadratic space complexity with respect to the number of tokens forbids its application to new LiDAR models with several hundreds of thousands of points per scan. In Chapter 7 we proposed an alternative to regular global attention, which has linear space complexity while maintaining the global field of view. The approach, termed Global Hierarchical Attention, computes local attention on multiple resolution hierarchies and aggregates the results across all hierarchies. We augment existing detection and segmentation networks with GHA and compare their performance with the baseline. On ScanNet (Dai *et al.*, 2017) semantic segmentation, GHA gives a +1.7% mIoU increase over the MinkowskiEngine (Choy *et al.*, 2019a). For object detection, GHA gives a +0.5% mAP increase over CenterPoint (Yin *et al.*, 2021a) on the nuScenes dataset (Caesar *et al.*, 2020), and a +0.2% mAP increase over SECOND (Yan *et al.*, 2018) on the KITTI dataset (Geiger *et al.*, 2013). In the ablation study using 3DETR (Misra *et al.*, 2021) as the baseline, GHA outperforms approaches using linear attention, demonstrating the benefit of keeping a global field of view.

8.2 Perspectives

The contributions in this thesis open up several intriguing research perspectives, which we summarize as follows:

Extension to 3D Point Cloud. Several components of the 2D-LiDAR-based detectors presented in this thesis can be adapted to work with 3D point clouds. The distance-robust preprocessing used in the DROW detector Beyer *et al.* (2016) provides a strong prior to the detector, leveraging the known geometry of the object class. It is especially effective since, unlike in images, the apparent shape of an object does not vary in 3D LiDAR scans based on its location, and the intra-class size variation for many classes of objects is small (*e.g.* cars). In addition, the spatial-attention and auto-regressive model introduced in DR-SPAAM can be extended to work with 3D point clouds by using a 2D neighborhood window on the range image. It provides a powerful learning-based feature association approach and removes the need to use odometry to align consecutive frames. Last but not least, the pseudo-label generation method in Chapter 5 naturally extends to 3D LiDAR points. The proposed range-based heuristic can be used to find foreground points within an image box corresponding to the object. An off-the-shelf image-based segmentation network (He *et al.*, 2017) can be used to further refine the pseudo-labels (Vora *et al.*, 2019). However, determining the size and center of the object is challenging, since only a part of the object may be observed and the true geometry of the object cannot be determined by naively fitting a bounding box (Shi *et al.*, 2020a).

Multi-Sensor Fusion. Methods presented in Chapter 5 and Chapter 6 leverage the multi-sensor (cameras, 2D and 3D LiDAR sensors) setup on the robots. In practice, mobile robotic platforms are often equipped with multiple sensors, and fusing information from multiple sensors can potentially provide the best perception performance. Preliminary experiments in Chapter 6 point out the benefit of ensembling the detections from 2D and 3D LiDAR sensors, since many persons are picked up by one sensor alone. Similar late fusion is used by Linder *et al.* (2016), where detections made independently from each sensor are stitched together and passed to downstream tasks. Such late fusion is easy to implement, since detections from each sensor can be carried out in a distributed fashion, and the overall system can still function if a failure happens to a single sensor. The early fusion strategy, on the contrary, first fuses measurements from all sensors and then performs detections on fused measurements (Liu *et al.*, 2023a). This setup allows the network to jointly explore information presented in all sensors and may result in a better-performing detector. Attention-based mechanisms (Vaswani *et al.*, 2017), like the one used in DR-SPAAM, are well-suited to fuse features from different sensor modalities. However, detectors trained with an early fusion strategy may be hard to transfer to platforms with a different sensor setup, and a single failure of a sensor may break the entire system.

Multi-Modality Contrastive Learning. The pseudo-label generation in Chapter 5 guides the training of a LiDAR-based detector with detections from an image-based detector. In prin-

ciple, this guidance can be extended to be bi-directional, where the detectors from both modalities learn to produce consensual results. A similar cross-modality supervision strategy is utilized by Ding *et al.* (2023) to learn scene flows using LiDAR and radar in a self-supervised fashion. Designing the supervision signal, however, is a challenging task, especially for detection tasks where the final results are bounding boxes, not masks or classification logits. The Hungarian matching used in DETR (Carion *et al.*, 2020) might be a valid approach but would require further investigation. Augmentations of a single sample may be utilized to provide additional supervision signals, as shown in image-based contrastive learning (Chen *et al.*, 2020d,e; He *et al.*, 2020b; Chen *et al.*, 2020f, 2021).

Uncertainty Estimation. It is known that neural networks, including the detectors presented in this thesis, are prone to making false predictions with high confidence, especially when the testing sample is out of the distribution of the training data. Such false predictions can potentially lead to catastrophic consequences (National Highway Traffic Safety Administration, 2016). Uncertainty estimation is an important topic in computer vision (Kendall and Gal, 2017; Loquercio *et al.*, 2020). For integration into autonomous systems, especially for safety-critical applications, the detectors presented in this thesis should be extended to output, for each detection, an associated uncertainty score, which can be leveraged for making downstream decisions.

Continual Learning. The pseudo-label generation in Chapter 5 hints towards a scenario where the robotic agent continuously improves its perception capabilities using new examples encountered during deployment. However, when and how to execute training updates is a topic that requires further investigation. A careless execution may lead to the phenomenon known as catastrophic forgetting, where the network drastically forgets the previously learned task while learning a new one. The latest advances in continual learning may help realize such a system (De Lange *et al.*, 2022; Wang *et al.*, 2023c). In addition, active learning (Schmidt *et al.*, 2020; Takezoe *et al.*, 2023) can be used to select samples relevant to improving the detectors, increasing the efficiency of the learning process.

Multi-Task Network. The methods presented in this thesis focus specifically on person detection. There are other human-related perception tasks that would benefit robotic applications, such as instance segmentation (Zhang *et al.*, 2020a), pose estimation (Sárándi *et al.*, 2021, 2023), and motion estimation (Luo *et al.*, 2020; Kocabas *et al.*, 2024). For applications requiring advanced human analysis, it is desirable to increase computational efficiency by having a multi-task network (Wallingford *et al.*, 2022) that can simultaneously accomplish these tasks in one forward pass. Recent advances in vision foundational models (Awais *et al.*, 2023) may be used as the base network, though such networks need to be compact and efficient enough for inference on mobile platforms.

End-To-End Perception. The detectors developed in this thesis have been combined with various downstream tasks, such as tracking (Linder *et al.*, 2016; EU FP7 Project SPENCER,

2020) and navigation (Paez-Granados *et al.*, 2022). In these setups, the final action of the autonomous agent, *e.g.* actuator command or monitor display, is determined from the sensor measurement via a pipeline composed of independent components, some of which are not based on deep learning. The end objectives of the pipeline are optimized in an ad-hoc fashion. The early components in the pipeline, such as person detection, typically receive no further adjustments once the network finishes training. End-to-end perception (Liang *et al.*, 2020a; Vu *et al.*, 2022) aims to determine the final action from the sensor measurement using solely deep neural networks. Paired with a differentiable objective, the entire pipeline can be optimized via gradient descent in an end-to-end fashion, potentially giving rise to an overall better-performing system.

— End —

Bibliography

Agarap, A. F. (2018). Deep Learning using Rectified Linear Units (ReLU). *arXiv:1803.08375*.

Aguirre, E. and García-Silvente, M. (2019). Using a Deep Learning Model on Images to Obtain a 2D Laser People Detector for a Mobile Robot. *International Journal of Computational Intelligence Systems*, **12**, 476–484.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall.

Akbari, H., Yuan, L., Qian, R., Chuang, W.-H., Chang, S.-F., Cui, Y., and Gong, B. (2021). VATT: Transformers for Multimodal Self-Supervised Learning from Raw Video, Audio and Text. In *Advances in Neural Information Processing Systems*.

Algan, G. and Ulusoy, I. (2019). Image Classification with Deep Learning in the Presence of Noisy Labels: A Survey. *arXiv:1912.05170*.

Alibeigi, M., Ljungbergh, W., Tonderski, A., Hess, G., Lilja, A., Lindstrom, C., Motorniuk, D., Fu, J., Widahl, J., and Petersson, C. (2023). Zenseact Open Dataset: A Large-Scale and Diverse Multimodal Dataset for Autonomous Driving. In *International Conference on Computer Vision*.

Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I., Fischer, M., and Savarese, S. (2016). 3D Semantic Parsing of Large-Scale Indoor Spaces. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Arras, K. O., Mozos, Ó. M., and Burgard, W. (2007). Using Boosted Features for the Detection of People in 2D Range Data. In *IEEE International Conference on Robotics and Automation*.

Arras, K. O., Grzonka, S., Luber, M., and Burgard, W. (2008). Efficient People Tracking in Laser Range Data Using a Multi-Hypothesis Leg-Tracker with Adaptive Occlusion Probabilities. In *IEEE International Conference on Robotics and Automation*.

Avigal, Y., Berscheid, L., Asfour, T., Kröger, T., and Goldberg, K. (2022). SpeedFolding: Learning Efficient Bimanual Folding of Garments. In *International Conference on Intelligent Robots and Systems*.

Awais, M., Naseer, M., Khan, S., Anwer, R. M., Cholakkal, H., Shah, M., Yang, M.-H., and Khan, F. S. (2023). Foundational Models Defining a New Era in Vision: A Survey and Outlook. *arXiv:2307.13721*.

Ba, J., Kiros, J. R., and Hinton, G. E. (2016). Layer Normalization. *arXiv:1607.06450*.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2015). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **39**, 2481–2495.

Bai, X., Hu, Z., Zhu, X., Huang, Q., Chen, Y., Fu, H., and Tai, C.-L. (2022). TransFusion: Robust LiDAR-Camera Fusion for 3D Object Detection with Transformers. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., and Gall, J. (2019). SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *International Conference on Computer Vision*.

Beltrán, J., Guindel, C., Moreno, F. M., Cruzado, D., García, F., and De La Escalera, A. (2018). BirdNet: A 3D Object Detection Framework from LiDAR Information. In *International Conference on Intelligent Transportation Systems (ITSC)*.

Benenson, R., Mathias, M., Timofte, R., and Van Gool, L. (2012). Pedestrian Detection at 100 Frames per Second. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum Learning. In *International Conference on Machine Learning*.

Bergstra, J., Yamins, D., and Cox, D. D. (2013). Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In *International Conference on Machine Learning*.

Bewley, A., Sun, P., Mensink, T., Anguelov, D., and Sminchisescu, C. (2020). Range Conditioned Dilated Convolutions for Scale Invariant 3D Object Detection. In *Conference on Robot Learning*.

Beyer, L., Hermans, A., and Leibe, B. (2016). DROW: Real-Time Deep Learning based Wheelchair Detection in 2D Range Data. *IEEE Robotics and Automation Letters (RA-L)*, **2**(2), 585–592.

Beyer, L., Breuers, S., Kurin, V., and Leibe, B. (2017). Towards a Principled Integration of Multi-camera Re-identification and Tracking Through Optimal Bayes Filters. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop*.

Beyer, L., Hermans, A., Linder, T., Arras, K. O., and Leibe, B. (2018). Deep Person Detection in 2D Range Data. *IEEE Robotics and Automation Letters (RA-L)*, **3**(3), 2726–2733.

Breiman, L. (2001). Random Forests. *Machine Learning*, **45**, 5–32.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A. D., and Vandergheynst, P. (2016). Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, **34**, 18–42.

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. (2020). nuScenes: A Multimodal Dataset for Autonomous Driving. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. In *European Conference on Computer Vision*.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging Properties in Self-Supervised Vision Transformers. In *International Conference on Computer Vision*.

Chai, Y., Sun, P., Ngiam, J., Wang, W., Caine, B., Vasudevan, V., Zhang, X., and Anguelov, D. (2021). To the Point: Efficient 3D Object Detection in the Range Image with Graph Convolution Kernels. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Chang, M.-F., Lambert, J. W., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., and Hays, J. (2019). Argoverse: 3D Tracking and Forecasting with Rich Maps. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Chen, C., Chen, Z., Zhang, J., and Tao, D. (2022a). SASA: Semantics-Augmented Set Abstraction for Point-based 3D Object Detection. In *Conference on Artificial Intelligence*.

Chen, J., Lei, B., Song, Q., Ying, H., Chen, D. Z., and Wu, J. (2020a). A Hierarchical Graph Network for 3D Object Detection on Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Chen, L., Wu, P., Chitta, K., Jaeger, B., Geiger, A., and Li, H. (2023a). End-to-end Autonomous Driving: Challenges and Frontiers. *arXiv:2306.16927*.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018). Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *European Conference on Computer Vision*.

Chen, Q., Sun, L., Cheung, E. C. H., and Yuille, A. L. (2020b). Every View Counts: Cross-View Consistency in 3D Object Detection with Hybrid-Cylindrical-Spherical Voxelization. In *Advances in Neural Information Processing Systems*.

Chen, Q., Sun, L., Wang, Z., Jia, K., and Yuille, A. (2020c). Object as Hotspots: An Anchor-Free 3D Object Detection Approach via Firing of Hotspots. In *European Conference on Computer Vision*.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020d). A Simple Framework for Contrastive Learning of Visual Representations. In *International Conference on Machine Learning*.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. (2020e). Big Self-Supervised Models are Strong Semi-Supervised Learners. In *Advances in Neural Information Processing Systems*.

Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. (2017). Multi-View 3D Object Detection Network for Autonomous Driving . In *IEEE Conference on Computer Vision and Pattern Recognition*.

Chen, X., Fan, H., Girshick, R., and He, K. (2020f). Improved Baselines with Momentum Contrastive Learning. *arXiv:2003.04297*.

Chen, X., Xie, S., and He, K. (2021). An Empirical Study of Training Self-Supervised Vision Transformers. In *International Conference on Computer Vision*.

Chen, X., Shi, S., Zhu, B., Cheung, K. C., Xu, H., and Li, H. (2022b). MPPNet: Multi-Frame Feature Intertwining with Proxy Points for 3D Temporal Object Detection. In *European Conference on Computer Vision*.

Chen, Y., Liu, S., Shen, X., and Jia, J. (2019). Fast Point R-CNN. In *International Conference on Computer Vision*.

Chen, Y., Li, Y., Zhang, X., Sun, J., and Jia, J. (2022c). Focal Sparse Convolutional Networks for 3D Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Chen, Y., Liu, J., Zhang, X., Qi, X., and Jia, J. (2023b). LargeKernel3D: Scaling up Kernels in 3D Sparse CNNs. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Chen, Y., Liu, J., Zhang, X., Qi, X., and Jia, J. (2023c). VoxelNeXt: Fully Sparse VoxelNet for 3D Object Detection and Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Cheng, B., Sheng, L., Shi, S., Yang, M., and Xu, D. (2021). Back-tracing Representative Points for Voting-based 3D Object Detection in Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., and Weller, A. (2020). Rethinking Attention with Performers. In *International Conference on Learning Representations*.

Choy, C., Gwak, J., and Savarese, S. (2019a). 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Choy, C., Park, J., and Koltun, V. (2019b). Fully Convolutional Geometric Features. In *International Conference on Computer Vision*.

Coradeschi, S. and Saffiatti, A. (2001). Perceptual Anchoring of Symbols for Action. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Cortes, C. and Vapnik, V. (1995). Support Vector Networks. *Machine Learning*, **20**, 273–297.

Cox, I. J. (1993). A Review of Statistical Data Association Techniques for Motion Correspondence. *International Journal of Computer Vision*, **10**.

Cui, J., Zha, H., Zhao, H., and Shibasaki, R. (2005). Tracking Multiple People Using Laser and Vision. In *International Conference on Intelligent Robots and Systems*.

Cui, J., Zha, H., Zhao, H., and Shibasaki, R. (2007). Laser-Based Detection and Tracking of Multiple People in Crowds. *Computer Vision and Image Understanding (CVIU)*, **106**.

Cybenko, G. V. (1989). Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals and Systems*, **2**, 303–314.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Nießner, M. (2017). ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Dalal, N. and Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. (2022). A Continual Learning Survey: Defying Forgetting in Classification Tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **44**.

Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., and Li, H. (2020). Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection. In *Conference on Artificial Intelligence*.

Deng, S., Liang, Z., Sun, L., and Jia, K. (2022). VISTA: Boosting 3D Object Detection via Dual Cross-VIew SpaTial Attention. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Ding, F., Palffy, A., Gavrila, D. M., and Lu, C. X. (2023). Hidden Gems: 4D Radar Scene Flow Learning Using Cross-Modal Supervision. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Ding, X., Zhang, X., Zhou, Y., Han, J., Ding, G., and Sun, J. (2022). Scaling Up Your Kernels to 31x31: Revisiting Large Kernel Design in CNNs. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021a). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021b). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., and Tian, Q. (2019). CenterNet: Keypoint Triplets for Object Detection. In *International Conference on Computer Vision*.

Dumoulin, V. and Visin, F. (2016). A Guide to Convolution Arithmetic for Deep Learning. *arXiv:1603.07285*.

Engelmann, F., Bokeloh, M., Fathi, A., Leibe, B., and Nießner, M. (2020). 3D-MPA: Multi Proposal Aggregation for 3D Semantic Instance Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Ess, A., Leibe, B., Schindler, K., and Van Gool, L. (2008). A mobile vision system for robust multi-person tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*.

EU FP7 Project SPENCER (2020). spencer_people_tracking. https://github.com/spencer-project/spencer_people_tracking. Accessed: 2023-12-28.

Fan, H., Yang, L., and Kankanhalli, M. (2021a). Point 4D Transformer Networks for Spatio-Temporal Modeling in Point Cloud Videos. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Fan, L., Xiong, X., Wang, F., Wang, N., and Zhang, Z. (2021b). RangeDet: In Defense of Range View for LiDAR-Based 3D Object Detection. In *International Conference on Computer Vision*.

Fan, L., Pang, Z., Zhang, T., Wang, Y.-X., Zhao, H., Wang, F., Wang, N., and Zhang, Z. (2022a). Embracing Single Stride 3D Object Detector with Sparse Transformer. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Fan, L., Wang, F., Wang, N., and Zhang, Z. (2022b). Fully Sparse 3D Object Detection. In *Advances in Neural Information Processing Systems*.

Fan, L., Wang, F., Wang, N., and Zhang, Z. (2023a). FSD V2: Improving Fully Sparse 3D Object Detection with Virtual Voxels. *arXiv:2308.03755*.

Fan, L., Yang, Y., Wang, F., Wang, N., and Zhang, Z. (2023b). Super Sparse 3D Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Feichtenhofer, C., Pinz, A., and Zisserman, A. (2017). Detect to Track and Track to Detect. In *International Conference on Computer Vision*.

Fod, A., Howard, A., and Mataric, M. J. (2002). A Laser-based People Tracker. In *IEEE International Conference on Robotics and Automation*.

Fukushima, K. (1980). Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, **36**, 193–202.

Ge, R., Ding, Z., Hu, Y., Wang, Y., Chen, S., Huang, L., and Li, Y. (2020). AFDet: Anchor Free One Stage 3D Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop*.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research*, **32**(11), 1231–1237.

Geyer, J., Kassahun, Y., Mahmudi, M., Ricou, X., Durgesh, R., Chung, A. S., Hauswald, L., Pham, V. H., Mühlegg, M., Dorn, S., Fernandez, T., Jänicke, M., Mirashi, S., Savani, C., Sturm, M., Vorobiov, O., Oelker, M., Garreis, S., and Schuberth, P. (2020). A2D2: Audi Autonomous Driving Dataset. *arXiv:2004.06320*.

Ghosh, A., Kumar, H., and Sastry, P. S. (2017). Robust Loss Functions under Label Noise for Deep Neural Networks. In *Conference on Artificial Intelligence*.

Girshick, R. B. (2015). Fast R-CNN. In *International Conference on Computer Vision*.

Glorot, X. and Bengio, Y. (2010). Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *International Conference on Artificial Intelligence and Statistics*.

Gockley, R., Forlizzi, J., and Simmons, R. G. (2007). Natural Person-Following Behavior for Social Robots. In *ACM/IEEE International Conference on Human-Robot Interaction*.

Goldberger, J. and Ben-Reuven, E. (2017). Training Deep Neural-networks using a Noise Adaptation Layer. In *International Conference on Learning Representations*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Graham, B. and van der Maaten, L. (2017). Submanifold Sparse Convolutional Networks. *arXiv:1706.01307*.

Graham, B., Engelcke, M., and van der Maaten, L. (2018). 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Guan, T., Wang, J., Lan, S., Chandra, R., Wu, Z., Davis, L. S., and Manocha, D. (2021). M3DETR: Multi-representation, Multi-scale, Mutual-relation 3D Object Detection with Transformers. In *Winter Conference on Applications of Computer Vision*.

Guo, M.-H., Cai, J.-X., Liu, Z.-N., Mu, T.-J., Martin, R. R., and Hu, S.-M. (2021). PCT: Point Cloud Transformer. *Computational Visual Media*, **7**, 187–199.

Gwak, J., Choy, C. B., and Savarese, S. (2020). Generative Sparse Detection Networks for 3D Single-shot Object Detection. In *European Conference on Computer Vision*.

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I. W., and Sugiyama, M. (2018a). Co-Teaching: Robust Training of Deep Neural Networks with Extremely Noisy Labels. In *Neural Information Processing Systems*.

Han, B., Yao, J., Gang, N., Zhou, M., Tsang, I., Zhang, Y., and Sugiyama, M. (2018b). Masking: A New Perspective of Noisy Supervision. In *Neural Information Processing Systems*.

Han, W., Khorrami, P., Paine, T. L., Ramachandran, P., Babaeizadeh, M., Shi, H., Li, J., Yan, S., and Huang, T. S. (2016). Seq-NMS for Video Object Detection. *arXiv:1602.08465*.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*. Springer.

He, C., Zeng, H., Huang, J., Hua, X.-S., and Zhang, L. (2020a). Structure Aware Single-stage 3D Object Detection from Point Cloud. In *IEEE Conference on Computer Vision and Pattern Recognition*.

He, C., Li, R., Li, S., and Zhang, L. (2022a). Voxel Set Transformer: A Set-to-Set Approach to 3D Object Detection from Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *International Conference on Computer Vision*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. (2017). Mask R-CNN. In *International Conference on Computer Vision*.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020b). Momentum Contrast for Unsupervised Visual Representation Learning. In *IEEE Conference on Computer Vision and Pattern Recognition*.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022b). Masked Autoencoders Are Scalable Vision Learners. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Hemachandra, S., Kollar, T., Roy, N., and Teller, S. (2011). Following and Interpreting Narrated Guided Tours. In *IEEE International Conference on Robotics and Automation*.

Hermans, A., Beyer, L., and Leibe, B. (2017). In Defense of the Triplet Loss for Person Re-Identification. *arXiv:1703.07737*.

Hess, J., Beinhofer, M., and Burgard, W. (2014). A Probabilistic Approach to High-Confidence Cleaning Guarantees for Low-Cost Cleaning Robots. In *IEEE International Conference on Robotics and Automation*.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, **9**(8), 1735–1780.

Hu, W., Zhao, H., Jiang, L., Jia, J., and Wong, T.-T. (2021a). Bidirectional Projection Network for Cross Dimensional Scene Understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Hu, Y., Ding, Z., Ge, R., Shao, W., Huang, L., Li, K., and Liu, Q. (2022). AFDetV2: Rethinking the Necessity of the Second Stage for Object Detection from Point Clouds. In *Conference on Artificial Intelligence*.

Hu, Z., Bai, X., Shang, J., Zhang, R., Dong, J., Wang, X., Sun, G., Fu, H., and Tai, C.-L. (2021b). VMNet: Voxel-Mesh Network for Geodesic-Aware 3D Semantic Segmentation. In *International Conference on Computer Vision*.

Huang, T., Liu, Z., Chen, X., and Bai, X. (2020). EPNet: Enhancing Point Features with Image Semantics for 3D Object Detection. In *European Conference on Computer Vision*.

International Federation of Robotics (2021). World Robotics 2021 – Service Robots report released. https://ifr.org/ifr-press-releases/news/service-robots-hit-double-digit-growth-worldwide. Accessed: 2023-12-28.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*.

Jafari, O. H., Mitzel, D., and Leibe, B. (2014). Real-Time RGB-D based People Detection and Tracking for Mobile Robots and Head-Worn Cameras. In *IEEE International Conference on Robotics and Automation*.

Jia, D. and Leibe, B. (2021). Person-MinkUNet: 3D Person Detection with LiDAR Point Cloud. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop*.

Jia, D., Hermans, A., and Leibe, B. (2020). DR-SPAAM: A Spatial-Attention and Auto-regressive Model for Person Detection in 2D Range Data. In *International Conference on Intelligent Robots and Systems*.

Jia, D., Steinweg, M., Hermans, A., and Leibe, B. (2021). Self-Supervised Person Detection in 2D Range Data using a Calibrated Camera. In *IEEE International Conference on Robotics and Automation*.

Jia, D., Hermans, A., and Leibe, B. (2022a). 2D vs. 3D LiDAR-based Person Detection on Mobile Robots. In *International Conference on Intelligent Robots and Systems*.

Jia, D., Hermans, A., and Leibe, B. (2022b). Global Hierarchical Attention for 3D Point Cloud Analysis. In *German Conference on Pattern Recognition*.

Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. (2018). MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels. In *International Conference on Machine Learning*.

Jiang, L., Zhao, H., Liu, S., Shen, X., Fu, C.-W., and Jia, J. (2019). Hierarchical Point-Edge Interaction Network for Point Cloud Semantic Segmentation. In *International Conference on Computer Vision*.

Jindal, I., Nokleby, M. S., and Chen, X. (2016). Learning Deep Networks from Noisy Labels with Dropout Regularization. In *IEEE International Conference on Data Mining*.

Kanezaki, A., Matsushita, Y., and Nishida, Y. (2021). RotationNet for Joint Object Categorization and Unsupervised Pose Estimation from Multi-View Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **43**.

Kang, K., Ouyang, W., Li, H., and Wang, X. (2016). Object Detection from Video Tubelets with Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Kendall, A. and Gal, Y. (2017). What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In *Advances in Neural Information Processing Systems*.

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

Kingma, D. P. and Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

Klein, F. (1893). Vergleichende Betrachtungen über neuere geometrische Forschungen. *Mathematische Annalen*, **43**, 63–100.

Kleinehagenbrock, M., Lang, S., Fritsch, J., Lömker, F., Fink, G. A., and Sagerer, G. (2002). Person Tracking with A Mobile Robot Based on Multi-Modal Anchoring. In *IEEE International Workshop on Robot and Human Interactive Communication*.

Kluge, B., Köhler, C., and Prassler, E. (2001). Fast and Robust Tracking of Multiple Moving Objects with a Laser Range Finder. In *IEEE International Conference on Robotics and Automation*.

Kocabas, M., Yuan, Y., Molchanov, P., Guo, Y., Black, M. J., Hilliges, O., Kautz, J., and Iqbal, U. (2024). PACE: Human and Motion Estimation from in-the-wild Videos. In *International Conference on 3D Vision*.

Ku, J., Mozifian, M., Lee, J., Harakeh, A., and Waslander, S. L. (2017). Joint 3D Proposal Generation and Object Detection from View Aggregation. In *International Conference on Intelligent Robots and Systems*.

Kuang, H., Wang, B., An, J., Zhang, M., and Zhang, Z. (2020). Voxel-FPN: Multi-Scale Voxel Feature Aggregation for 3D Object Detection from LIDAR Point Clouds. *Sensors*, **20**.

Kuhn, H. W. (1955). The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, **2**(1–2), 83–97.

Kumar, M. P., Packer, B., and Koller, D. (2010). Self-Paced Learning for Latent Variable Models. In *Neural Information Processing Systems*.

Lai, X., Liu, J., Jiang, L., Wang, L., Zhao, H., Liu, S., Qi, X., and Jia, J. (2022). Stratified Transformer for 3D Point Cloud Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Lai, X., Chen, Y., Lu, F., Liu, J., and Jia, J. (2023). Spherical Transformer for LiDAR-based 3D Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Landrieu, L. and Boussaha, M. (2019). Point Cloud Oversegmentation with Graph-Structured Deep Metric Learning. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Landrieu, L. and Simonovsky, M. (2018). Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. (2019). PointPillars: Fast Encoders for Object Detection From Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Law, H. and Deng, J. (2020). CornerNet: Detecting Objects as Paired Keypoints. *International Journal of Computer Vision*.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, **1**, 541–551.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. (2019). Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *International Conference on Machine Learning*.

Leigh, A., Pineau, J., Olmedo, N., and Zhang, H. (2015). Person Tracking and Following with 2D Laser Scanners. In *IEEE International Conference on Robotics and Automation*.

Lewandowski, B., Liebner, J., Wengefeld, T., Mueller, St., and Gross, H. M. (2019). A Fast and Robust 3D Person Detector and Posture Estimator for Mobile Robotic Application. In *IEEE International Conference on Robotics and Automation*.

Li, G., Müller, M., Qian, G., Perez, I. C. D., Abualshour, A., Thabet, A. K., and Ghanem, B. (2021a). DeepGCNs: Making GCNs go as deep as CNNs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Li, X. R. and Bar-Shalom, Y. (1996). Tracking in clutter with nearest neighbor filters: analysis and performance. *IEEE Transactions on Aerospace and Electronic Systems*, **32**.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., and Chen, B. (2018). PointCNN: Convolution on X-Transformed Points. In *Neural Information Processing Systems*.

Li, Y., Yu, A. W., Meng, T., Caine, B., Ngiam, J., Peng, D., Shen, J., Wu, B., Lu, Y., Zhou, D., Le, Q. V., Yuille, A., and Tan, M. (2022a). DeepFusion: Lidar-Camera Deep Fusion for Multi-Modal 3D Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Li, Y., Chen, Y., Qi, X., Li, Z., Sun, J., and Jia, J. (2022b). Unifying Voxel-based Representation with Transformer for 3D Object Detection. In *Advances in Neural Information Processing Systems*.

Li, Z., Wang, F., and Wang, N. (2021b). LiDAR R-CNN: An Efficient and Universal 3D Object Detector. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Liang, M., Yang, B., Wang, S., and Urtasun, R. (2018). Deep Continuous Fusion for Multi-sensor 3D Object Detection. In *European Conference on Computer Vision*.

Liang, M., Yang, B., Chen, Y., Hu, R., and Urtasun, R. (2019). Multi-Task Multi-Sensor Fusion for 3D Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Liang, M., Yang, B., Zeng, W., Chen, Y., Hu, R., Casas, S., and Urtasun, R. (2020a). PnPNet: End-to-End Perception and Prediction With Tracking in the Loop. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Liang, Z., Zhang, M., Zhang, Z., Zhao, X., , and Pu, S. (2020b). RangeRCNN: Towards Fast and Accurate 3D Object Detection with Range Image Representation. *arXiv:2009.00206*.

Liang, Z., Zhang, Z., Zhang, M., Zhao, X., and Pu, S. (2021). RangeIoUDet: Range Image based Real-Time 3D Object Detector Optimized by Intersection over Union. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Liao, Y., Xie, J., and Geiger, A. (2022). KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. (2014). Microsoft COCO: Common Objects in Context. *arXiv:1405.0312*.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and He, K. (2017). Focal Loss for Dense Object Detection. In *International Conference on Computer Vision*.

Linder, T., Breuers, S., Leibe, B., and Arras, K. O. (2016). On Multi-Modal People Tracking from Mobile Platforms in Very Crowded and Dynamic Environments. In *IEEE International Conference on Robotics and Automation*.

Linder, T., Pfeiffer, K. Y., Vaskevicius, N., Schirmer, R., and Arras, K. O. (2020). Accurate Detection and 3D Localization of Humans using a Novel YOLO-based RGB-D Fusion Approach and Synthetic Training Data. In *IEEE International Conference on Robotics and Automation*.

Linder, T., Vaskevicius, N., Schirmer, R., and Arras, K. O. (2021). Cross-Modal Analysis of Human Detection for Robotics: An Industrial Case Study. In *International Conference on Intelligent Robots and Systems*.

Lindstrom, M. and Eklundh, J.-O. (2001). Detecting and Tracking Moving Objects from a Mobile Platform Using a Laser Range Scanner. In *International Conference on Intelligent Robots and Systems*.

Liu, J., Chen, Y., Ye, X., Tian, Z., Tan, X., and Qi, X. (2022a). Spatial Pruned Sparse Convolution for Efficient 3D Object Detection. In *Advances in Neural Information Processing Systems*.

Liu, J.-J., Hou, Q., Cheng, M.-M., Wang, C., and Feng, J. (2020a). Improving Convolutional Networks with Self-Calibrated Convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In *European Conference on Computer Vision*.

Liu, Z., Zhao, X., Huang, T., Hu, R., Zhou, Y., and Bai, X. (2020b). TANet: Robust 3D Object Detection from Point Clouds with Triple Attention. In *Conference on Artificial Intelligence*.

Liu, Z., Zhang, Z., Cao, Y., Hu, H., and Tong, X. (2021a). Group-Free 3D Object Detection via Transformers. In *International Conference on Computer Vision*.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021b). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *International Conference on Computer Vision*.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. (2022b). A ConvNet for the 2020s. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Liu, Z., Tang, H., Amini, A., Yang, X., Mao, H., Rus, D., and Han, S. (2023a). BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation. In *IEEE International Conference on Robotics and Automation*.

Liu, Z., Yang, X., Tang, H., Yang, S., and Han, S. (2023b). FlatFormer: Flattened Window Attention for Efficient Point Cloud Transformer. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Loquercio, A., Segu, M., and Scaramuzza, D. (2020). A General Framework for Uncertainty Estimation in Deep Learning. *IEEE Robotics and Automation Letters (RA-L)*.

Loshchilov, I. and Hutter, F. (2017). SGDR: stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*.

Loshchilov, I. and Hutter, F. (2019). Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

Lu, D. V. and Smart, W. D. (2013). Towards More Efficient Navigation for Robots and Humans. In *International Conference on Intelligent Robots and Systems*.

Lu, Y., Lu, C., and Tang, C.-K. (2017). Online Video Object Detection Using Association LSTM. In *International Conference on Computer Vision*.

Luber, M., Tipaldi, G. D., and Arras, K. O. (2011). Place-dependent people tracking. *International Journal of Robotics Research*, **30**(3), 280–293.

Luo, Z., Golestaneh, S. A., and Kitani, K. M. (2020). 3D Human Motion Estimation via Motion Compression and Refinement. In *Asian Conference on Computer Vision*.

MacLachlan, R. and Mertz, C. (2006). Tracking of Moving Objects from a Moving Vehicle Using a Scanning Laser Rangefinder. In *2006 IEEE Intelligent Transportation Systems Conference*.

Mao, J., Wang, X., and Li, H. (2019). Interpolated Convolutional Networks for 3D Point Cloud Understanding. In *International Conference on Computer Vision*.

Mao, J., Niu, M., Jiang, C., Liang, X., Li, Y., Ye, C., Zhang, W., Li, Z., Yu, J., Xu, C., *et al.* (2021a). One Million Scenes for Autonomous Driving: ONCE Dataset. *arXiv:2106.11037*.

Mao, J., Xue, Y., Niu, M., Bai, H., Feng, J., Liang, X., Xu, H., and Xu, C. (2021b). Voxel Transformer for 3D Object Detection. In *International Conference on Computer Vision*.

Mao, J., Shi, S., Wang, X., and Li, H. (2022). 3D Object Detection for Autonomous Driving: A Comprehensive Survey. *arXiv:2206.09474*.

Martin-Martin, R., Patel, M., Rezatofighi, H., Shenoi, A., Gwak, J., Frankel, E., Sadeghian, A., and Savarese, S. (2021). JRDB: A Dataset and Benchmark for Visual Perception for Navigation in Human Environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Maturana, D. and Scherer, S. (2015). VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In *International Conference on Intelligent Robots and Systems*.

Mcculloch, W. and Pitts, W. (1943). A Logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, **5**, 127–147.

Menon, A. K., Rawat, A. S., Reddi, S. J., and Kumar, S. (2020). Can gradient clipping mitigate label noise? In *International Conference on Learning Representations*.

Meyer, G. P., Laddha, A., Kee, E., Vallespi-Gonzalez, C., and Wellington, C. K. (2019). LaserNet: An Efficient Probabilistic 3D Object Detector for Autonomous Driving. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Meyer, G. P., Charland, J., Pandey, S., Laddha, A. G., Gautam, S., Vallespi-Gonzalez, C., and Wellington, C. K. (2020). LaserFlow: Efficient and Probabilistic Object Detection and Motion Forecasting. *IEEE Robotics and Automation Letters (RA-L)*, **6**, 526–533.

Milioto, A., Vizzo, I., Behley, J., and Stachniss, C. (2019). RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *International Conference on Intelligent Robots and Systems*.

Minsky, M. and Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA.

Misra, I., Girdhar, R., and Joulin, A. (2021). An End-to-End Transformer Model for 3D Object Detection. In *International Conference on Computer Vision*.

MMDetection3D (2020). MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d. Accessed: 2023-12-28.

Montemerlo, M., Thrun, S., and Whittaker, W. (2002). Conditional Particle Filters for Simultaneous Mobile Robot Localization and People-Tracking. In *IEEE International Conference on Robotics and Automation*.

Mucientes, M. and Burgard, W. (2006). Multiple Hypothesis Tracking of Clusters of People. In *International Conference on Intelligent Robots and Systems*.

Munaro, M. and Menegatti, E. (2014). Fast RGB-D People Tracking for Service Robots. *Autonomous Robots*, **37**.

Narayan, N., Sankaran, N., Setlur, S., and Govindaraju, V. (2018). Re-Identification for Online Person Tracking by Modeling Space-Time Continuum. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop*.

National Highway Traffic Safety Administration (2016). Preliminary Report, PE 16-007. https://static.nhtsa.gov/odi/inv/2016/INCLA-PE16007-7876.PDF. Accessed: 2023-12-28.

Navarro-Serment, L. E., Mertz, C., Vandapel, N., and Hebert, M. (2008). LADAR-based Pedestrian Detection and Tracking. In *IEEE International Conference on Robotics and Automation Workshop*.

Nekrasov, A., Schult, J., Litany, O., Leibe, B., and Engelmann, F. (2021). Mix3D: Out-of-Context Data Augmentation for 3D Scenes. In *International Conference on 3D Vision*.

Noh, H., Hong, S., and Han, B. (2015). Learning Deconvolution Network for Semantic Segmentation. In *International Conference on Computer Vision*.

Noh, J., Lee, S., and Ham, B. (2021). HVPR: Hybrid Voxel-Point Representation for Single-stage 3D Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Paez-Granados, D. F., He, Y., Gonon, D. J., Jia, D., Leibe, B., Suzuki, K., and Billard, A. (2022). Pedestrian-Robot Interactions on Autonomous Crowd Navigation: Reactive Control Methods and Evaluation Metrics. In *International Conference on Intelligent Robots and Systems*.

Pan, X., Xia, Z., Song, S., Li, L. E., and Huang, G. (2021). 3D Object Detection With Pointformer. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Pantofaru, C. (2010). ROS leg_detector package. https://wiki.ros.org/leg_detector. Accessed 2018-02-22.

Park, C., Jeong, Y., Cho, M., and Park, J. (2022). Fast Point Transformer. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Patil, A., Malla, S., Gang, H., and Chen, Y.-T. (2019). The H3D Dataset for Full-Surround 3D Multi-Object Detection and Tracking in Crowded Urban Scenes. In *IEEE International Conference on Robotics and Automation*.

Pham, Q.-H., Sevestre, P., Pahwa, R. S., Zhan, H., Pang, C. H., Chen, Y., Mustafa, A., Chandrasekhar, V., and Lin, J. (2020). A*3D Dataset: Towards Autonomous Driving in Challenging Environments. In *IEEE International Conference on Robotics and Automation*.

Piewak, F., Pinggera, P., Schafer, M., Peter, D., Schwarz, B., Schneider, N., Enzweiler, M., Pfeiffer, D., and Zollner, M. (2018). Boosting LiDAR-based Semantic Labeling by Cross-Modal Training Data Generation. In *European Conference on Computer Vision*.

Pitropov, M., Garcia, D. E., Rebello, J., Smart, M., Wang, C., Czarnecki, K., and Waslander, S. (2021). Canadian Adverse Driving Conditions Dataset. *International Journal of Robotics Research*, **40**(4-5), 681–690.

Prassler, E., Scholz, J., and Elfes, A. (1999). Tracking People in a Railway Station During Rush-Hour. In *International Conference on Computer Vision Systems*.

Preparata, F. P. and Shamos, M. I. (1985). *Computational Geometry - An Introduction*. Springer.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017a). PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Neural Information Processing Systems*.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017b). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Qi, C. R., Liu, W., Wu, C., Su, H., and Guibas, L. J. (2018). Frustum PointNets for 3D Object Detection from RGB-D Data. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Qi, C. R., Litany, O., He, K., and Guibas, L. J. (2019). Deep Hough Voting for 3D Object Detection in Point Clouds. In *International Conference on Computer Vision*.

Qi, C. R., Chen, X., Litany, O., and Guibas, L. J. (2020). ImVoteNet: Boosting 3D Object Detection in Point Clouds with Image Votes. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Qian, R., Lai, X., and Li, X. (2022). 3D Object Detection for Autonomous Driving: A Survey. *Pattern Recognition*, **130**, 108796.

Qian, X., Yu-kun, L., Jing, W., Zhoutao, W., Yiming, Z., Kai, X., and Jun, W. (2020). ML-CVNet: Multi-Level Context VoteNet for 3D Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C., and Dosovitskiy, A. (2021). Do Vision Transformers See Like Convolutional Neural Networks? *arXiv:2108.08810*.

Rapoport-Lavie, M. and Raviv, D. (2021). It's All Around You: Range-Guided Cylindrical Network for 3D Object Detection. In *International Conference on Computer Vision Workshop*.

Reda, F. A., Liu, G., Shih, K. J., Kirby, R., Barker, J., Tarjan, D., Tao, A., and Catanzaro, B. (2018). SDC-Net: Video Prediction Using Spatially-Displaced Convolution. In *European Conference on Computer Vision*.

Redmon, J. and Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *arXiv:1804.02767*.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Reid, D. (1979). An Algorithm for Tracking Multiple Targets. *IEEE Transactions on Automatic Control*, **24**.

Ren, M., Zeng, W., Yang, B., and Urtasun, R. (2018). Learning to Reweight Examples for Robust Deep Learning. In *International Conference on Machine Learning*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Neural Information Processing Systems*.

Riegler, G., Ulusoy, A. O., and Geiger, A. (2017). OctNet: Learning Deep 3D Representations at High Resolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, **9351**, 234–241.

Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, **65**(6), 386–408.

Rukhovich, D., Vorontsova, A., and Konushin, A. (2022). FCAF3D: Fully Convolutional Anchor-Free 3D Object Detection. In *European Conference on Computer Vision*.

Rukhovich, D., Vorontsova, A., and Konushin, A. (2023). TR3D: Towards Real-Time Indoor 3D Object Detection. In *International Conference on Image Processing*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, pages 318–362. MIT Press.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. (2018). How Does Batch Normalization Help Optimization? In *Neural Information Processing Systems*.

Sárándi, I., Linder, T., Arras, K. O., and Leibe, B. (2021). MeTRAbs: Metric-Scale Truncation-Robust Heatmaps for Absolute 3D Human Pose Estimation. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, **3**.

Sárándi, I., Hermans, A., and Leibe, B. (2023). Learning 3D Human Pose Estimation from Dozens of Datasets using a Geometry-Aware Autoencoder to Bridge Between Skeleton Formats. In *Winter Conference on Applications of Computer Vision*.

Schapire, R. E. (2013). Explaining AdaBoost. In *Empirical inference*, pages 37–52. Springer.

Scheutz, M., McRaven, J., and Cserey, G. (2004). Fast, Reliable, Adaptive, Bimodal People Tracking for Indoor Environments. In *International Conference on Intelligent Robots and Systems*.

Schmidt, S., Rao, Q., Tatsch, J., and Knoll, A. (2020). Advanced Active Learning Strategies for Object Detection. In *IEEE Intelligent Vehicles Symposium*.

Schulz, D. (2006). A probabilistic Exemplar Approach to Combine Laser and Vision for Person Tracking. In *Robotics: Science and Systems Conference*.

Schulz, D., Burgard, W., Fox, D., and Cremers, A. (2001). Tracking Multiple Moving Targets with A Mobile Robot Using Particle Filters and Statistical Data Association. In *IEEE International Conference on Robotics and Automation*.

Schulz, D., Burgard, W., Fox, D., and Cremers, A. B. (2003). People Tracking with Mobile Robots Using Sample-Based Joint Probabilistic Data Association Filters. *International Journal of Robotics Research*, **22**(2), 99–116.

Sheeny, M., De Pellegrin, E., Mukherjee, S., Ahrabian, A., Wang, S., and Wallace, A. (2020). RADIATE: A Radar Dataset for Automotive Perception. *arXiv:2010.09076*.

Sheng, H., Cai, S., Liu, Y., Deng, B., Huang, J., Hua, X.-S., and Zhao, M.-J. (2021). Improving 3D Object Detection with Channel-wise Transformer. In *International Conference on Computer Vision*.

Shi, G., Li, R., and Ma, C. (2022a). PillarNet: Real-Time and High-Performance Pillar-based 3D Object Detection. In *European Conference on Computer Vision*.

Shi, S., Wang, X., and Li, H. (2019). PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Shi, S., Wang, Z., Shi, J., Wang, X., and Li, H. (2020a). 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*.

Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., and Li, H. (2020b). PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Shi, S., Jiang, L., Deng, J., Wang, Z., Guo, C., Shi, J., Wang, X., and Li, H. (2022b). PV-RCNN++: Point-Voxel Feature Set Abstraction With Local Vector Representation for 3D Object Detection. *International Journal of Computer Vision*, **131**, 531–551.

Shu, J., Xie, Q., Yi, L., Zhao, Q., Zhou, S., Xu, Z., and Meng, D. (2019). Meta-Weight-Net: Learning an Explicit Mapping For Sample Weighting. In *Neural Information Processing Systems*.

Sindagi, V. A., Zhou, Y., and Tuzel, O. (2019). MVX-Net: Multimodal voxelnet for 3D object detection. In *IEEE International Conference on Robotics and Automation*.

Smith, L. N. and Topin, N. (2019). Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. In *Artificial intelligence and machine learning for multi-domain operations applications*.

Song, H., Kim, M., Park, D., and Lee, J.-G. (2020). Learning from Noisy Labels with Deep Neural Networks: A Survey. *arXiv:2007.08199*.

Song, S., Lichtenberg, S. P., and Xiao, J. (2015). SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, **15**(56), 1929–1958.

Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.-H., and Kautz, J. (2018). SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., *et al.* (2020). Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Sun, P., Wang, W., Chai, Y., Elsayed, G., Bewley, A., Zhang, X., Sminchisescu, C., and Anguelov, D. (2021). RSN: Range Sparse Net for Efficient, Accurate LiDAR 3D Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Sun, P., Tan, M., Wang, W., Liu, C., Xia, F., Leng, Z., and Anguelov, D. (2022). SWFormer: Sparse Window Transformer for 3D Object Detection in Point Clouds. In *European Conference on Computer Vision*.

Takezoe, R., Liu, X., Mao, S., Chen, M. T., Feng, Z., Zhang, S., and Wang, X. (2023). Deep Active Learning for Computer Vision: Past and Future. *APSIPA Transactions on Signal and Information Processing*, **12**.

Tan, M., Pang, R., and Le, Q. V. (2020). EfficientDet: Scalable and Efficient Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. (2020). Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *Neural Information Processing Systems*.

Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., and Han, S. (2020). Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution. In *European Conference on Computer Vision*.

Tang, H., Liu, Z., Li, X., Lin, Y., and Han, S. (2022). TorchSparse: Efficient Point Cloud Inference Engine. In *Conference on Machine Learning and Systems*.

Tatarchenko, M., Park, J., Koltun, V., and Zhou, Q.-Y. (2018). Tangent Convolutions for Dense Prediction in 3D. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. (2020). Efficient Transformers: A Survey. *arXiv:2009.06732*.

Taylor, G. and Kleeman, L. (2004). A Multiple Hypothesis Walking Person Tracker with Switched Dynamic Model. In *Australasian Conference on Robotics and Automation*.

Thomas, H., Qi, C., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. (2019). KP-Conv: Flexible and Deformable Convolution for Point Clouds. In *International Conference on Computer Vision*.

Topp, E. A. and Christensen, H. I. (2005). Tracking for Following and Passing Persons. In *International Conference on Intelligent Robots and Systems*.

Toyama, K., Krumm, J., Brumitt, B., and Meyers, B. (1999). Wallflower: Principles and Practice of Background Maintenance. In *International Conference on Computer Vision*.

Tripathi, S., Lipton, Z. C., Belongie, S. J., and Nguyen, T. Q. (2016). Context Matters: Refining Object Detection in Video with Recurrent Neural Networks. *arXiv:1607.04648*.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. S. (2016). Instance Normalization: The Missing Ingredient for Fast Stylization. *arXiv:1607.08022*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is All You Need. In *Neural Information Processing Systems*.

Viola, P. and Jones, M. (2002). Robust Real-time Object Detection. *International Journal of Computer Vision*, **57**(2), 137–154.

Vora, S., Lang, A. H., Helou, B., and Beijbom, O. (2019). PointPainting: Sequential Fusion for 3D Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Vu, D., Ngo, B., and Phan, H. (2022). HybridNets: End-to-End Perception Network. *arXiv:2203.09035*.

Wallingford, M., Li, H., Achille, A., Ravichandran, A., Fowlkes, C., Bhotika, R., and Soatto, S. (2022). Task Adaptive Parameter Sharing for Multi-Task Learning. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Wang, B. H., Chao, W.-L., Wang, Y., Hariharan, B., Weinberger, K. Q., and Campbell, M. (2019a). LDLS: 3-D Object Segmentation Through Label Diffusion From 2-D Images. *RAL*, **4**(3), 2902–2909.

Wang, C., Ma, C., Zhu, M., and Yang, X. (2021a). PointAugmenting: Cross-Modal Augmentation for 3D Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2023a). YOLOv7: Trainable Bag-Of-Freebies Sets New State-Of-The-Art for Real-Time Object Detectors. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Wang, D. Z. and Posner, I. (2015). Voting for Voting in Online Point Cloud Object Detection. In *Robotics: Science and Systems Conference*.

Wang, G., Tian, B., Ai, Y., Xu, T., Chen, L., and Cao, D. (2021b). CenterNet3D: An Anchor free Object Detector for Autonomous Driving. *IEEE Transactions on Intelligent Transportation Systems*.

Wang, H., Shi, C., Shi, S., Lei, M., Wang, S., He, D., Schiele, B., and Wang, L. (2023b). DSVT: Dynamic Sparse Voxel Transformer with Rotated Sets. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Wang, L., Huang, Y., Hou, Y., Zhang, S., and Shan, J. (2019b). Graph Attention Convolution for Point Cloud Semantic Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Wang, L., Zhang, X., Su, H., and Zhu, J. (2023c). A Comprehensive Survey of Continual Learning: Theory, Method and Application. *arXiv:2302.00487*.

Wang, S., Suo, S., Ma, W.-C., Pokrovsky, A., and Urtasun, R. (2018a). Deep Parametric Continuous Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Wang, S., Zhou, Y., Yan, J., and Deng, Z. (2018b). Fully Motion-Aware Network for Video Object Detection. In *European Conference on Computer Vision*.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. (2020a). Linformer: Self-Attention with Linear Complexity. *arXiv:2006.04768*.

Wang, Y. and Solomon, J. M. (2019). Deep Closest Point: Learning Representations for Point Cloud Registration. In *International Conference on Computer Vision*.

Wang, Y. and Solomon, J. M. (2021). Object DGCNN: 3D Object Detection using Dynamic Graphs. In *Advances in Neural Information Processing Systems*.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2019c). Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics*, **38**.

Wang, Y., Ma, X., Chen, Z., Luo, Y., Yi, J., and Bailey, J. (2019d). Symmetric cross entropy for robust learning with noisy labels. In *International Conference on Computer Vision*.

Wang, Y., Fathi, A., Kundu, A., Ross, D. A., Pantofaru, C., Funkhouser, T., and Solomon, J. (2020b). Pillar-based Object Detection for Autonomous Driving. In *European Conference on Computer Vision*.

Wang, Y., Guizilini, V., Zhang, T., Wang, Y., Zhao, H., , and Solomon, J. M. (2021c). DETR3D: 3D Object Detection from Multi-view Images via 3D-to-2D Queries. In *Conference on Robot Learning*.

Wang, Z., Ding, S., Li, Y., Zhao, M., Roychowdhury, S., Wallin, A., Fenn, J., Sapiro, G., Qiu, Q., Martin, L., and Ryvola, S. (2020c). Cirrus: A Long-range Bi-pattern LiDAR Dataset. *arXiv:2012.02938*.

Weinrich, C., Wengefeld, T., Schroeter, C., and Gross, H.-M. (2014). People Detection and Distinction of Their Walking Aids in 2D Laser Range Data Based on Generic Distance-Invariant Features. In *IEEE International Symposium on Robot and Human Interactive Communication*.

Weng, X., Wang, J., Held, D., and Kitani, K. (2020). 3D Multi-Object Tracking: A Baseline and New Evaluation Metrics. In *International Conference on Intelligent Robots and Systems*.

Weng, X., Man, Y., Cheng, D., Park, J., O'Toole, M., and Kitani, K. (2021). All-In-One Drive: A Large-Scale Comprehensive Perception Dataset with High-Density Long-Range Point Clouds. www.aiodrive.org. Accessed 2024-05-20.

Werbos, P. J. (1981). Applications of Advances in Nonlinear Sensitivity Analysis. In *System Modeling and Optimization*.

Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J. K., Ramanan, D., Carr, P., and Hays, J. (2021). Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Advances in Neural Information Processing Systems*.

Wu, H., Chen, Y., Wang, N., and Zhang, Z. (2019a). Sequence Level Semantics Aggregation for Video Object Detection. In *International Conference on Computer Vision*.

Wu, W., Qi, Z., and Fuxin, L. (2019b). PointConv: Deep Convolutional Networks on 3D Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Wu, X., Lao, Y., Jiang, L., Liu, X., and Zhao, H. (2022). Point transformer V2: Grouped Vector Attention and Partition-based Pooling. In *Neural Information Processing Systems*.

Wu, Y. and He, K. (2018). Group Normalization. In *European Conference on Computer Vision*.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3D ShapeNets: A Deep Representation for Volumetric Shapes. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Xavier, J., Pacheco, M., Castro, D., Ruano, A., and Nunes, U. (2005). Fast Line, Arc/Circle and Leg Detection from Laser Scan Data in a Player Driver. In *IEEE International Conference on Robotics and Automation*.

Xiao, F. and Lee, Y. J. (2017). Video Object Detection with an Aligned Spatial-Temporal Memory. In *European Conference on Computer Vision*.

Xiao, P., Shao, Z., Hao, S., Zhang, Z., Chai, X., Jiao, J., Li, Z., Wu, J., Sun, K., Jiang, K., Wang, Y., and Yang, D. (2021). PandaSet: Advanced Sensor Suite Dataset for Autonomous Driving. In *IEEE International Intelligent Transportation Systems Conference (ITSC)*.

Xiao, T., Xia, T., Yang, Y., Huang, C., and Wang, X. (2015). Learning from Massive Noisy Labeled Data for Image Classification. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Xie, Q., Lai, Y.-K., Wu, J., Wang, Z., Lu, D., Wei, M., and Wang, J. (2021). VENet: Voting Enhancement Network for 3D Object Detection. In *International Conference on Computer Vision*.

Xu, D., Anguelov, D., and Jain, A. (2018a). PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Xu, J., Miao, Z., Zhang, D., Pan, H., Liu, K., Hao, P., Zhu, J., Sun, Z., Li, H., and Zhan, X. (2022). INT: Towards Infinite-frames 3D Detection with An Efficient Framework. In *European Conference on Computer Vision*.

Xu, Y., Fan, T., Xu, M., Zeng, L., and Qiao, Y. (2018b). SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters. In *European Conference on Computer Vision*.

Yan, Y. (2021). SpConv: Spatially Sparse Convolution Library. https://github.com/traveller59/spconv. Accessed: 2023-12-28.

Yan, Y., Mao, Y., and Li, B. (2018). SECOND: Sparsely Embedded Convolutional Detection. *Sensors*, **18**.

Yang, B., Liang, M., and Urtasun, R. (2018a). HDNET: Exploiting HD Maps for 3D Object Detection. In *Conference on Robot Learning*.

Yang, B., Luo, W., and Urtasun, R. (2018b). PIXOR: Real-Time 3D Object Detection From Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Yang, H., Liu, Z., Wu, X., Wang, W., Qian, W., He, X., and Cai, D. (2022a). Graph R-CNN: Towards Accurate 3D Object Detection with Semantic-Decorated Local Graph. In *European Conference on Computer Vision*.

Yang, H., He, T., Liu, J., Chen, H., Wu, B., Lin, B., He, X., and Ouyang, W. (2023a). GD-MAE: Generative Decoder for MAE Pre-training on LiDAR Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Yang, H., Wang, W., Chen, M., Lin, B., He, T., Chen, H., He, X., and Ouyang, W. (2023b). PVT-SSD: Single-Stage 3D Object Detector With Point-Voxel Transformer. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Yang, Z., Sun, Y., Liu, S., Shen, X., and Jia, J. (2018c). IPOD: Intensive Point-Based Object Detector for Point Cloud. *arXiv:1812.05276*.

Yang, Z., Sun, Y., Liu, S., Shen, X., and Jia, J. (2019). STD: Sparse-to-Dense 3D Object Detector for Point Cloud. In *International Conference on Computer Vision*.

Yang, Z., Sun, Y., Liu, S., and Jia, J. (2020). 3DSSD: Point-based 3D Single Stage Object Detector. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Yang, Z., Zhou, Y., Chen, Z., and Ngiam, J. (2021). 3D-MAN: 3D Multi-frame Attention Network for Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Yang, Z., Li, Z., Jiang, X., Gong, Y., Yuan, Z., Zhao, D., and Yuan, C. (2022b). Focal and Global Knowledge distillation for Detectors. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Ye, M., Xu, S., and Cao, T. (2020). HVNet: Hybrid Voxel Network for LiDAR Based 3D Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Ye, S., Chen, D., Han, S., and Liao, J. (2021). Learning with Noisy Labels for Robust Point Cloud Segmentation. In *International Conference on Computer Vision*.

Yi, L., Kim, V. G., Ceylan, D., Shen, I.-C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., and Guibas, L. J. (2016). A Scalable Active Framework for Region Annotation in 3D Shape Collections. *ACM Transactions on Graphics*, **35**.

Yin, T., Zhou, X., and Krähenbühl, P. (2021a). Center-based 3D Object Detection and Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Yin, T., Zhou, X., and Krähenbühl, P. (2021b). Multimodal Virtual Point 3D Detection. In *Advances in Neural Information Processing Systems*.

Yoo, J. H., Kim, Y., Kim, J. S., and Choi, J. W. (2020). 3D-CVF: Generating Joint Camera and LiDAR Features Using Cross-View Spatial Feature Fusion for 3D Object Detection. In *European Conference on Computer Vision*.

Zagoruyko, S. and Komodakis, N. (2017). Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. In *International Conference on Learning Representations*.

Zaheer, M., Kottur, S., Ravanbhakhsh, S., Póczos, B., Salakhutdinov, R., and Smola, A. J. (2017). Deep Sets. In *Neural Information Processing Systems*.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., *et al.* (2020). Big Bird: Transformers for Longer Sequences. In *Neural Information Processing Systems*.

Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., and Funkhouser, T. (2017). 3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Zhang, F., Guan, C., Fang, J., Bai, S., Yang, R., Torr, P. H., and Prisacariu, V. (2020a). Instance Segmentation of LiDAR Point Clouds. In *IEEE International Conference on Robotics and Automation*.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). Mixup: Beyond Empirical Risk Minimization. In *International Conference on Learning Representations*.

Zhang, J., Wu, X., and Sheng, V. S. (2016). Learning from Crowdsourced Labeled Data: A Survey. *Artificial Intelligence Review*, **46**(4), 543–576.

Zhang, Y., Hu, Q., Xu, G., Ma, Y., Wan, J., and Guo, Y. (2022). Not All Points Are Equal: Learning Highly Efficient Point-based Detectors for 3D LiDAR Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Zhang, Z. and Sabuncu, M. R. (2018). Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. In *Neural Information Processing Systems*.

Zhang, Z., Sun, B., Yang, H., and Huang, Q. (2020b). H3DNet: 3D Object Detection Using Hybrid Geometric Primitives. In *European Conference on Computer Vision*.

Zhang, Z., Girdhar, R., Joulin, A., and Misra, I. (2021). Self-Supervised Pretraining of 3D Features on any Point-Cloud. In *International Conference on Computer Vision*.

Zhao, H., Jiang, L., Fu, C.-W., and Jia, J. (2019). PointWeb: Enhancing Local Neighborhood Features for Point Cloud Processing. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Zhao, H., Jia, J., and Koltun, V. (2020). Exploring Self-attention for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Zhao, H., Jiang, L., Jia, J., Torr, P. H., and Koltun, V. (2021). Point Transformer. In *International Conference on Computer Vision*.

Zheng, W., Tang, W., Chen, S., Jiang, L., and Fu, C.-W. (2021). CIA-SSD: Confident IoU-Aware Single-Stage Object Detector From Point Cloud. In *Conference on Artificial Intelligence*.

Zhou, X., Wang, D., and Krähenbühl, P. (2019a). Objects as Points. *arXiv:1904.07850*.

Zhou, X., Koltun, V., and Krähenbühl, P. (2021). Probabilistic Two-Stage Detection. *arXiv:2103.07461*.

Zhou, Y. and Tuzel, O. (2017). VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Zhou, Y., Sun, P., Zhang, Y., Anguelov, D., Gao, J., Ouyang, T., Guo, J., Ngiam, J., and Vasudevan, V. (2019b). End-to-End Multi-View Fusion for 3D Object Detection in LiDAR Point Clouds. In *Conference on Robot Learning*.

Zhou, Z., Zhao, X., Wang, Y., Wang, P., and Foroosh, H. (2022). CenterFormer: Center-based Transformer for 3D Object Detection. In *European Conference on Computer Vision*.

Zhu, B., Jiang, Z., Zhou, X., Li, Z., and Yu, G. (2019a). Class-balanced Grouping and Sampling for Point Cloud 3D Object Detection. *arXiv:1908.09492*.

Zhu, B., Wang, Z., Shi, S., Xu, H., Hong, L., and Li, H. (2023). ConQueR: Query Contrast Voxel-DETR for 3D Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Zhu, X., Wang, Y., Dai, J., Yuan, L., and Wei, Y. (2017a). Flow-Guided Feature Aggregation for Video Object Detection. In *International Conference on Computer Vision*.

Zhu, X., Dai, J., Yuan, L., and Wei, Y. (2017b). Towards High Performance Video Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Zhu, X., Ma, Y., Wang, T., Xu, Y., Shi, J., and Lin, D. (2020). SSN: Shape Signature Networks for Multi-class Object Detection from Point Clouds. In *European Conference on Computer Vision*.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., and Dai, J. (2021). Deformable DETR: Deformable Transformers for End-to-End Object Detection. In *International Conference on Learning Representations*.

Zhu, Y., Sapra, K., Reda, F. A., Shih, K. J., Newsam, S., Tao, A., and Catanzaro, B. (2019b). Improving Semantic Segmentation via Video Propagation and Label Relaxation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Zhu, Z. and Soricut, R. (2021). H-Transformer-1D: Fast One-Dimensional Hierarchical Attention for Sequences. In *Annual Meeting of the Association for Computational Linguistics*.