# Object Localisation for Scene Understanding in 3D


Sangyun Shin

Worcester College

University of Oxford

*A thesis submitted for the degree of Doctor of Philosophy*

Summer 2024

# Acknowledgements

I am very grateful for all the support I have received during my DPhil studies. Before starting the study, I did not know what I would face as a DPhil candidate, including both academic and non-academic challenges. Despite my previous research training, I found myself constantly confronted with new and unforeseen obstacles throughout my DPhil program. Many people have generously supported me in overcoming the challenges and enlightened me in my journey to become an independent researcher.

My sincere gratitude first goes to my supervisors, Professor Niki Trigoni and Professor Andrew Markham, who have been continuously guiding me since the beginning of this journey. Their approaches and insights on various research topics have profoundly shaped my ability to think like a researcher. They have provided advice regularly and always made time, whenever I asked, even during the pandemic, to shape my research, including high-level motivation, specific methods, and academic writing. Amidst the many uncertainties that I faced throughout my DPhil, one thing remains clear: I would not have reached this stage of completing my DPhil without their tremendous support.

Secondly, I am profoundly thankful to all the friends that I met who have been supporting me both academically and mentally in keeping me on track. I would like to thank Madhu Vankadari, Yuhang He, Kaichen Zhou, Stuart Golodetz, Aluna Everitt, Qian Xie, Tim Ta-Ying Cheng, and Vit Ruzicka.

Finally, my deepest gratitude goes to my families in South Korea and France for their unwavering support, which kept me refreshed and passionate throughout my DPhil journey. Their encouragement, love, and understanding helped me maintain a positive outlook and a balanced perspective, allowing me to stay motivated and focused on my research.

# Abstract

Understanding the surrounding physical world is key for the successful application of many intelligent systems, including self-driving cars and delivery robots. As a fundamental component towards this capability, object localisation has been widely studied using various sensors, particularly those operating in the optical regime such as cameras and LIDAR. The most prevalent form of object localisation is accomplished by detection and segmentation in a local sensor frame followed by tracking over multiple frames. Thanks to its structured nature, object localisation using this approach has seen impressive progress in image-based systems. However, for complex and dynamic scenes such as roads, 3D object localisation is still not yet sufficiently accurate for reliable real-world applications. Moreover, the continually changing nature of outdoor scenes requires object localisation systems to be robust and adaptive to dynamic environments and transferrable to different operating conditions.

This thesis explores the challenges of 3D object localisation based on visual sensors and presents methods that improve the aforementioned issues. Firstly, to improve the performance of 3D object localisation, we study current limitations in object representations based on Cartesian coordinates, e.g. axis-aligned bounding boxes. Based on this observation, we design a Spherical Mask that uses alternative object representations based on spherical coordinates for precise 3D object localisation. Secondly, to address the issue of high cost for 3D annotation of changing real-world environments, we develop an auto-annotation method for 3D object localisation. The method utilises motion cues from the sensors as signals for finding moving objects on various road scenes. These objects serve as pseudo annotations, which can be used to train standard object localisation models. We also show that using the Spherical representation is beneficial compared to the existing box representation for finding pseudo labels. Thirdly, we explore strategies to make the best use of existing annotations and pseudo annotations

acquired in the previous step. Specifically, to improve the robustness of the model in changing environments, a pseudo-label-guided clustering approach for domain adaptation is presented. Here, given the learned representation of each cluster centroid as prompt input, the localisation model's task is to continuously learn to localise objects similar to the given cluster, making the model focus more on pseudo labels, which is the centroid of the clusters. This strategy improves the robustness of the model as all existing labels are distributed to each cluster that focuses on pseudo labels. Each cluster is used by the model to find objects similar to the pseudo labels that belong to the cluster, in contrast to the existing approach that considers labels from the source environment and pseudo-labels from different environments as the same.

The proposed three techniques can be combined to deploy a more precise object localisation model, which can adapt to a continuously changing environment of the real world while minimising the cost caused by human-provided annotations based on pseudo-labels found using natural cues from the sensors.

# Contents

# Chapter 1

# Introduction

One of the essential building blocks for understanding dynamic scenes is estimating the position and pose of objects within them. Many important and emerging applications could benefit from this. For example, autonomous driving [90, 13] relies heavily on the precise localization of dynamic objects such as other vehicles, pedestrians, and cyclists to ensure safe and efficient operation. At a higher level of autonomy, behaviour monitoring requires object localisation, which involves continuously tracking and predicting the movements of objects to understand their intentions and actions. By accurately monitoring the behaviour of dynamic objects, a self-driving vehicle can make informed decisions to avoid potential accidents, such as slowing down for a pedestrian crossing the street or adjusting its path to avoid a suddenly appearing obstacle. Furthermore, this capability allows the vehicle to manoeuvre its speed and position optimally, contributing to smoother traffic flow and enhanced traffic control. Thus, effective localisation of objects is fundamental to achieving the safety and efficiency for autonomous driving technology.

Another emerging scenario requiring dynamic scene understanding is scene rendering in Virtual Reality (VR) and Augmented Reality (AR) [190], where dynamic objects and multiple real and virtual agents need to interact with each other simultaneously. In these immersive environments, accurate and real-time object localisation allows for the seamless interaction between virtual elements and the physical world. For example, in VR, virtual characters and objects must respond realistically to the user's movements and actions to maintain immersion, while in AR, virtual content must align accurately with real-world objects and respond to changes in the environment. This interaction demands precise object localisation using data from available sensors to continuously adapt to the dynamic scene, ensuring that all elements from both the virtual and real world interact in a coherent and realistic manner. Such capabilities are crucial for creating positive experiences

in VR and AR, where the quality of interaction directly impacts user engagement and satisfaction.

Another interesting application is the indoor service robots in hotels and restaurants and outdoor delivery robots using visual Simultaneous Localization and Mapping (SLAM), where accurately locating moving objects is essential for constructing reliable maps, especially given the typical assumption of a static scene. A robot must update a map of a dynamic environment while simultaneously determining its position within the map. This process becomes significantly more complex when dynamic objects, such as people and vehicles, are present because their movements introduce inaccuracies and ambiguities during the positioning. Therefore, distinguishing and accurately localizing the moving objects ensures that the robot based on the SLAM system differentiates between static features, which should be included in the map, and dynamic features, which should be tracked separately. This distinction is crucial for constructing the map based on the system's localization, enabling reliable navigation and interaction within the environment.

For safe and performant real-world deployment of the aforementioned applications, it is essential that the system can precisely localise objects in continuously changing environments due to weather, time of day, region, etc. The purpose of this thesis is to investigate the limitations of existing approaches for object localization and ways to address domain gaps caused by changing environments.

## 1.1 Motivation

The process of object localisation can be summarized as ego-localisation followed by object localisation in the local sensor frame as illustrated in Figure 1.1. More specifically, the agent or robot carrying sensors is tracked with odometry between sequential frames to acquire a global trajectory. In each frame, the objects are then found in the local sensor's frame and tracked by taking the global trajectory from odometry into account. Combined with powerful Deep Neural Networks (DNNs) and visual sensors, the popular forms of finding objects are the detection and segmentation in local sensor coordinates for each frame, which can both be directly used for tracking with data association techniques for continuous localization. Since accurate detection and segmentation in each frame leads to precise tracking, more studies have focused on detection and segmentation rather than tracking for object localisation.

Typically, detection and segmentation models are trained on a given dataset and tested on a given test set that has the same or a sufficiently similar environment

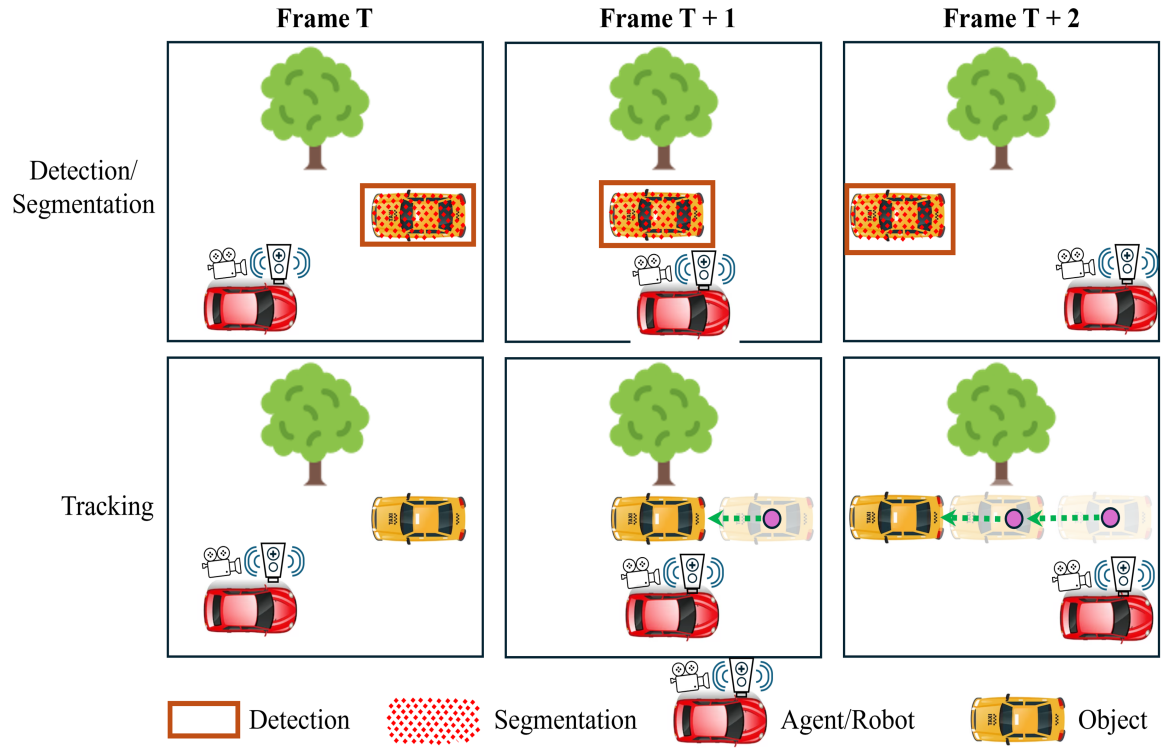



Figure 1.1: Object localisation in the forms of detection, segmentation, and tracking. In this thesis, the term localisation refers to measuring poses and dimensions of objects (yellow car) with respect to the sensor (red car) while tracking them in consecutive frames. Although the figure is drawn in 2D space for simplicity, this thesis focuses mainly on object localisation in 3D space.

as the training set. However, for real-world deployments where the environment changes naturally due to time, weather, and region, the adaptation ability of the object localisation model is essential. There have been many studies that focus on visual odometry with very high precision [1]. However, there has been comparably less progress in object localisation in terms of precision. This is largely because objects in point cloud can be perceived significantly different depending on their rotations and distance to the sensors. For example, objects that are further from the sensors naturally do not have much points on their surfaces, which causes insufficient features to determine if they are objects of interest, compared to the objects that are close to the sensors. On the other hand, visual odometry is not affected by sparse points on surfaces as the task only requires the static and reliable points on any surfaces.

Despite its importance for reliable industrial applications, 1) improving the precision and 2) robustness of object localisation against domain change, such as object size, different specifications of sensors, and weather change, has not yet been studied extensively. In this DPhil research, I aim to explore the problem of

localizing objects robustly against the common changes in daily life.

## 1.2 Open Challenges

Based on the motivation above, the challenges of object localisation can be summarized into three problems as follows.

### 1.2.1 Challenges in 3D Visual Object Localisation

Object localisation in the form of detection and segmentation has seen impressive progress with Deep Neural Networks (DNNs) in the last decade. However, compared to 2D localisation in image space, the performance of 3D localisation alone without multimodality is less reliable for real-world deployment, even in the same domain that the DNN model is trained on, as the detection accuracy could directly affect the safety. For example, the State-Of-The-Art (SOTA) method for the 3D detection is around 40% lower than that of the 2D detection in terms of the widely-used PASCAL metric [41] for detecting ***pedestrians*** in the same sequences of KITTI benchmark [48]. This is due to the characteristics of the input data, point clouds, that have an unordered and unstructured nature. For example, the appearance of an object could be perceived with significant variation depending on the viewing angle and location with respect to the sensors. The unstructured nature of the point cloud also poses challenges for learning the semantic/class information. A motivating and challenging scenario for this is the precise detection/segmentation of multiple visually similar objects that are very closely located.

### 1.2.2 Annotation Cost for Dynamic Environments

Unlike most available datasets with train sets and test sets, the real world naturally contains more diverse scenes depending on the weather, time of the day, season, and region. In fact, for 3D object localisation, studies [165, 184] show that the model's performance trained on one dataset drastically drops around 36% even for the same type of objects in a different environment. The straightforward way to fix the problem is to annotate a new dataset and train the model again whenever there is a change that causes domain gaps from the original training dataset. However, annotating the label, especially for 3D, is labor-expensive and inefficient due to far more complicated labels compared to 2D. For example, even with an advanced tool for 3D bounding box annotations, on average, it takes 98.98 minutes to label a data sequence recorded for 7 minutes and 12 seconds in a mid-sized city in KITTI

dataset [88, 48]. To avoid the issue, a promising direction to minimize the human-provided annotation is to use available cues that can be used to find objects. The found objects can then be used as pseudo-labels for the training of the object localisation model. Another potential direction would be to use foundation models, such as DINO [118] for pseudo-label generation from image to point cloud with calibrated sensors.

### 1.2.3 Continuous Learning for Object Localisation

For real-world deployment of the object localisation model, it is vital that the model reflects the inherent nature of the continuously changing real-world environments due to renovation, weather, variation of objects in size and shape, etc. In terms of the generalisation of the model, the challenge comes when certain dominant features in one environment are less common in other environments, which makes the model fail if it encounters different environments. For example, the average size of cars can significantly vary depending on the region [165], which was reported to cause significant performance drops when the model was deployed in a region other than the one that the model was trained. The common approach to address the issue is to collect pseudo-labels in the other environments and use them to retrain the model. However, typically, the number of pseudo-labels is much smaller than the existing labels due to the conservative pseudo-label collection strategy to reduce false positives. This creates an imbalance problem, where the pseudo-labels from the other environments get less attention during training as they are a minor group among existing labels. Straight-forward methods to address the issue, such as prioritising pseudo labels with weights, could cause another problem in that the model would forget the objects from the original environment. Therefore, a careful strategy to make the best use of pseudo-labels while preserving the knowledge from the existing labels is necessary.

## 1.3 Research Questions

Motivated by the challenges above, there are three main questions presented as part of this DPhil thesis.

1. **Can we improve the precision of current object localisation performance?** Unlike image-based localisation, object localisation in point-clouds imposes two challenges. First, point-clouds are unordered and sparse, which leads to high input variation even for the same object. This negatively affects the detection and segmentation of objects. Secondly, due to the binary-mask

based localisation for each instance, current algorithms experience a significant class-imbalance problem. For instance, the task of each binary mask is to classify foreground points as 1 and background points as 0. However, normally in existing datasets, the foreground points comprise only around 2% of the entire point cloud, which causes a performance drop even in the same domain.

2. **Can we acquire trainable annotation (pseudo annotation) without expensive human effort?** Changing the operating environment in point-cloud causes domain gaps. To minimize the gap, the current approach that produces the highest precision is by manually making a new ground truth set for the new environment and train the model again. However, doing so in point cloud datasets is a cumbersome and time-consuming task. An alternative approach that we investigate is to find objects undergoing motion. Typically, moving things, such as vehicles, pedestrians, and cyclists are important classes of objects that many applications are interested in. These objects could be segregated from their static surroundings by looking into distinct motions. However, finding the right motions from a large outdoor environment is challenging due to two reasons. Firstly, there are many motions that could cause false positives, such as leaves blowing in the wind. Secondly, there could be multiple objects showing similar motions, such as cars heading in the same direction on a highway, that could be grouped as one object.

3. **Can we improve the utilization of pseudo annotation with existing ground-truth for continual learning?** Many factors introduce domain gaps when point cloud-based object localisation systems are deployed to another domain. For instance, object size [165], different sensor specifications [123], and weather conditions [178] can all negatively impact the perception of the system. Since the features that models learn are geometric relationships such as distances between points and locations, these changes significantly impact the system's performance.

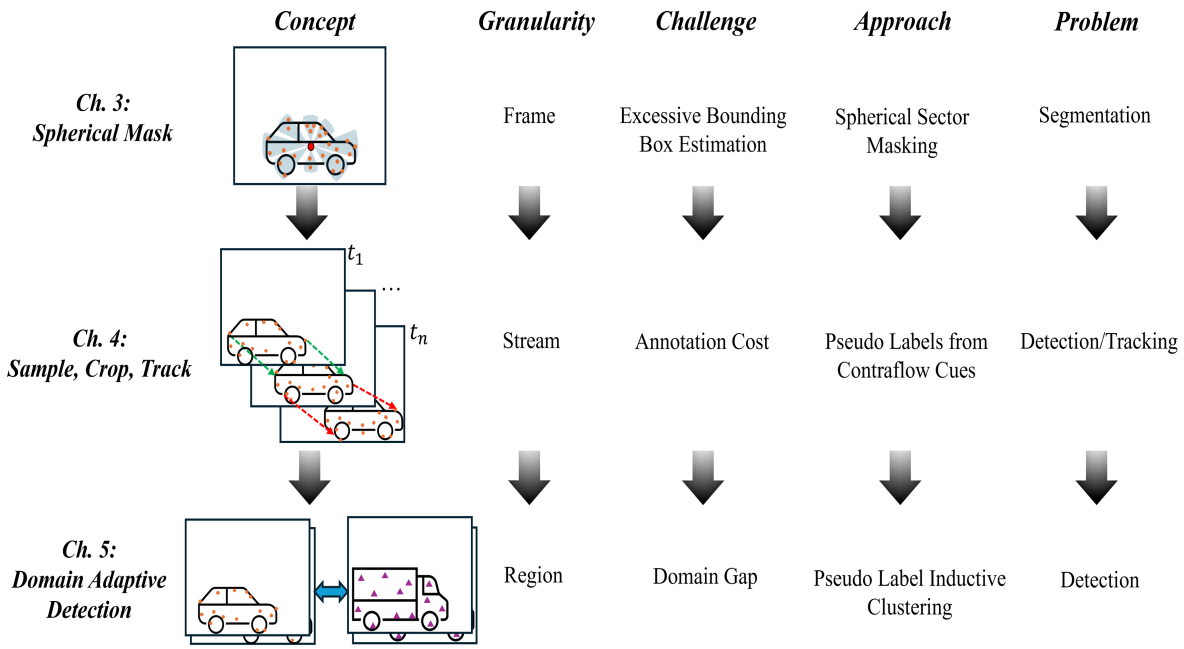



Figure 1.2: DPhil research roadmap.

## 1.4 Proposed Contributions

Based on the aforementioned questions, my DPhil research spans three main contributions. Figure 1.2 shows the roadmap of the three projects of my DPhil in accordance with the contributions. The detailed contributions are listed as follows.

1. In Chapter 3, a more precise object localization scheme based on spherical representation is introduced. This chapter specifically addresses the limitations associated with existing localization schemes that rely on Cartesian coordinates. The proposed spherical representation offers several advantages over Cartesian coordinates, particularly in the context of detection and segmentation. By leveraging the geometric properties of spherical coordinates, the proposed scheme enhances the accuracy of object localization that depends on precise spatial understanding, such as segmentation. Extensive experimental evaluations on ScanNet [31], S3DIS [3], and STPLS3D [18] datasets demonstrate the effectiveness of the proposed scheme in both indoor and outdoor environment for object localisation. Also, the code for the project is made public at: https://github.com/yunshin/SphericalMask. This contribution is in accordance with the first publication in Section 1.5.1.

2. In Chapter 4, a novel object localization method that does not require human-provided annotations is introduced. The method utilises cues from sensors to acquire annotations autonomously. More specifically, the method

leverages rigid motion-based cues and the expected size of objects to generate accurate pseudo annotations, significantly reducing the cost and effort associated with manual annotation. The rigid motion-based cue helps distinguish moving objects based on their movement, which violates the static scene assumption, while the expected object size helps ensure the objects from motion cues are the objects of interest. This automated approach facilitates continuous learning, allowing the system to adapt over time without the extensive labour required for manual annotation. This contribution corresponds to the second publication in Section 1.5.1.

3. In Chapter 5, a novel approach to optimize the use of existing annotations alongside newly generated pseudo annotations is introduced in the context of domain adaptation for continual learning. Specifically, this approach addresses the challenge of integrating pseudo-labels with the manually provided existing annotations and proposes a grouping strategy to improve the object localisation model in environments different from the original training data. A key aspect of this method is a weighted distribution mechanism to manage the available annotations, specifically designed to mitigate the issue of false negatives that can occur during domain adaptation. By assigning appropriate weights to each group of annotations based on their learned similarities, the method ensures that the learning algorithm can effectively discern and prioritize objects for changing environments. This strategy not only improves the accuracy and robustness of the model in new domains but also paves the way for continuous learning, allowing the system to adapt progressively to new data with minimal manual intervention. Extensive domain adaptation experiments on challenging outdoor datasets demonstrate the potential of the proposed strategy that effectively leverages both existing and pseudo annotations for object localisation. This contribution is consistent with the third publication in Section 1.5.1.

## 1.5 Publications

In accordance with the contributions, I have led three projects, resulting in publishable outcomes for each of them. Additionally, I have collaborated with colleagues in CPS group, which also resulted in publications. All publications during my DPhil and my contributions to each paper are summarized below.

### 1.5.1 Publications Related to the Thesis

1. **Publication:** Sangyun Shin, Kaichen Zhou, Madhu Vankadari, Andrew Markham and Niki Trigoni. "Spherical Mask: Coarse-to-Fine 3D Point Cloud Instance Segmentation with Spherical Representation." In Conference on Computer Vision and Pattern Recognition (CVPR), 2024.

   **Contribution:** This publication is in accordance with the first contribution mentioned in the previous section. I led the project as the leading author and actively carried out idea development, implementation, experiments and paper writing.

2. **Publication:** Sangyun Shin, Stuart Golodetz, Madhu Vankadari, Kaichen Zhou, Andrew Markham and Niki Trigoni. "Sample, Crop, Track: Self-Supervised Mobile 3D Object Detection for Urban Driving LiDAR." In International Conference on Robotics and Automation (ICRA), 2023.

   **Contribution:** This publication is in accordance with the second contribution mentioned in the previous section. As the main author, I participated in most processes of the project and contributed to idea development, implementation, experiments, and paper writing.

3. **Publication:** Sangyun Shin, Yuhang He, Madhu Vankadari, Andrew Markham and Niki Trigoni. "Towards Learning Group-Equivariant Features for Domain Adaptive 3D Detection." In Conference on Neural Information Processing Systems (NeurIPS), 2024.

   **Contribution:** This publication is based on the third contribution mentioned in the previous section. As the leading author, I actively participated in all project tasks, including idea development, implementation, experiments, and paper writing.

### 1.5.2 Additional Publications

4. **Publication:** Yuhang He, Sangyun Shin, Anoop Cherian, Niki Trigoni and Andrew Markham. "Sound3DVDet: 3D Sound Source Detection using Multiview Microphone Array and RGB Images." Winter Conference on Applications of Computer Vision (WACV) 2024

   **Contribution:** As the second author, I helped the leading author by implementing the calibration of multi-modal data, including microphone arrays and multiview cameras for sound source localization. Additionally, I also contributed to the paper writing and visualization of 3D figures in the paper.

5. **Publication:** Madhu Vankadari, Samuel Hodgson, Sangyun Shin, Kaichen Zhou, Andrew Markham and Niki Trigoni. "Dusk Till Dawn: Self-supervised Nighttime Stereo Depth Estimation using Visual Foundation Models." In International Conference on Robotics and Automation (ICRA), 2024

   **Contribution:** As the third author, I contributed to idea development, implementation of data preprocessing, visualisation, and paper writing.

6. **Publication:** Kaichen Zhou, Jia-Xing Zhong, Sangyun Shin, Kai Lu, Yiyuan Yang, Andrew Markham and Niki Trigoni, "DynPoint: Dynamic Neural Point For View Synthesis." in Conference on Neural Information Processing Systems (NeurIPS), 2023

   **Contribution:** As the third author, I mainly contributed to the idea development and paper writing.

7. **Publication:** Madhu Vankadari, Stuart Golodetz, Sourav Garg, Sangyun Shin, Andrew Markham and Niki Trigoni, "When the Sun Goes Down: Repairing Photometric Losses for All-Day Depth Estimation.", in Conference on Robot Learning (CoRL), 2022

   **Contribution:** As the fourth author, I contributed to the idea development and paper writing.

8. **Publication:** Stuart Golodetz, Madhu Vankadari, Aluna Everitt, Sangyun Shin, Andrew Markham and Niki Trigoni, "Real-Time Hybrid Mapping of Populated Indoor Scenes using a Low-Cost Monocular UAV.", in International Conference on Intelligent Robots and Systems (IROS), 2022

   **Contribution**: As the fourth author, I actively participated in idea development, data collection, and paper writing.

## 1.6 Summary

The outline of this thesis is as follows.

Chapter 2 provides a comprehensive overview of the background knowledge essential for understanding the subsequent chapters. It covers the basic concepts and methodologies of object localisation that underpin research and innovations discussed later in the text.

Chapter 3 introduces a novel spherical representation-based method aimed at enhancing object localization. This new approach addresses and overcomes the

limitations of Cartesian coordinate-based methods, which have inherent disadvantages due to the use of a bounding box.

Chapter 4 explores a self-supervised approach to obtaining annotations for training object localization models, reducing reliance on labour-intensive human-provided annotations. This method leverages motion cues from sensor inputs, such as rigid motion patterns and expected object sizes, to autonomously generate annotations.

Chapter 5 presents a novel domain adaptation strategy designed to enhance object localization. This method effectively integrates existing annotated data and newly generated pseudo-annotations to adapt the model to different domains, addressing the variability and challenges inherent in changing environments.

Chapter 6 provides a comprehensive summary of the research findings of this thesis, highlighting the contributions and advancements made throughout the study. It also outlines potential directions for future research, suggesting areas where further investigation could build upon the current work, and exploring new applications of the developed methods.

# Chapter 2

# Background

This chapter firstly reviews the visual sensors that are commonly used for object localisation, followed by outlining key tasks, namely: detection, segmentation, and tracking.

## 2.1 Common Sensors and Configurations

The most popular visual sensors for object localisation are camera and LiDAR. Typically they are attached to a robot/agent as can be seen in Figure 1.1 The usage of these sensor perceptions as input for object localisation is broadly divided into three categories: (1) Camera-based object localisation, (2) LiDAR-based object localisation, and (3) Camera-LiDAR-based localisation.

### 2.1.1 Camera-based System

Object localisation for camera-based systems has a long history in the computer vision community with machine learning algorithms. Over the last decade, DNNs have boosted the performance of object localisation significantly for challenging scenarios. In this setting, the RGB image from camera, $I \in \mathbb{R}^{H \times W \times 3}$, serves as input for DNN models that localise objects in image space. In very recent studies, multi-view cameras with input $\{I_1, I_2, ..., I_n\}$ are also used to localise objects in a reference camera coordinate (3D) using epipolar constraint [167]. However, the performance of multi-view object localisation methods in comparison to range sensors is limited for outdoor scenarios due to the limited baseline.

### 2.1.2 Range-based System

One of the most widely used range sensors for 3D object localisation in the road scene is LiDAR. With the different specs, including beam density and field of view,

LiDAR directly creates a point cloud with precise $(x, y, z)$, $P \in \mathbb{R}^{N_p \times 3}$, that can be used as an input for DNNs designed for object localisation. Typically, multiple scans over a time window are accumulated to form an input point cloud due to the point sparsity. The goal of object localisation in this setting is to find the 3D location of objects with respect to the range sensor.

### 2.1.3 Camera-range Combined System

An emerging sensor configuration is to combine multi-view cameras and LiDARs as complementary signals. Cameras carry colour information that has been proven to be effective in image-based localization but lack depth information. In contrast, LiDARs lack colour information, although they directly produce point clouds with rich 3D information. In order to combine sensors to create complementary signals, calibrated intrinsic $K \in \mathbb{R}^{4\times4}$ and extrinsic parameters $T \in \mathbb{R}^{4\times4}$ are used for the following procedure.

Firstly, the point cloud from LiDAR, $P_{LiDAR}$, is projected into the camera coordinate frame, $P_{Camera}$ using extrinsic $T^{Camera}_{LiDAR}$ as:

$$P_{Camera} = T^{LiDAR}_{Camera} P_{LiDAR}, \tag{2.1}$$

where $T^{LiDAR}_{Camera}$ is given as:

$$T = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \tag{2.2}$$

Here, $R \in \mathbb{R}^{3\times3}$ and $t \in \mathbb{R}^{1\times3}$ are rotation and translation matrices representing the extrinsic parameters for the camera and the LiDAR. A point $p_{Camera} = (x, y, z, 1)^T$ in $P_{Camera}$ can be then projected to image space to acquire its pixel position $(u, v)$ using the intrinsic matrix $K$ as:

$$(u, v, 1) = K p_{Camera}, \tag{2.3}$$

where $K$ is given as:

$$K = \begin{pmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.4}$$

Here, $(f_u, f_v)$ and $(c_u, c_v)$ stand for focal lengths and the principal point in horizontal and vertical directions, respectively. Finally, using $(u, v)$, the RGB value corresponding to $p_{Camera}$ can be acquired and concatenated to construct $p_{CamLidar}$ as:

$$p_{CamLiDAR} = (x, y, z, r, g, b) \tag{2.5}$$

By applying the same procedure for all points in $P_{Camera}$, we acquire $P_{CamLiDAR}$, which is a point cloud with colour information. In many recent works, $P_{CamLiDAR}$

is used directly as input for DNNs to make the model learn the colour information together with geometric information from the point cloud.

### 2.1.4 Coordinate Systems

This section gives a summary of the two dominant coordinate systems that can be used in the application of 3D object localization.

**Cartesian Coordinate System** The Cartesian coordinate system is on the the most commonly used in object localisation. It is used for describing spatial positions of points in a 3D space. This system consists of three perpendicular axes (x, y, z) that are intersecting at the origin, which is noted as (0, 0, 0). Each of the points present in this system is identified in a unique way, representing its distances from the origin along each axis. The Cartesian coordinate system is used and is predominant in many other fields that are heavily based on 3D space, such as the robotics and computer graphics domains.

**Spherical Coordinate System** The spherical coordinate system is another approach to representing points in 3D space. Instead of describing a point's position using (x, y, z) as in the Cartesian system, the spherical system uses a radial distance ($r$) from a central origin, a polar angle ($\theta$) measured from a reference axis, and an azimuthal angle ($\phi$) measured in the x-y plane from a reference direction. This coordinate system is widely used in fields that involve radial symmetry or spherical objects, as it simplifies the mathematical representation and calculations of spatial relationships and motions, such as in physics, astronomy, etc. Compared to Cartesian coordinates, Spherical coordinates offer several advantages, particularly in scenarios where radial or angular relationships are important. For example, a point in 3D space can be often more intuitively described by measuring distances from a central point.

## 2.2 Object Detection

Given an input from a visual sensor, object detection aims at estimating boxes that bound or enclose an object's perimeter. The process of detection can be divided into two categories, depending on the input type, namely image and point cloud. This section will review the common pipeline depending on the input data type, such as image and point cloud, and introduce existing works for each of them.

### 2.2.1 Image-based Detection System

Object detection based on images has a long and significant history. In this subsection, we will review the common pipeline shared in image-based detection systems, and review existing works depending on their task, namely 2D or 3D detection.

**Pipeline** The common pipeline of image-based detection systems consists of (1) Feature extraction, (2) Regional Proposals, and (3) Post Processing. The detailed description for each step is as follows.

1. *Feature Extraction*: Given an input image, an encoder extracts spatial features. Popular encoder architectures typically consist of multiple layers of Convolutional Neural Networks (CNN) to extract and group spatial and hierarchical features from low-level edges and textures to high-level object parts.

2. *Regional Proposals:* Regional proposals [134, 100, 131] identify candidate regions within an image that are likely to contain objects, thus focusing computational resources on these promising areas. The proposals are typically generated by using DNN layers over the feature map to predict 'objectness' scores and bounding box coordinates. Typically, predefined anchor boxes for each proposal help regression of object sizes and aspect ratios. More specifically, each proposal is classified for an object class and refined with the predicted offset to form final box predictions $B$ as:

$$B = (cx + \Delta cx, cy + \Delta cy, w + \Delta w, h + \Delta h) \tag{2.6}$$

3. *Post Processing (Non-Maximum Suppression):* Given the refined proposals from the previous step, Non-Maximum Suppression (NMS) [115] is applied to eliminate redundant bounding boxes. Specifically, NMS eliminates multiple overlapping bounding boxes for the same object from the regional proposals. NMS addresses this by first sorting these boxes based on their confidence scores. Starting with the highest-scoring box, NMS suppresses all other boxes that have a high overlap, measured by Intersection over Union (IoU) with the selected box. This process iterates until all boxes have been considered. The result is a reduced set of bounding boxes, each representing a unique object, improving the precision of the detection system by eliminating redundant and false-positive detections.

**Recent Works on 2D Object Detection** The pioneering works on object detection in images are Faster RCNN [134], Yolo [131], and SSD [100]. Their differences are based on how they utilize RPN. Faster R-CNN is a two-stage detector that first

generates region proposals using RPN and then classifies these proposals, offering high accuracy and precision but at the cost of slower processing speeds. YOLO (You Only Look Once) is a single-stage detector that treats object detection as a regression problem, predicting bounding boxes and class probabilities directly from the entire image in one pass. Specifically, the image is divided into pre-defined grids, where each grid serves as a proposal. This eliminates the need for sequential regional proposal prediction from faster RCNN, which makes it exceptionally fast and suitable for real-time applications. However, there exists a trade-off between accuracy and speed, particularly for smaller objects. SSD (Single Shot MultiBox Detector) also employs a single-stage approach, using multiple feature maps at different scales to detect objects of various sizes, balancing speed and accuracy effectively. While SSD is faster than Faster R-CNN and generally more accurate than YOLO, it provides a good compromise between the two in terms of speed and detection performance. Later, YOLO-V2 [132] utilizes Batch Normalization and a fully convolutional network to improve the execution speed. YOLO-V3 [133] replaces the backbone with a more powerful backbone, DarkNet 53, to improve the performance. From this, there have been many variants of YOLO-V3, such as YOLO-V4 [7], YOLO-V5 [75], PP-YOLO [108], PP-YOLO-V2 [63]. The most recent one is YOLO-V11 [76], which optimizes the training process and proposes trainable bag-of-freebies methods with a dynamic label assignment strategy.

Similar to YOLO, SSD also produces a line of research that alleviates its limitations. The limitations are two-fold: (1) The context information across different levels of feature maps is not fully utilized. (2) Only a small portion of the generated anchor boxes are positioned close to the actual object, resulting in a class imbalance between positive and negative samples during the training process. To address the first problem, feature fusion approaches across different levels of feature maps are proposed in DSSD [44], R-SSD [69], ESSD [200], FSSD [182]. The second problem is addressed by using focal loss [97] that increases the weight of hard samples, sparse prediction model in RefineDet [195], and bottom-up feature pyramids to guide the refinement of anchors in EFGRNet [116].

More recently, instead of using anchors for RPN, some works propose strategies to use anchor-free RPN. The limitations that these works address are twofold: 1) The design of hyperparameters, including the size and aspect ratio of anchor boxes, significantly influences the performance of the detection model. 2) The positive anchor boxes face issues with positioning ambiguity and background feature interference.

To address these issues, CornerNet [87] proposes to represent a prediction box using sparse key corners. Based on this, MatrixNet [130] and CenterNet [36] improved speed and key point-based box representation using heatmap, respectively.

The latest trend in image-based object detection harnesses the powerful Transformer architecture. Since the pioneering work DETR [14] was proposed, many studies have been published boosting the performance of object detection. Dynamic DETR [32] simulates the encoder to dynamically adjust the attention more effectively. Deformable DETR [206] introduces multi-scale attention, which improves the small object detection with the trade-off between performance and complexity of the model. Sparse DETR [136] improves the inference speed by input token sampling based on the scoring network. Anchor DETR [166] revisits the anchor and utilizes anchor points with queries to reduce the complexity of learning. Dense Distinct Queries (DDQ) [196] addresses the low recall problem of Transformer-based method by laying dense queries like traditional detectors and then selecting distinct ones for assignments. Light DETR [91] significantly reduces the complexity of the detection head while keeping the performance reasonable by an innovative encoder block that interleavely updates high-level and low-level features. Co-DETR [208] addresses the issues with query assignment. Namely, too few query assignments in DETR during one-to-one set matching leads to sparse supervision which significantly hurts the learning of the detector. Focusing more on the efficiency side of DETR, RT-DETR [199] introduces an efficient hybrid encoder that swiftly processes multi-scale features by separating intra-scale interaction from cross-scale fusion to enhance speed. Rank-DETR [124] proposes rank-oriented alignment for confidence score and localisation to overcome the performance drop due to misalignment between confidence score and localisation in DETR. Following studies, such as Salience DETR [60] and Ease DETR [47], also focus on improving the query assignment issue.

**Recent Works on 3D Object Detection** A pioneering method for image-based 3D object detection is Mono3D [20]. Mono3D achieves the 3D detection on monocular images by the proposal generation, which distributes positions object candidates with a probabilistic model in 3D based on a ground plane prior. Each candidate box can be projected onto the image plane to be scored using contextual information, such as size and location priors, and typical object shape. Roddick et al. [135] point out that the reliance on image-based features makes it difficult to infer 3D information, such as depth, in Mono3D. To address this, they introduce the orthographic feature transform, which maps image-based features into an orthographic 3D space, allowing for a consistent scale and meaningful distances between objects, which is integrated into an end-to-end deep learning architecture.

Mousavian et al. [113] alter the network architecture, where the initial network predicts the object orientation utilizing a novel discrete-continuous loss, which is more effective than L2 loss for the training. The subsequent layer estimates the 3D object dimensions, benefiting from relatively low variance and allowing for accurate predictions across various object types. Lastly, combining the sequential predictions with the geometric constraints from the 2D bounding box produces a 3D bounding box. MonoPSR [81] leverages object proposals and shape reconstruction based on pinhole camera model to project 2D detection into 3D space. Additionally, for each object proposal, an object-centered point cloud is generated to provide local context, such as scale and shape. SMOKE [105] introduces a single-stage method that predicts a 3D bounding box for each object by combining a keypoint and predictions to create a 3D bounding box.

Recently, multi-view-based 3D object detection methods have shown promising direction. As one of the pioneering works, DETR3D [167] extracts 2D features from multi-view camera images and utilizes a set of 3D object queries to fuse into image features. This approach effectively fuses 3D features, such as positions, and image features from multi-view images through camera transformation matrices. Specifically, the model first predicts bounding boxes for each object query, employing set-to-set criteria that the object features from multi-view images are very similar to each other, where the image features can be acquired by projecting the 3D position of the object into each image space using camera transformation matrices. The loss function then penalizes the discrepancy between ground truth and predictions. Inspired by this, PETR [102] introduces 3D coordinate generator, which is similar to a voxel-grid, to better encode 3D features with image features, which is called 3D position-aware features. This approach makes PETR more effective and efficient for learning. PETRv2 [103] incorporates temporal information from previous frames to improve detection performance. It extends the 3D position-aware features proposed in PETR to better temporally aligned features across frames. PETRv2 also demonstrates the effectiveness of the temporally aligned features by multi-task learning, such as segmentation and 3D lane detection.

Image-based object detection systems leverage advanced neural network architectures to accurately and efficiently identify objects. By combining CNNs, region proposal mechanisms, classification, and localization processes, these systems achieve robust performances in various real-world applications.

### 2.2.2 Point Cloud Object Detection

Compared to image-based object detection, 3D object detection has received attention only recently due to the more recent development and availability of 3D sensors that generate point clouds, such as LiDAR, stereo cameras, depth cameras, and radar. Unlike traditional 2D object detection, which only captures information in image space, 3D object detection provides a more comprehensive understanding by incorporating depth (z-axis) information. This added dimension is crucial for applications requiring precise spatial awareness and interaction with the environment, such as autonomous driving, augmented reality, robotics, and surveillance systems. For 3D detection, the bounding box is defined as $(cx, cy, cz, w, h, l, rx, ry, rz)$, where $(cx, cy, cz)$ are the 3D centre of the object, $(w, h, l)$ are dimensions of bounding box in width, height, and length in the Cartesian space. Lastly, $(rx, ry, rz)$ refers to the 3D rotation of the object.

**Pipeline** Pipeline-wise, the difference between 2D detection and 3D detection is only in preprocessing and feature extraction. The pipeline is as follows.

1. *Preprocessing:* Due to the unordered and unstructured nature, a point cloud typically requires preprocessing, such as voxelization. Voxelization converts a sparse and irregular point cloud into a structured grid representation known as a voxel grid. Here, each voxel in this grid represents a small volume of space, similar to how a pixel represents a small area in the image space. This structured representation facilitates efficient processing and analysis using 3D CNNs and other advanced architectures of DNNs.

2. *Feature Extraction:* There are two DNNs based models that are used for extracting features from voxels. (1) 3D convolution. (2) Sparse convolution.

   3D convolution [112] is an extension of the 2D convolution operation commonly used in image processing, and it plays a crucial role in processing volumetric data such as video sequences, medical imaging scans, and voxel grids derived from point clouds. In 3D convolution, the filter or kernel moves through three dimensions (depth, height, and width) to capture spatial features across all three axes.

   Sparse convolution [30] is a specialized form of convolution designed to efficiently process sparse data structures, particularly in the context of 3D point clouds and volumetric data. Unlike traditional dense convolutions, which operate on uniformly structured data, sparse convolutions focus on optimizing operations where the data is mostly empty or has a high degree of sparsity. This approach is crucial in applications such as LiDAR point

cloud processing, where the data points are distributed sparsely across a large volume.

The extracted features from either 3D convolution or sparse convolution are compressed into Birds Eye's View (BEV) map, which can take advantage of the same existing architecture, such as regional proposals and post-processing, that are used for classification and object localization in the image-based detection systems.

**Recent Works** 3D Object Detection within point cloud data can be categorized into a) point-based methods, which are input-wise permutation invariant, and b) voxel-based methods, which utilize grid representations. Figure 2.1 shows the representative works.

• **Voxel-based Methods:** As a pioneering work, VoxelNet [204] eliminates the need for manual feature engineering for 3D point clouds. VoxelNet partitions the point cloud into evenly spaced 3D voxels and converts groups of points within each voxel into a unified feature representation using the newly introduced voxel feature encoding (VFE) layer. This approach encodes the point cloud into a descriptive volumetric representation, which is then connected to an RPN to generate detections. Following VoxelNet, SECOND [180] explores an enhanced sparse convolution method for these networks, which substantially boosts the speed of both training and inference. Additionally, SECOND introduces a novel angle loss regression technique to improve orientation estimation performance and a new data augmentation strategy that enhances convergence speed and overall performance. Later, PointPillars [85] combines point-based and voxel-based methods. Initially, the point cloud is divided into grids in the x-y coordinates, forming a set of pillars. Each point in the cloud (x, y, z, reflectance) is transformed from a 4D vector into a 9D vector by extending it with the distance to the arithmetic mean of the pillar and the distance to the centre of the pillar using PointNet [128]. Each pillar in 2D space is then represented by a structured (fixed-dimension) vector. The rationale behind using 2D voxelization grids is that they are more computationally efficient than directly using 3D voxelization grids. Additionally, the 2D voxelization grids enable the usage of advanced RPN architectures that are proposed for 2D detection. Voxel-RCNN [33] argues that the precise positioning of raw points is not crucial for achieving high-performance 3D object detection and that using coarser voxel granularity can still provide adequate detection accuracy, proposing an effective voxel-based framework that leverages voxel features in a two-stage process. VoxelNeXt [23] introduces fully sparse networks that directly predict objects using sparse voxel features, eliminating the need for hand-crafted proxies. This robust

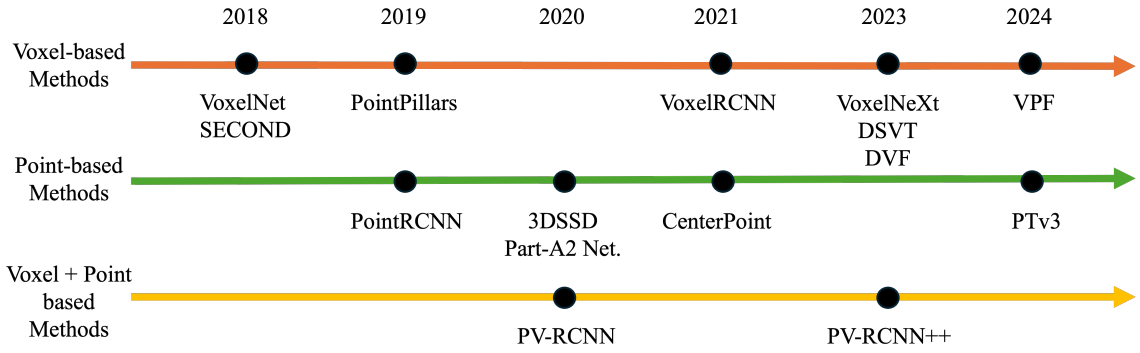



Figure 2.1: Representative works on point cloud object detection in chronological order.

sparse convolutional network detects and tracks 3D objects solely through voxel features without requiring sparse-to-dense conversion or NMS post-processing. Similarly, to efficiently handle sparse points, DSVT [161] proposes Dynamic Sparse Window Attention, which divides each window into a series of local regions based on their sparsity. Dense Voxel Fusion (DVF) [111] introduces a camera-lidar fusion method that produces dense voxel representation from two modalities, improving the expressiveness of the two visual sensors. Voxel-Pillar Fusion (VPF) [64] leverages the complementary advantages of both voxels and pillars. Specifically, a sparse voxel-pillar encoder that transforms point clouds into voxel and pillar features using 3D and 2D sparse convolutions is introduced. The encoded features are then fed to Sparse Fusion Layer (SFL) to enable bidirectional interaction between the sparse voxel and pillar features.

• **Point-based Methods:** PointRCNN [149] pioneered the raw point cloud-based object detection, utilizing the two-stage framework, where the first stage focuses on bottom-up 3D proposal generation, and the second stage refines the proposals in canonical coordinates for the final box prediction. 3DSSD [185], (3D Single Shot Detection) enhances inference speed while maintaining the accuracy of two-stage detectors without relying on voxel usage. This approach involves key points sampling and extracting features using set abstraction modules from raw point-based 3D detectors, such as PointNet. Specifically, the point sampling method utilizes a feature distance metric alongside the Euclidean distance for key point sampling, improving the recall of points and balancing foreground and background points. Part-A2 Net. [150] extends PointRCNN and incorporates part awareness layer into a two-stage pipeline: the part-aware stage and the part aggregation stage. In the part-aware stage, free part supervisions from 3D ground-truth boxes are fully utilized to predict 3D proposals along with intra-object part locations simultaneously. These predicted intra-object part locations within the

same proposal are grouped using a RoI-aware point cloud pooling module, which creates an effective representation to encode the geometry-specific features of each 3D proposal. In the subsequent part-aggregation stage, the model learns to re-score and refine the box location by analyzing the spatial relationship between the pooled intra-object parts. Despite the high performance due to fine-grained learning of parts, one of the limitations of Part-A2 Net is expensive computation. Using different formulations of box prediction, CenterPoint [189] introduces a new heatmap-based regional proposal in a two-stage framework. Specifically, after extracting features from the input point cloud, the resulting BEV feature map is fed into a 2D CNN detection head, producing a heatmap, where peaks indicate the detected objects' centre locations. The corresponding size, orientation, and velocity of the objects are estimated using different regression heads. Due to the rotational-invariant nature of points, the network achieves higher accuracy in predicting rotated objects compared to anchor-based detectors. More recently, Point Transformer V3 (PTv3) [172] recognizes that model performance is more significantly impacted by scaling than by intricate mechanisms. Based on the observation, PTv3 focuses on simplicity and efficiency, replacing precise neighbour search methods like KNN with a more efficient serialized neighbour mapping of point clouds organized in specific patterns without significant loss in accuracy.

Rather than focusing only on voxel or point, PVRCNN [146] aims to combine both approaches. Initially, a small set of key points is sampled from the raw point cloud to represent the entire scene. Simultaneously, the voxels are encoded into the key points. These fused features of key points are then fed into RPN for final box prediction. PVRCNN++ [148] extends the previous method by improving the sampling strategy of the key point. Specifically, in the new sampling strategy, the key points are constrained to be around 3D proposals to acquire coherent features for regional proposals. PolarFormer|[73] introduces polar coordinates-based detection that is adaptable to various input structures, effectively handling irregular Polar grids. Specifically, it implements a multi-scale Polar representation learning strategy in the polar grid. Due to the invariant features brought by the 2D polar grid, PolarFormer demonstrates its robustness in dealing with diverse and irregular input data. Far3D [72] addresses the problem of long-range 3D detection by introducing adaptive queries from high-quality 2D object priors. To effectively learn robust features across various views and scales, especially for objects located further away from the sensor, Far3D proposes a perspective-aware aggregation module. PARQ [176] employs appearance-enhanced queries, which are initialized from 3D reference points, similar to anchors. The reference locations are then updated with recurrent cross-attention operations. MvACon [101] points out

that previous methods struggle with either low-resolution 2D features due to high computational demands or inadequately grounding 3D queries to multi-scale 2D features.

**Relevance to the thesis:** Chapters 4 and 5 of this thesis employ the advancement made in the existing works. Specifically, the research in Chapters 4 and 5 for point cloud object detection also adopt voxelization and BEV feature map for the regional proposal networks used in aforementioned works [149, 185, 150, 189, 148].

### 2.2.3 Dataset for Point Cloud Object Detection

Several datasets are available for public benchmark of 3D object detection. Among them, the most widely used datasets are KITTI, NuScenes, and Waymo datasets.

**KITTI 3D Object Detection Benchmark** KITTI 3D Object Detection Dataset is a component of the KITTI Vision Benchmark Suite, designed for 3D perception in autonomous driving. The data is collected from a car equipped with multiple sensors, including cameras, a Velodyne LiDAR scanner, and GPS, driving in Germany. The dataset consists of 7,481 labeled samples for training and 7,518 samples for testing, with comprehensive annotations for objects, such ascars, pedestrians, and cyclists.

Each labeled sample provides 3D bounding boxes, object class labels, occlusion and truncation levels. The annotations are designed to evaluate object detection in three levels of difficulty: Easy, Moderate, and Hard, based on factors like object size, distance, and occlusion.

KITTI's benchmark evaluates algorithms using mean Average Precision (mAP), a widely used metric in detection tasks. It focuses on the accuracy of localizing and classifying objects in 3D space using data from LiDAR, cameras, or both. The KITTI dataset has become one of the most popular benchmarks in point cloud object detection.

**NuScenes 3D Object Detection Benchmark** NuScenes 3D Object Detection Dataset is a large-scale dataset designed for 3D perception in autonomous driving. The dataset contains 1,000 driving scenes, each lasting 20 seconds, for a total of 1.4M annotated frames. Each scene includes synchronized data from multiple sensors, including six cameras (360° coverage), a LiDAR scanner (360° point clouds), five radars, GPS, and an inertial measurement unit (IMU) captured in urban environments of Boston and Singapore.

Annotations cover 3D bounding boxes for 23 object classes, such as cars, pedestrians, bicycles, and traffic cones. The dataset also introduces tracking IDs to facil-

itate object tracking across frames and provides information about environmental conditions, such as time of day and weather.

The benchmark evaluates methods on mean Average Precision (mAP) and NuScenes Detection Score (NDS), which incorporates additional factors for velocity and orientation estimation.

**Waymo 3D Object Detection Benchmark** Waymo Open Dataset is a large-scale benchmark designed for 3D perception, including object detection in autonomous driving. It provides diverse and high-quality sensor data collected from vehicles operating in urban, suburban, and highway settings across the United States. The dataset includes over 1,150 driving segments, each 20 seconds long, with synchronized data from four LiDAR sensors (360° point clouds) and five high-resolution cameras (panoramic images). It offers detailed annotations for five object classes—vehicles, pedestrians, cyclists, signs, and other road users—with 3D bounding boxes, velocities, and tracking IDs for motion analysis. Performance is evaluated using metrics, mAP and mAP with heading (mAPH), with difficulty levels based on object visibility and LiDAR point density. With its rich multimodal data and large-scale annotations, Waymo dataset has become a key resource for training and benchmarking state-of-the-art 3D object detection models.

### 2.2.4 Summary

In this section, we reviewed the image-based and point cloud-based object detection systems. Compared to image-based 2D detection systems, 3D detection systems' performances are relatively weak due to the more challenging requirement for 3D pose and size estimation. In 3D detection systems, multi-view camera-based detection systems and point cloud-based detection systems are widely used. Although multi-view camera-based detection systems have gained growing attention, they still struggle to precisely detect objects further away due to the lack of 3D information. On the other hand, point cloud-based detection systems can take advantage of precise 3D information about the surroundings. However, due to their unordered and unstructured nature, they face challenges in terms of generalisation.

## 2.3 Object Segmentation

Object segmentation aims to find objects in a local sensor frame from the same sensors as detection. However, compared to detection, segmentation produces fine-grained localization of objects by classifying every pixel for image or point for

point cloud. Due to its nature of seeking high precision, instance segmentation is considered a more challenging learning task for object localization compared to detection using bounding boxes. There are two subcategories of segmentation: 1) Semantic segmentation and 2) Instance (panoptic) segmentation.

The goal of semantic segmentation is to classify every pixel or point into given categories. Although it is powerful to understand the types of objects in the scene, it is not suitable for object localization. For example, semantic segmentation cannot distinguish individual objects if they belong to the same category. On the other hand, instance (panoptic) segmentation focuses on finding binary masks for each object, where the foreground parts that belong to the object surface are segmented as 1, and the background that does not belong to the object is 0. For image-based instance segmentation, each binary mask has the same dimension as the image, and for 3D instance segmentation, it is typically processed with a point cloud, where each mask also has the same dimension as the point cloud.

### 2.3.1 Image-based Instance Segmentation

The process of image instance segmentation involves several stages that are different depending on the method. The most commonly used pipeline is listed as follows.

**Pipeline** The general pipeline for image-based instance segmentation is as follows.

1. *Feature Extraction:* Similar to detection, an input image is fed into a backbone network for extracting high-level features. The resulting feature maps capture essential spatial information and patterns that are crucial for identifying objects in the later stages.

2. *Region Proposal:* The extracted feature maps are then processed by a Region Proposal Network (RPN) from the detection, which generates a set of candidate object proposals. These proposals are potential locations within the image where objects might be present.

3. *Object Detection:* Each proposed region from the RPN is refined to predict the bounding boxes and classify the objects within them as object detection. Here, the purpose of box prediction is to provide a rough boundary of the object, so that the segmentation layer could focus on the small box rather than the entire image.

4. *Region of Interest (RoI) Pooling:* The detected boxes with various sizes need to be processed to have uniform size, as DNNs layer can only accept the

fixed-size input matrix. This is achieved with RoI pooling, which extracts fixed-size feature maps from the variable-sized RoIs generated by the RPN. More advanced techniques, such as RoI Align, could improve performance by preserving spatial information during the pooling process.

5. *Mask Generation:* Given extracted feature maps from RoI pooling, the segmentation network proceeds to generate pixel-wise binary masks for each box [54]. A segmentation head, typically consisting of several convolutional and deconvolutional (upsampling) layers, is employed to produce these masks. Each mask indicates the exact shape and boundaries of the object within the corresponding RoI, differentiating it from the background and other objects.

6. *Post-Processing:* Similar to detection, post-processing involves applying NMS to remove the redundant segmentation by retaining only the most confident ones, followed by thresholding with the prediction confidence score.

**Existing Works** As a pioneering work, Mask R-CNN [54] enhances Faster R-CNN by introducing an additional branch for predicting object masks alongside the existing bounding box recognition branch. This extension maintains simplicity in training and incurs minimal overhead, which can be easily adapted to other tasks, such as human pose estimation, within the same framework. Following Mask R-CNN, Mask scoring RCNN [65] introduces a network block designed to learn the quality of predicted instance masks. This block regresses the mask IoU by taking both the instance feature and the predicted mask. The mask scoring strategy corrects the misalignment between mask quality and score. BMask RCNN [27] leverages object boundary information to enhance mask localization accuracy. More specifically, it includes a boundary-preserving mask head where object boundaries and masks are mutually learned through feature fusion blocks. This approach ensures that the predicted masks are more accurately aligned with the object boundaries, improving overall segmentation performance. DCT-Mask [143] claims that low-resolution grids from Mask-RCNN lack detail and introduces the discrete cosine transform (DCT) to encode high-resolution binary grid masks into compact vectors while maintaining efficiency. PANet [99] enhances the feature hierarchy with localization signals in lower layers using path augmentation, preserving the information from the lowest to topmost features. For this, adaptive feature pooling is proposed to connect the feature grid across all levels of features and to directly share informative features at each level for subsequent proposal networks. Additionally, a complementary branch is created to capture different views for each proposal, further refining mask predictions. These improvements are straightforward

to implement and introduce minimal computational overhead. HTC [17] leverages the reciprocal relationship between detection and segmentation to adopt powerful Cascade architecture into instance segmentation. It includes a fully convolutional branch to provide spatial context, aiding in distinguishing difficult foregrounds from cluttered backgrounds. PointRend [78] adopts a concept of rendering from computer graphics methods to address the issue of over- and undersampling in segmentation tasks. It performs point-wise predictions at adaptively chosen locations using an iterative subdivision algorithm. RefineMask the problem of feature loss during downsampling in the feature pyramid and instance-wise pooling, which is particularly negative for large objects. To overcome the problem, RefineMask [193] utilizes fine-grained features with a multi-stage, where each stage progressively fuses detailed information to continuously refine high-quality masks. DynaMask addresses the problem of both low-resolution masks and high-resolution masks: Low-resolution masks lack detail, and high-resolution masks require significantly more computation than low-resolution masks. To address this issue, DynaMask introduces a dynamic selection of optimal mask resolutions for different object proposals. Specifically, Feature Pyramid Network (FPN) is combined with a region-level top-down path (r-FPN) to learn contextual and detailed deep features from various stages of the image-level FPN (i-FPN).

### 2.3.2 Point Cloud Instance Segmentation

**Pipeline** Applying instance segmentation for point clouds is similar to 3D detection. Given the point cloud from sensors, existing works typically follow four steps.

1. *Preprocessing:* Given a point cloud as input, voxelization is optionally applied to convert the irregular point cloud into a structured 3D grid of voxels similar to point cloud detection. Although some methods based on direct point processing argue that voxelization can lead to loss of detail due to discretization, most of the state-of-the-art methods on point cloud instance segmentation apply voxelization.

2. *Feature Extraction:* Once the point cloud is preprocessed, the next step is feature extraction. This involves using neural network architectures [128, 30] designed to handle 3D data, such as the set abstraction based on PointNet, or more recent models like Transformer. These networks extract geometric and contextual features from the point cloud by processing individual points and sub-sampling.

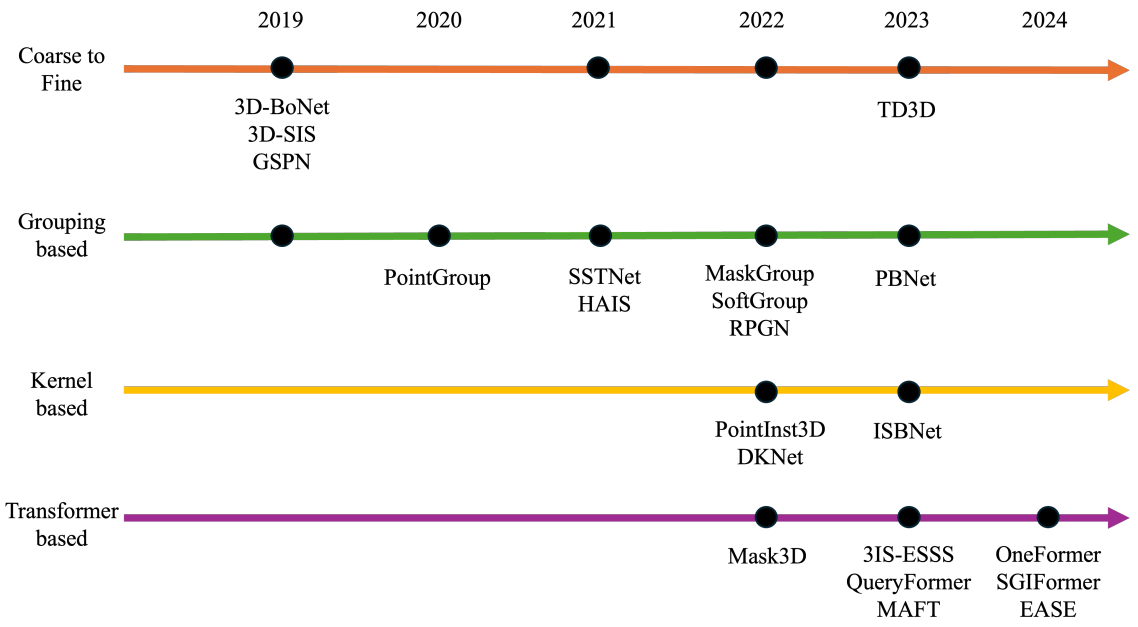



Figure 2.2: Representative works on point cloud instance segmentation in chronological order.

3. *Instance Segmentation and Classification:* Each extracted feature is fed into segmentation and classification heads to produce binary masks and object class. For binary mask generation, every point is classified into 1 for the foreground or 0 for the background.

4. *Post-Processing:* The initial segmentation output often includes overlapping or redundant regions as similar to bounding boxes in detection. Post-processing steps require NMS to eliminate duplicate detections and retain the most confident predictions. Additional steps, such as smoothing and refinement, are optionally performed to improve the quality of the segmentation masks, ensuring accurate and clean boundaries between objects.

**Recent Works** Depending on how the binary mask for each object is generated, point cloud instance segmentation can be broadly categorized into four: (1) Coarse to Fine (2) Grouping (3) Kernel (4) Transformer, as illustrated in Figure 2.2.

Following similar work, Mask RCNN, in image instance segmentation, pioneering works on point cloud instance segmentation adopt the coarse-to-fine approach. S3DIS [59] jointly learns both geometric and color signals harnessing high-resolution RGB input by linking 2D images with a volumetric grid through pose alignment of the 3D reconstruction. As another pioneering work, 3D-BoNet [181] directly predicts 3D bounding boxes for all instances in a point cloud and simultaneously generates a point-level mask for each instance by using two parallel heads:

one for bounding box regression and another for point mask prediction. Although it is conceptually single-stage, joint learning of boxes enforces the mask prediction to be aligned inside the box. Instead of using bounding box as coarse representation, GSPN [187] adopts an analysis-by-synthesis method, generating proposals by reconstructing shapes from noisy scene observations called Region-based PointNet (R-PointNet), which facilitates flexible proposal refinement and instance segmentation generation. TD3D [79] introduces a fully sparse-convolutional method for point cloud instance segmentation. Similar to Mask-RCNN, this approach begins by detecting Axis-Aligned Bounding Boxes (AABBs) within the point cloud data. Once these bounding boxes are identified, TD3D extracts per-point features from within these boxes. These extracted features are then used to perform binary classification, distinguishing foreground and background points inside the boxes.

Grouping-based methods focus on learning latent embeddings of points to facilitate detailed per-point predictions. Utilizing these embeddings, the model performs per-point predictions to assign semantic categories, such as identifying whether a point belongs to different object classes. Clustering algorithms are then applied to these points, grouping those with similar embeddings together to form binary instance masks. Typically, due to the integration of latent embeddings with per-point semantic predictions and clustering, grouping-based methods outperform coarse to fine methods with comparably expensive computational costs. As a pioneering work on grouping, PointGroup [71] first attracts points to their instance's centroid by predicting offsets for each point, which creates a shifted version of the original point cloud. The shifted point clouds are then utilized together to form clusters. The rationale behind this approach is that the shifted points will be closer to their respective object centres, making it easier to group points that belong to the same object. Built upon this concept, HAIS [19] addresses the issue of over-segmentation or under-segmentation of the grouping method and introduces hierarchical aggregation that progressively creates instance proposals. This involves initially aggregating points to form initial clusters and then combining these clusters to output complete instance masks. SSTNet [95] introduces a novel intermediate structure called the semantic superpoint tree, which is built using learned semantic features of superpoints [34]. By adopting superpoints, the complexities of learning point cloud are reduced significantly. Additionally, SSTNet includes a refinement module named CliqueNet, which prunes the superpoints. SoftGroup [160] combines bottom-up soft grouping with top-down refinement. Specifically, it allows each point to be associated with multiple classes, reducing the impact of hard semantic prediction errors. PBNet [198] adopts a divide-and-conquer strategy that binarizes each point before using them for clustering. The

binary clustering categorizes each points into high-density points (HPs), which are typically important, and low-density points (LPs), which can cause noise. Specifically, the first stage only groups HPs and LPs are assigned to HPs using a neighbour voting method for group refinement.

Kernal-based approaches are built on DyCo3D. DyCo3D [56] learns to generate suitable convolution kernels based on the characteristics of the instances. In this setting, given the per-point feature of the point cloud, instances are decoded using several simple convolutional layers. For the decoder, a lightweight transformer is employed to overcome the limited receptive field introduced by sparse convolution. PointInst3D [57] introduces a fully-convolutional method for per-point predictions. The key insight of the method is target assignment for the foreground points using the optimal transport approach. DKNet [173] facilitates Dynamic Convolution for mask inference, eliminating the need for proposals or heuristic clustering algorithms. Specifically, it introduces a sampling strategy that aggregates duplicates and gathers contextual information around the merged centroids to form instance kernels. These instance kernels then enable the reconstruction of instance masks through dynamic convolutions.

Mask 3D [141] introduces the pioneering Transformer-based approach for point cloud instance segmentation, leveraging generic Transformer building blocks to directly predict instance masks from instance queries. The transformer learns the instance by iteratively attending to point cloud features across multiple scales, enabling accurate and efficient 3D instance segmentation. Following Mask3D, QueryFormer [109] employs a novel query initialization module that ensures a high coverage and low repetition rate. Additionally, a decoder that suppresses the interference of background points is introduced to help foreground queries concentrate on the discriminative parts of instances. Similarly MAFT [82] addresses the problem of query initialisation with low recall that leads to slow convergence. To address this, MAFT replace the mask attention with an auxiliary centre regression task that can prioritize certain important queries.

**Relevance to the thesis:** This thesis compares the existing works on coarse-to-fine, grouping, kernel, and transformer based approaches for point cloud instance segmentation. In particular, Chapter 3 focuses on the coarse-to-fine approach proposed in [59, 181, 79] to improve the performance while comparing with other approaches.

### 2.3.3 Datasets for Point Cloud Instance Segmentation

**ScanNet V2** ScanNet Instance Segmentation Dataset is a large-scale benchmark for 3D instance segmentation tasks, focusing on indoor environments. It is part of the broader ScanNet dataset, which contains over 1,500 indoor scene reconstructions captured from RGB-D video sequences in real-world settings, such as homes, offices, and classrooms. The dataset provides richly annotated 3D point clouds with instance-level segmentation labels for 20 semantic categories, including furniture, appliances, and structural elements, such as walls and ceilings. Each instance is assigned a unique ID, enabling precise object segmentation within scenes. The dataset has high diversity in terms of environment and instances, making it a popular benchmark for developing and evaluating models for 3D perception and autonomous systems operating in indoor environments. The evaluation uses mAP to assess segmentation quality.

**Stanford Large-Scale 3D Indoor Scenes (S3DIS)** S3DIS dataset is a benchmark for 3D point cloud instance segmentation and semantic segmentation tasks, focusing on indoor environments. It consists of detailed 3D scans of 6 large-scale indoor areas with 271 rooms, such as offices, conference rooms, and lobbies, captured using a RGB-D scanner. The dataset contains 215 million points, each labeled with one of 13 semantic categories, such as walls, chairs, tables, and floors, along with instance-level annotations to differentiate individual objects of the same class.

The S3DIS dataset represents complex, cluttered indoor spaces with diverse furniture arrangements and architectural structures, making it ideal for evaluating models' ability to handle realistic 3D environments. Its annotations include geometric, spatial, and semantic details, allowing for robust learning and evaluation of 3D perception models. Evaluation metrics are mean Intersection over Union (mIoU) and mAP for assessing segmentation quality.

**3D Aerial Photogrammetry Point Cloud Dataset (STPLS3D)** STPLS3D dataset is a benchmark designed, for instance segmentation and semantic segmentation of large-scale 3D point clouds, focusing on urban and peri-urban environments. It is created using photo-realistic synthetic aerial photogrammetry for accurate and dense 3D point clouds. STPLS3D covers 16 $km^2$ of landscapes and up to 18 fine-grained semantic categories and instance-level labels for various urban objects such as buildings, roads, vehicles, and infrastructure. The dataset contains diverse geographic locations, offering a wide range of urban layouts, densities, and architectural styles. STPLS3D enables robust benchmarking of 3D instance segmentation models, particularly for applications involving aerial 3D data. The evaluation metrics use mAP and mIoU.

### 2.3.4 Summary

In this section, we reviewed the various instance segmentation methods based on their input type, such as image or point cloud. Due to the more challenging requirement for localisation, it is typically considered a more difficult task than object detection. Although great progress has been made, making binary masks for each object has a class imbalance problem.

## 2.4 Object Tracking with Motion

This section reviews basic techniques for flow-based motion tracking between frames. As one of the fundamental topics in computer vision, many works have been published on flow-based tracking. Since we just adopted existing techniques for optical-flow in the contributing work of this thesis, we highlight a few representative works on flow methods, namely optical and scene flows, followed by a more in-depth review of how flow-based motion tracking is used for self-supervised detection.

**Optical Flow** Optical flow estimates the motion in a visual scene based on the movement of pixels between consecutive frames. In particular, optical flow is the velocity vector for each pixel, capturing the direction and speed of motion. It is essential for various applications such as video stabilization, motion detection, object tracking, and autonomous navigation. It is a widely studied field where techniques range from traditional Lucas-Kanade to modern DNN-based methods [68, 164, 66]. By providing detailed motion information, optical flow enhances the understanding and analysis of dynamic scenes, enabling more robust and precise vision-based applications.

**Scene Flow** Scene flow extends the concept of optical flow to 3D scenes, providing a detailed representation of the motion of every point in 3D over time. Unlike optical flow, which only captures 2D motion in the image plane, scene flow encompasses both the 2D motion on the image plane and the depth changes of each point, resulting in a comprehensive 3D motion vector field. Scene flow is particularly valuable in applications requiring a precise understanding of dynamic environments, such as autonomous driving, augmented reality, and robotic navigation. Due to its high potential for many applications, scene flow has been one of the popular fields that are making impressive progress with DNN [169, 67, 157, 144, 203].

**Recent Works on Self-Supervised Object Detection with Tracking** One of the promising ways to find objects without human-provided annotation is to use mo-

tion cues. Assuming the rigid motion, existing works address the self-supervised approach for finding objects, such as vehicles, to reduce the cost of human-provided annotations for changing environments. The found objects are then used as pseudo-labels to train the object localisation model. Track, Check, Repeat [53] introduces the Expectation Maximization (EM) approach for finding pseudo labels. More specifically, the method starts with motion cues to segment objects by comparing optical flow and camera motion to find regions that move independently to the background. These initial regions serve as pseudo-labels to train an ensemble of 2D and 3D object detection models with extensive data augmentation. The heavy data augmentation enables the detection of new instances of "moving" objects, even if they are stationary. These newly detected objects are merged as new pseudo-labels. The method operates as an expectation-maximization algorithm: in the expectation step, all modules are activated, and their agreement is assessed; in the maximization step, the modules are re-trained to enhance this agreement. Similarly, LISO [6] addresses trajectory-regularized self-training. The method leverages a self-supervised lidar scene flow network to generate, track, and iteratively refine pseudo-labels in a similar manner as Track, Check, Repeat. This approach allows the object detection network to learn effectively from the pseudo-labeled data, enhancing its performance without the need for manually annotated training sets. Their integration of trajectory regularization improves the reliability of pseudo labels. HyperMODEST [177] aims to speed up the multi-stage framework proposed by Track, Check, Repeat by focusing on pseudo labels. Specifically, it tries to filter out intermediate pseudo-labels used for data augmentation by discarding those with low confidence scores. This filtering process ensures that only high-quality, reliable pseudo-labels are used, which in turn improves the overall accuracy and efficiency of the training process. Another LiDAR scene flow-based method [39] first trains a self-supervised scene flow estimation model using cycle consistency. The 3D detection model embraces the motion representations from the scene flow estimation model to effectively find dynamic objects based on their movement patterns, which can used as pseudo labels.

### 2.4.1 Summary

Using flow-based tracking between frames has become the main approach for finding pseudo labels. In principle, the moving object can be found easily by looking at the motions that violate the egomotion between frames. The remaining challenge is how to cluster independent motions existing in the frame. Also, the performance inevitably depends largely on the performance of flow estimation.

## 2.5 Domain Adaptive Object Detection

Domain adaptive object localisation is an emerging field in object localisation that aims to enhance robustness across different domains, reflecting the real world. It addresses the issue of performance degradation of the object localisation model when applied to new domains that differ from the training data in terms of sensor types, environmental conditions, or scene characteristics. Specifically, the domain adaptation techniques focus on strategies to generalize better to the new environments without the need for extensive manual labelling [184, 183, 61].

The process typically involves transferring knowledge from a source domain, where labelled data is available, to a target domain without labels. This can be achieved through various methods such as adversarial training [45], where the model learns to minimize discrepancies between the domains, and self-training [184], where the model iteratively refines its predictions using pseudo-labels generated in the target domain. Additionally, feature alignment techniques [16] are used to make the feature distributions of the source and target domains more similar, enhancing the model's ability to recognize objects in both domains.

**Recent Works** In the context of object localisation in 3D, one of the first works that address the performance degradation due to the inter-domain gap is SN [165]. SN first observes that datasets are collected from a limited number of cities within one country and under similar weather conditions to address the challenge of adapting 3D object detectors from one dataset to another. It identifies that one of the primary challenges during the adaptation is differences in object sizes across geographic regions. This observation leads them to propose a size normalization technique, which artificially normalizes the size of objects in the training set to be similar to the target domain. Similarly, 3D Contrastive Cotraining (3D-CoCo) focuses on the inter-domain gap caused by varying physical environments or different LiDAR sensor configurations. Specifically, it addressed the challenge of learning transferable features between a labelled source domain and a novel target domain without target labels. 3D-CoCo argue that bird-eye-view (BEV) features are more transferable than low-level geometry features, as they could contain more robust features that are less domain-specific than other features. As a result, it introduces a co-training scheme with separate 3D encoders having domain-specific parameters and a BEV feature for learning domain-invariant features. Additionally, 3D-CoCo extends contrastive instance alignment to point cloud detection, mitigating performance issues caused by the mismatch between pseudo-label-induced BEV feature distributions and true distributions of the target domain.

Along with 3D-CoCo, one of the pioneering works that currently dominates the field of domain adaptive object localisation is the self-training-based method. ST3D [184] begins by pre-training the 3D detector on the source domain using augmentations, such as random object scaling, to mitigate the potential bias from the source domain. The pre-trained detection model is adapted to the target domain by iteratively refining the detector through two steps: (1) updating pseudo labels with a quality-aware triplet memory bank (2) training the model with curriculum data augmentation. The iterative two-step strategy ensures that the detector is trained with consistent and high-quality pseudo labels while preventing overfitting to the abundance of easy examples in the pseudo-label set. ST3D++ [183] extends ST3D by focusing on noisy gradient directions during self-training and prevention of overfitting caused by noisy pseudo-labeled data. GPA3D [94] leverages the intrinsic geometric relationships of objects in point cloud objects to minimize feature discrepancies, enhancing cross-domain transferability. Specifically, GPA3D assigns learnable prototypes to point cloud objects with different representative geometric structures. These prototypes align bird's-eye-view (BEV) features from corresponding point cloud objects in both source and target domains, reducing distributional discrepancies. REDB [24] addresses the issue of a significant performance drop in multi-class adaptation settings due to low-quality pseudo labels and class imbalance issues. To address this, REDB introduces the concept of cross-domain examination to evaluate the quality of pseudo labels, where the pseudo labels from the target domain are copied to the source domain to check prediction consistency. Additionally, a class-balanced augmentation for both pseudo and source labels is introduced to alleviate the class-imbalance problem. DTS [62] focuses on the point density-induced inter-domain gap and introduces the density-insensitive domain adaptation framework. Random Beam Re-Sampling (RBRS) is used during the pre-training phase in the source domain to improve the robustness of 3D detectors by varying LiDAR beam densities. Using the pre-trained detector as the backbone, a task-specific teacher-student framework is employed to process unlabeled target domain data and generate high-quality pseudo labels. To further instil density insensitivity in the target domain, the teacher and student networks both take the same sample at different densities as input while enforcing consistency in output.

### 2.5.1 Summary

In this section, we reviewed various domain adaptive detection methods. Currently, a dominant approach is self-training, where the pseudo-labels are itera-

tively collected in multi-stage training. One remaining challenge is to effectively distribute continuously accumulating pseudo-labels with respect to existing labels. This is essential for generalising the detection model without an imbalance between new pseudo-labels and existing labels.

## 2.6 Foundation Models

Recently, large models trained on the immense amount of data became available, tackling the problems that were previously considered very challenging. The models typically serve as a layer for downstream tasks, allowing the fine-tuning or adaptation for specific purposes with minimal additional data. Examples include large language models for natural language processing and computer vision. These models leverage transformers and are typically trained using self-supervised learning.

### 2.6.1 Foundation Models for 2D

Foundation models for 2D are primarily designed for images, but also applicable to other forms of 2D data, such as text rendered as images or 2D spatial data. These models are typically pretrained on massive datasets of images and related annotations, enabling them to generalize across a wide range of 2D visual tasks. The notable foundation models that can be applied to object localizations are DINO series [15, 118] and Segment Anything (SAM) [77].

DINO leverages vision transformers (ViT) and focuses on learning dense, semantically meaningful features for each pixel or patch in an image with self-supervised learning. Without explicit labels, DINO V2 shows powerful performance in zero-shot and few-shot settings, making it a robust choice for real-world applications. Its ability to generalize across tasks stems from its strong emphasis on aligning features semantically. By utilizing the highly generalized feature from DINO, the downstream detection and segmentation tasks can be more robust against domain changes.

SAM is capable of segmenting any object in an image, guided by prompts such as points, bounding boxes, or freeform text, making it highly versatile and interactive. It is trained on a massive dataset containing billions of image-mask pairs, enabling it to handle a wide range of objects, textures, and contexts with strong performance, even in zero-shot scenarios. It can significantly reduce the need for manual annotation, making it a powerful tool for object detection and segmentation.

### 2.6.2 Foundation Models for Point Cloud

The progress of foundation models for image has been significant recently. However, achieving similar success in 3D models are still ongoing problem due to challenges in point cloud such as non-unified data formats and the scarcity of labeled data with diverse masks. Although there are no models that are generally used popularly like DINO and SAM for image, recent works have tried to introduce initial versions of foundation models for point cloud.

Following the interactive prompt based segmentation ability of SAM for image, Point-SAM [205] leverages part-level and object-level annotations for training from a mixture of existing datasets. In order to generate more data for training, Point SAM uses SAM for performing segmentation in images and projects the results to 3D using structure from motion to get the pseudo labels.

Seal [104] leverages 2D-to-3D representation distillation. More specifically, Seal first extracts superpixels, which are the visually similar regions in the image, using foundation models for image and apply contrastive learning to transfer the features from superpixels to corresponding points in the point cloud found by reprojection.

### 2.6.3 Summary

This section reviewed the representative foundation models related to object localization. Foundation models have become pivotal tools for object localization tasks in images. Although the progress of foundation models on point cloud is behind the performance of the foundation models for image, they still have a great potential for domain generalization.

# Chapter 3

# Spherical Mask: Coarse-to-Fine 3D Point Cloud Instance Segmentation with Spherical Representation

This chapter addresses how to improve the existing object (instance) segmentation. We first start by introducing the limitations of the existing coarse-to-fine approach and then motivate the proposed approach based on the spherical coordinates. The advantages and disadvantages of other approaches are also addressed.

Coarse-to-fine 3D instance segmentation methods show weak performances compared to recent Grouping-based, Kernel-based and Transformer-based methods. We argue that this is due to two limitations: 1) Instance size overestimation by axis-aligned bounding box(AABB) 2) False negative error accumulation from inaccurate box to the refinement phase. In this section of the thesis, we introduce **Spherical Mask**, a novel coarse-to-fine approach based on spherical representation, overcoming those two limitations with several benefits. Specifically, our coarse detection estimates each instance with a 3D polygon using centre and radial distance predictions, which avoids excessive size estimation of AABB. To cut the error propagation in the existing coarse-to-fine approaches, we virtually migrate points based on the polygon, allowing all foreground points, including false negatives, to be refined. During inference, the proposal and point migration modules run in parallel and are assembled to form binary masks of instances. We also introduce two margin-based losses for the point migration to enforce corrections for the false positives/negatives and cohesion of foreground points, significantly improving the performance. Experimental results from three datasets, such as ScanNetV2, S3DIS, and STPLS3D, show that our proposed method outperforms existing works, demonstrating the effectiveness of the new instance representation with spherical coordinates. The code is available at https://github.com/yunshin/SphericalMask.

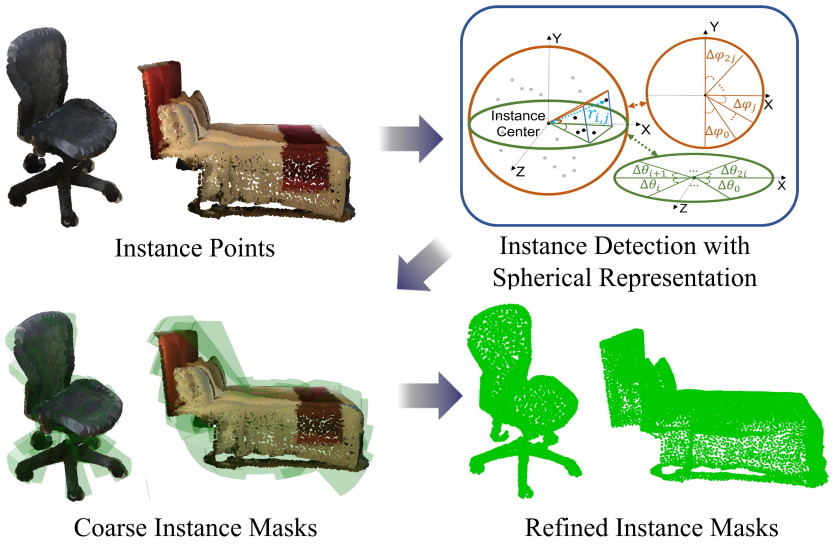



Figure 3.1: Pipeline of Spherical Mask with coarse-to-fine framework. Given point cloud, instances are detected with 3D polygons defined in spherical coordinates. In the refinement phase, the points virtually migrate based on the polygon to estimate fine instance masks.

## 3.1 Introduction

3D instance segmentation has gained immense attention with its wide range of applications for Indoor Scanning[202], Augmented Reality(AR)[89], and Autonomous Driving[121]. Similar to 2D instance segmentation, the goal of the task is to identify each object along with its class label. Nevertheless, the sparse and unordered nature of point clouds has led to the development of methods different from 2D image segmentation.

Existing approaches for 3D instance segmentation are broadly categorized into coarse-to-fine based[181, 187, 98, 59, 79], grouping-based[71, 19, 95, 160, 198], kernel-based[56, 71, 57, 173, 158], and Transformer-based[141, 152, 82, 109, 2] approaches. Recent progress in clustering techniques and attention mechanisms have driven the performances of the last three approaches to state-of-the-art. Compared to these three approaches, the coarse-to-fine approach has received relatively less attention due to low accuracy, which is caused by two limitations: 1) False negative error propagation from the coarse detection to the refinement stage. 2) Overestimation of instance size.

Coarse-to-fine instance segmentation first performs coarse detection, followed by refinement using coarse detection as a hard reference. The basic assumption of the approach is that the coarse detection stage always provides a neat detection for refinement. However, this assumption is often violated as the coarse detection stage cannot always produce neat outputs, which causes an issue of upper bound accuracy. For example, if the first coarse detection does not include

all foreground points, the following refinement(binary classification) step has no means to include them, only possibly accumulating the error from false negative prediction. The other limitation is that the axis-aligned-bounding-box(AABB) estimation, which is typically used for coarse detection, has been claimed to be an ill-defined problem[160] because commonly used box regression losses(L1, L2) result in overestimation of object sizes. For instance, the target values of AABB are minimum and maximum values in x,y, and z cartesian coordinates, making the box include redundant empty space, as points only lie on the surface of the object.

In this section of the thesis, we address the aforementioned two limitations of coarse-to-fine instance segmentation. Our core intuition comes from the fact that the weakness of the coarse-to-fine approach is based on the structural disentanglement of coarse detection and fine refinement phase. Instead of assuming that the coarse part should be perfect, we take a relaxation approach, which regards the coarse detection as a soft reference during the refinement phase, providing more access for refinement yet restricting access to unnecessary background points. Specifically, to improve the coarse detection part, we estimate a 3D polygon in spherical coordinates instead of AABB, alleviating the issue of excessive object size estimation. To remove the error propagation from inaccurate coarse detection to the refinement stage, we virtually migrate points based on the 3D polygon with reduced complexity in spherical coordinates.

In summary, our method **Spherical Mask** finds each instance by estimating a 3D polygon fitting to an instance in spherical coordinates and migrating points inside or outside the polygon to produce the fine instance mask. Our contributions are:

- We introduce a new alternative instance representation based on spherical coordinates, which overcomes the limitations in existing coarse-to-fine approaches.
- To circumvent the issue of excessive estimation of instance size, we propose **Radial Instance Detection**(**RID**) that formulates an instance into a 3D polygon as a coarse detection.
- To cut the error propagation from coarse detection to the refinement phase, we introduce **Radial Point Migration(RPM)**, capable of refining both false positive and false negative points from RID.
- Extensive experiments on ScanNetV2 [31], S3DIS [3], and STPLS3D [18] show the effectiveness of our approach, pushing the boundary of current SOTA.

## 3.2 Related Work

Existing works on point cloud instance segmentation can be categorized into proposal-based, clustering-based, kernel-based and transformer-based methods.

### 3.2.1 Coarse-to-Fine (Proposal-based) Approach

Coarse-to-fine-based methods are built on a conceptually simple design, where they first perform coarse detection, followed by a refinement stage to acquire fine segmentation. Typically, the 3D bounding box is employed for coarse detection. 3D-SIS[59] performs instance segmentation by first detecting the 3D boxes and refining points inside. 3D-BoNet[181] matches query AABBs and ground-truth instance using Hungarian algorithm for the supervision. The predicted AABBs are then concatenated with point features to produce binary masks for each instance. GSPN[187] adopts set-abstraction[128] to get query points and infer AABBs. The features inside the AABBs are extracted and used for per-point mask segmentation. More recently, TD3D [79] proposes a fully sparse-convolutional approach for point instance segmentation. It first detects AABBs and extracts perpoint features inside the boxes to perform binary classification. Most of the works use coarse detection(bounding box) directly as geometric features for predicting per-point binary instance masks. Thus, their accuracies greatly depend on the precision of the coarse detection, as inaccurate detection could easily lead to a large number of false negative points.

### 3.2.2 Grouping-based Approach

Grouping-based methods learn latent embeddings to perform per-point predictions, such as semantic categories and clustering to acquire instances. PointGroup[71] predicts centroid offsets of each point and utilizes this shifted point cloud and original point cloud to obtain the clusters. Based on this concept, many studies improve the clustering technique with hierarchical intra-instance predictions[19], superpoint-based divisive grouping[95], soft grouping[160], and binary clustering[198]. The clustering-based methods have high expectations of the quality of per-point centre prediction in 3D, which is challenging to generalize with various spatial extents of objects. There is a similarity between grouping-based approaches and coarse-to-fine approaches that the proposed work in this thesis focuses on. For example, grouping based methods also first predict the first proposal of groups, which is similar to the corase detection step, followed by the refinement. However, in the refinement step, grouping based approach requires

access to individual points, which is less efficient compared to box prediction with the fixed number of parameters in coarse-to-fine approach.

### 3.2.3 Kernel-based Approach

Kernel-based methods learn convolution kernels that aggregate point features to estimate instance masks. DyCo3D[56] proposes discriminative kernels by applying the clustering method from PointGroup[71]. Built on this, more recent works improved the performance by replacing the clustering part with farthest-point sampling[57], candidate localization[173], and instance-aware point sampling for the high-recall[158]. In fact, learning kernel can be generally applied to improve the regression performance without sacrificing the efficiency in terms of execution time. In the proposed Spherical Mask, we also employ the dynamic convolution proposed in DyCo3D for the refinement step.

### 3.2.4 Transformer-based Approach

Recently, transformer-based methods have set the new SOTA. Based on mask-attention[26, 25], Mask3D[141] and SPFormer[152] present the pipeline that learns to output instance prediction directly from a fixed number of object queries from voxel and superpoint features, respectively. Based on these works, recent studies improve the performance with auxiliary centre regression[82], query initialization and set grouping[109], spatial and semantic supervision[2]. Although the powerful architectural advantage has driven the performance of transformer-based approaches, low recall and the difficulty of distributing initial queries remain challenges. Moreover, as the transformer-based approaches directly generate the final prediction mask without the refinement step, there is no straightforward method to apply the transformer-based approach to the proposed coarse-to-fine method. However, it has significant potential to incorporate the refinement step into transformer-based methods to improve the segmentation performance.

### 3.2.5 Summary

In summary, we introduce existing point cloud instance segmentation works with four categories. Due to the similar architecture for detection, the coarse-to-fine approach has pioneered the field. However, their limitations for object representation with box and the two-stage architecture with error accumulation led to

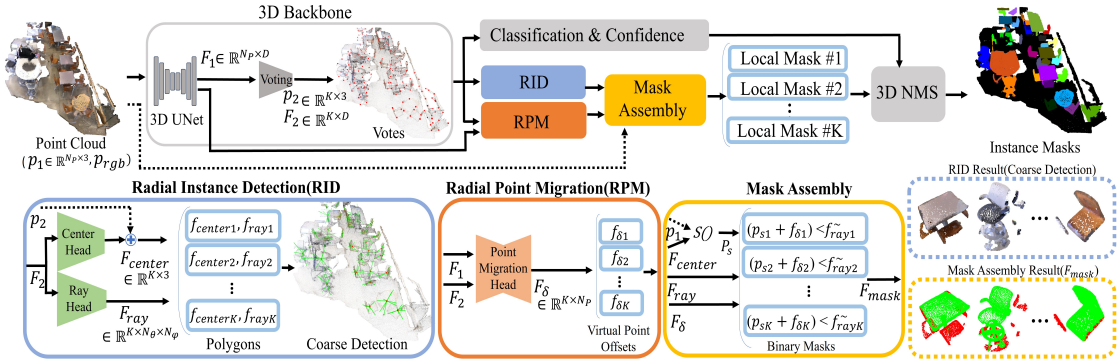



Figure 3.2: The overall pipeline of our proposed method is based on a coarse-to-fine approach. Given the point cloud, the backbone produces base features with 3D UNet and a Voting module. Based on this, RID performs coarse detection while RPM produces the virtual point offsets to refine the coarse detection. In Mask Assembly, $K$ local binary masks are generated, where each mask is a proposal for a single instance. 3D NMS is applied to acquire the final instance masks using local binary masks, classifications, and confidence scores.

the development of other techniques, such as grouping, kernel, and transformer-based approaches. Although the coarse-to-fine approach has limitations, its two-stage structure, the coarse detection in particular, has a clear advantage in reducing the class imbalance problem caused by outnumbering background points. In the following section, we delve into the coarse-to-fine approach for utilizing its advantage while sidestepping the limitations.

## 3.3 Method

### 3.3.1 Overview

Given an input point cloud $p_1 \in \mathbb{R}^{N_p \times 3}$ in 3-dimensional cartesian coordinates and the corresponding colour information $p_{\text{rgb}} \in \mathbb{R}^{N_p \times 3}$, we aim to design a system that segments the point cloud into local binary masks of instances $\{o^{(i)} \in \mathbb{R}^{N_p \times 1}\}_{i=1}^{N_o}$ using a coarse to fine approach. Here, $N_p, N_o$ are the total number of points and the number of instances, respectively. This is achieved using the proposed method depicted in Figure 3.2. Our system consists of mainly two modules: 3D backbone at the top-left of Figure 3.2 and proposed instance mask estimation. The details of these modules are in Section 3.3.2 and Section 3.3.3, respectively.

### 3.3.2 3D Backbone

Our 3D backbone is similar to [158] and is composed of two modules: a 3D encoder and a voting module. The 3D encoder uses U-Net [138] with sparse convolutions [50] to encode the given point cloud into deep features $F_1 \in \mathbb{R}^{N_p \times D}$. Then, $F_1$ and the respective input points $p_1$ are fed into the voting module. The voting module performs set abstraction [128], consisting of three main steps: sampling, grouping, and feature extraction. The first step employs the farthest point sampling (FPS) strategy to select a subset of representative points, reducing computational complexity while preserving coverage. Next, the grouping step organizes neighboring points into local regions around the sampled points, using a predefined radius or k-nearest neighbors (k-NN). Finally, feature extraction applies the original PointNet [125] architecture within each group to learn local features, which are then aggregated using max pooling to form a compact representation of the point cloud.

In our work, the set abstraction produces $K$ votes with query points $p_2 \in \mathbb{R}^{K \times 3}$ and features $F_2 \in \mathbb{R}^{K \times D}$. These votes are spread in the scene, providing features to be further processed through the proposed instance mask estimation module. The details are explained in the following sections.

### 3.3.3 Instance Mask Estimation

In this section, we estimate the instance masks from the votes predicted in the 3D backbone section using three modules: Radial Instance Detection, Radial Point Migration, and Mask Assembly. All of the modules are explained in the following sections. For the notation simplicity, we will explain how each vote feature is processed. Therefore, we write $f_2$ to refer to a single vote feature of $F_2$ from here onwards.

**Radial Instance Detection(Coarse Mask)**: Radial Instance Detection(RID) aims to detect instances for further refinement. Similar to PolarMask[175] for 2D segmentation, we define an instance as a 3D polygon with a centre $f_{\text{center}}$ and multiple rays $f_{\text{ray}}$ emitting from the centre forming each spherical sectors. Here, the sectors are defined by preset angles. Each ray then determines the distance to be considered for their corresponding sectors, as shown in Figure 3.3.

We estimate the closest instances' centre $f_{\text{center}}$ using offsets predicted from an MLP network, *CenterHead*, which takes the respective $f_2$ as input and outputs offsets. The offsets are added to $p_2$ to infer $f_{center}$. After this, the input point cloud is converted into spherical coordinates using a transformation as $p_s = S(p_1)$ centred

around $f_{\text{center}}$, where $S : (x, y, z) \rightarrow (r, \theta, \varphi)$ as follows:

$$\begin{aligned} r &= \sqrt{x^2 + y^2 + z^2}, \\ \theta &= \arctan \tfrac{x}{y}, \\ \varphi &= \arccos \tfrac{z}{\sqrt{x^2+y^2+z^2}}. \end{aligned} \tag{3.1}$$

Here, based on the relationship between Cartesian and Spherical cooridnate systems, $r$, $\theta$, $\varphi$, arctan, and arccos refer to radius (or Radial Distance), horizontal (or Azimuth Angle), and vertical angles (or Inclination Angle) in spherical coordinates, the inverse function of the tangent, and the inverse cosine function, respectively. Specifically, $r$ refers to the radial distance of the point from the origin. The inclination angle is determined by the ratio of the vertical height to the radial distance measured using the inverse cosine function. The Azimuth angle uses the two-dimensional arctangent function to account for the full 360 degree in the xy-plane.

The $p_s$ is then divided uniformly with $N_\theta$ and $N_\varphi$ separations for horizontal and vertical direction respectively, resulting in $N_\theta \cdot N_\varphi$ sectors, as shown in Figure 3.3 (b) and (c).

We consider points inside the same sector to have identical $(\theta, \varphi)$, which enables us to close the sector using a boundary estimated by $f_{\text{ray}}$. To estimate $\{f_{ray}^{(i)}\}_{i=1}^{N_\theta N_\varphi}$, another MLP named *Ray Head* is employed, which takes $f_2$ as input.

At this point, every point inside the corresponding sector's boundary from $f_{ray}$ is considered foreground. Using the boundary, RID offers tighter boundaries of instances than AABB in the point cloud, as each sector is closed at the distance of the farthest foreground point in the sector, alleviating the problem of redundant space in AABB. Please refer to Section 3.4.7 for detailed results comparing RID and AABB. Also, since the ground truth for each vote is assigned based on its distance to the closest object center, it is agnostic to the number of points on the surface of the object, which can be learned when the object is not well-viewed due to sparse points on its surface. We chose the object center as the goal for the regression output of *CenterHead* as 1) it is easy to calculate using the foreground points from the ground truth and 2) it could provide reliable features for each class of objects in Spherical Coordinate (e.g. average distance between the center and bottom of chairs and desk), compared to using other locations as the origin of the spherical coordinates of the objects.

**Coarse Instance Loss:** During training, the estimated $f_{\text{centre}}$ and $f_{\text{ray}}$ are compared against their respective ground truth $\mathbf{g}_{\text{center}}$ and $\mathbf{g}_{\text{ray}}$ to calculate $L_{\text{coarse}}$:

$$L_{\text{coarse}} = L_{\text{ray}} + L_{\text{center}}, \tag{3.2}$$

where $L_{\text{ray}}$ and $L_{\text{center}}$ are defined with L1 loss:

$$L_{\text{ray}} = \frac{1}{N_\theta N_\varphi} \sum_{i=1}^{N_\theta N_\varphi} \left| f_{\text{ray}}^{(i)} - \mathbf{g}_{\text{ray}}^{(i)} \right|_1 \tag{3.3}$$

$$L_{\text{center}} = \left| f_{\text{center}} - \mathbf{g}_{\text{center}} \right|_1, \tag{3.4}$$

Here, the ground-truth instance $\mathbf{g}$ is matched with $f_{\text{center}}$ by an injective mapping obtained using Hungarian algorithm as [181, 158]. For the details of the matching, please refer to Sec 3.3.6. $\mathbf{g}_{\text{ray}}^{(i)}$ is set to the distance between $\mathbf{g}_{\text{center}}$ and the furthest foreground point in the $i_{th}$ sector. If there are no foreground points in the sector, $\mathbf{g}_{\text{ray}}^{(i)}$ is set to minimum as $1e-5$. The target centre $\mathbf{g}_{\text{center}}$ is calculated as mean values of foreground points of the matched ground-truth instance in cartesian coordinates.

### 3.3.4 Radial Point Migration (Mask Refinement)

In this section, we introduce a refinement process to perform per-point fine-tuning. This is because the coarse detection will invariably include points belonging to the background or other instances (i.e. false positives) inside its boundary and neglect some instance points that fall outside (i.e. false negatives). We propose a conceptually simple yet effective dual that jitters individual points to belong to the correct instance. In particular, this is enabled by our innovative use of spherical coordinates - we only need to learn a single radial delta for each point to move it along the ray to the instance centroid while keeping angular quantities $\phi$ and $\theta$ constant. Note that this is a *virtual* point motion - we do not alter the final point cloud. We use this as a virtual offset to obtain clean instance labels without modifying the coarse sector.

By estimating an offset value for each point $p_1$, these misclassified points can be virtually migrated to being in the correct region. Based on its good performance on per-point prediction, we adopt Dynamic Convolution in a similar manner to [56, 158] as our *Point Migration Head*, for predicting per-point offsets $F_\delta \in \mathbb{R}^{K \times N_p}$ using the vote features $F_2 \in \mathbb{R}^{K \times D}$ as queries against the point features $F_1 \in \mathbb{R}^{N_p \times D}$. For the coherence with the notations, we write $f_\delta \in \mathbb{R}^{N_p}$ to refer to an output of $F_\delta$, corresponding to one vote. For the learning of $f_\delta$, we divide points into two groups. The first is to learn the radial delta for the case of misclassification, and the second is to make instances more compact and cohesive by migrating points to the centroid of the sector.

**Misclassification Correction Loss**: This process aims to estimate a radial delta to move the misclassified points either inside or outside the estimated coarse sector. There are two possible cases where the misclassification could occur, as shown in

Figure 3.4 (a): Instance points could lie outside the sector boundary, acting as false negatives, or background points could incorrectly lie within the sector boundary, acting as false positives. The goal is to move these points to the correct region.

Formally, given the point indices of foreground points $\{j+^{(i)}\}_{i=1}^{N+}$ and background points $\{j-^{(i)}\}_{i=1}^{N-}$ from the groundtruth, the false negative points $p_{\text{fn}}$ and the false positive points $p_{\text{fp}}$ are defined as:

$$p_{\text{fn}} = \{p_s^{(j+)} : (p_s^{(j+)} + f_\delta^{(j+)}) > f_{\text{ray}}^{(\tilde{j+})}\}, \tag{3.5}$$

$$p_{\text{fp}} = \{p_s^{(j-)} : (p_s^{(j-)} + f_\delta^{(j-)}) < f_{\text{ray}}^{(\tilde{j-})}\}, \tag{3.6}$$

where $N+$ and $N-$ stand for the number of foreground and background points, respectively. Here, $\tilde{j} = \textit{findSector}(p_s^{(j)})$ where *findSector(.)* is a function that takes $p_s^{(j)}$ as input and returns the index of the sector that $p_s^{(j)}$ belongs to. Please refer to Section 3.4.3 for the implementation of *findSector(.)* function.

The union of $p_{\text{fp}}$ and $p_{\text{fn}}$ forms the misclassified points $p_{\text{miss}}$ that we are interested: $p_{\text{miss}} = p_{\text{fp}} \cup p_{\text{fn}}$. Our aim is to push or pull them inside/outside of the ray with margins. Thus, the loss function $L_{mc}$ is formulated with soft margin loss as:

$$L_{mc} = \frac{1}{N_{\text{miss}}} \sum_{i=1}^{N_{\text{miss}}} \log(1 + \exp(y * \tanh(p_{\text{miss}}^{(i)} + f_\delta^{(\hat{i})} - f_{\text{ray}}^{(\tilde{i})}))) \tag{3.7}$$

where

$$y = \begin{cases} 1 & \text{if } p_{\text{miss}}^{(i)} \in p_{\text{fp}}, \\ -1 & \text{if } p_{\text{miss}}^{(i)} \in p_{\text{fn}}, \end{cases} \tag{3.8}$$

Here, tanh is hyperbolic-tangent function and $L_{mc}$ is calculated for all the votes that are assigned to the ground truth instance. $N_{\text{miss}}$ stands for the number of element in $p_{\text{miss}}$ and $\hat{i}$ is index of $f_\delta$ corresponding to $p_{\text{miss}}^{(i)}$. $f_{\text{ray}}$ is only used for reference and the gradient for learning $f_{ray}$ is not calculated.

The misclassified points around the edge,$g_{\text{ray}}$, of an instance are provided with comparably easy learning targets. In contrast, misclassified points far from the predicted rays are assigned targets with large discrepancies, encouraging larger gradients during the training.

**Sector Cohesion Loss** The goal of this block is to move true-positive points to the centroid of the sector. By doing so, the sector becomes more cohesive, encouraging the learning of common and shared features of an instance as the foreground features are getting close to each other. In addition, this helps to provide a learning signal for true-positive points, as $L_{\text{mc}}$ only considers false negatives/positives. This is shown more clearly in Figure 3.4 (b).

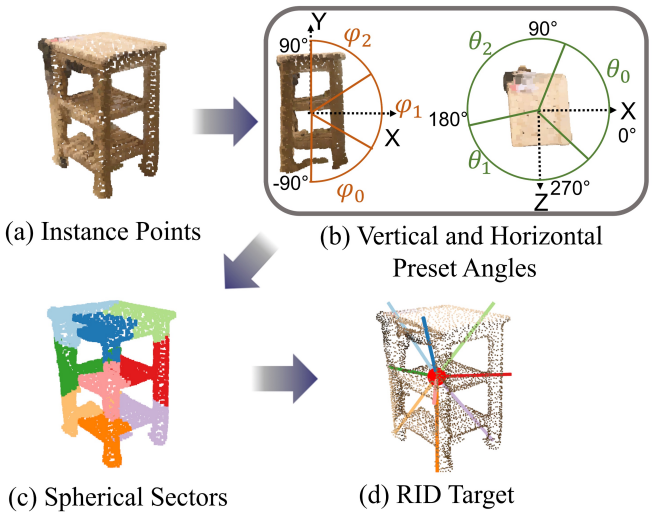



Figure 3.3: Process of RID. (a) Object points in cartesian coordinates (b) Converting points into a spherical coordinate system, using $f_{\text{center}}$, and preset angles $\theta$ and $\varphi$. (c) Assigning points to each sector defined by $\theta$ and $\varphi$. The example shows 3/3 for $\theta/\varphi$. (d) For each sector, the distance between the farthest point and the centre becomes the target of $f_{\text{ray}}$. During inference, points with smaller distance than $f_{\text{ray}}$ are considered foreground.

Similar to $L_{mc}$, using point indice of foreground points $j+$, we extract true positives $p_{\text{tp}}$ as :

$$p_{\text{tp}} = \{p_s^{(j+)} : (p_s^{(j+)} + f_\delta^{(j+)}) < f_{\text{ray}}^{(\tilde{j}+)}\} \tag{3.9}$$

and formulate the loss with the soft margin calculation:

$$L_{sc} = \frac{1}{N_{\text{tp}}} \sum_{i=1}^{N_{\text{tp}}} \log(1 + \exp(\tanh(f_\delta^{(\hat{i})} + p_{\text{tp}}^{(i)} - f_{\text{center}}))), \tag{3.10}$$

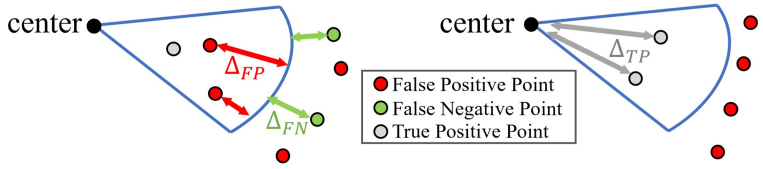
where $N_{\text{tp}}$ stands for the number of element in $p_{\text{tp}}$ and $\hat{i}$ is index of $f_\delta$ corresponding to $p_{\text{tp}}^{(i)}$. Since $f_{\text{center}}$ is always 0 in centred spherical coordinate, the loss can be simplified as:

$$L_{sc} = \frac{1}{N_{\text{tp}}} \sum_{i=1}^{N_{\text{tp}}} \log(1 + \exp(\tanh(f_\delta^{(\hat{i})} + p_{\text{tp}}^{(i)}))), \tag{3.11}$$

$L_{mc}$ and $L_{sc}$ together form the refinement loss $L_{\text{fine}}$ as:

$$L_{\text{fine}} = L_{mc} + L_{sc}. \tag{3.12}$$

Our proposed virtual point migration brings three advantages for refinement over existing approaches that strictly disentangle coarse detection and refinement: 1) Instead of only focusing on points inside the coarse detection, predicting the offsets of all points allows the refinement of even false negative points outside of $f_{\text{ray}}$, sidestepping the error accumulation from the coarse detection. 2) By considering the sector radius, $g_{\text{ray}}$, it is possible to have a soft target for each point rather than a




(a) Misclassification Correction Loss (b) Sector Cohesion Loss

Figure 3.4: Conceptual diagram showing per-point migration following both (a)$L_{\text{mc}}$ and (b)$L_{\text{sc}}$. $\Delta_{\text{FP}}$ and $\Delta_{\text{FN}}$ are distances penalized by $L_{\text{mc}}$ with margin for misclassified points. $\Delta_{\text{TP}}$ is the distance that $L_{\text{sc}}$ penalizes to enforce the learning of general features of an instance by making each sample close to the other around the centre.

hard target which is the centre of the sector/instance. For example, false negative points outside of the sector boundary only need to be migrated a small distance to being with the sector (soft) rather than being driven towards the centre (hard). This makes it easier to learn how to perform the point migration. 3) We only need to learn a one-dimensional number to migrate the point virtually along the radial line. This is far simpler than having to learn a three-dimensional offset in cartesian coordinates.

### 3.3.5 Mask Assembly

Our final mask is assembled by comparing virtually migrated points and $f_{ray}$. Specifically, the local binary mask $\{f^{(i)}_{\text{mask}}\}^{N_p}_{i=1}$ is formed as:

$$f^{(i)}_{\text{mask}} = \begin{cases} 1 & \text{if } (p^{(i)}_s + f^{(i)}_\delta) < f^{(\tilde{i})}_{\text{ray}} \\ 0 & \text{otherwise,} \end{cases} \tag{3.13}$$

### 3.3.6 Training

For the training, we also learn classification and confidence with respective MLPs. For classification, we apply cross-entropy loss, $L_{\text{cls}}$, for learning the classes of matched ground-truth instances. For the confidence scores of the proposals, we apply L2 loss to learn IoUs between the proposals and the ground-truth instances. Similar to [158], we also duplicate the number of ground-truth for 4 times and create a cost matrix $C$:

$$C(k,i) = L_{\text{coarse}}(k,i) + L_{\text{fine}}(k,i) + L_{\text{cls}}(k,i), \tag{3.14}$$

where $L(,)$ refers to the loss value calculated using $k_{th}$ vote and $i_{th}$ ground-truth instance. Referring to $C$, we apply Hungarian algorithm to find the least-cost injective mapping from each ground-truth instance to the votes. The final loss using

the acquired ground-truths is:

$$L = \lambda_1 L_{\text{cls}} + \lambda_2 L_{\text{conf}} + \lambda_3 L_{\text{coarse}} + \lambda_4 L_{\text{fine}} \tag{3.15}$$

## 3.4 Experiments

| Method | mAP | $mAP_{50}$ | bath | bed | bk.shf | cabinet | chair | counter | curtain | desk | door | other | picture | fridge | s. cur. | sink | sofa | table | toilet | wind. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3D-BoNet(C)[181] | 25.3 | 48.8 | 51 | 32 | 25 | 13 | 34 | 3 | 41 | 6 | 16 | 13 | 5 | 20 | 33 | 14 | 30 | 30 | 65 | 17 |
| TD3D(C)[79] | 48.9 | 75.1 | 85 | 51 | 43 | 32 | 73 | 10 | 51 | 35 | 34 | 46 | 28 | 51 | 67 | 26 | 67 | 51 | 90 | 32 |
| SoftGroup(G)[160] | 50.4 | 76.1 | 66 | 57 | 37 | 38 | 69 | 7 | 67 | 30 | 38 | 53 | 31 | 58 | 75 | 31 | 64 | 49 | 90 | 38 |
| PBNet(G)[198] | 57.3 | 74.7 | 92 | 57 | 61 | 47 | 73 | 23 | 48 | 38 | 45 | 50 | 53 | 58 | 76 | 40 | 71 | 55 | 96 | 38 |
| DKNet(K)[173] | 53.2 | 71.8 | 81 | 62 | 51 | 37 | 74 | 10 | 50 | 30 | 43 | 47 | 58 | 53 | 77 | 33 | 64 | 50 | 90 | 38 |
| ISBNet(K)[158] | 55.9 | 75.7 | 93 | 65 | 38 | 42 | 76 | 18 | 53 | 38 | 49 | 50 | 62 | 42 | 70 | 46 | 64 | 57 | 94 | 40 |
| MAFT(T)[82] | 57.8 | 78.6 | 88 | 72 | 44 | 46 | 76 | 25 | 55 | 40 | 50 | 53 | 61 | 61 | 85 | 48 | 68 | 55 | 93 | 45 |
| QueryFormer(T)[109] | 58.3 | 78.7 | 92 | 70 | 39 | 50 | 73 | 27 | 52 | 37 | 47 | 53 | 53 | 69 | 72 | 43 | 74 | 59 | 95 | 36 |
| Ours(C) | **61.6** | **81.2** | 94 | 65 | 55 | 43 | 76 | 27 | 60 | 44 | 50 | 54 | 69 | 71 | 77 | 48 | 74 | 57 | 92 | 43 |

Table 3.1: Quantitative comparison of top-performing methods for each approach on ScanNetV2 **hidden** test set. (C),(G),(K), and (T) next to the names of the methods refer to coarse-to-fine, grouping, kernel, and Transformer based methods, respectively. All the methods take the same input, such as point cloud and corresponding colour information. The best results are in bold, and the second best ones are underlined.

### 3.4.1 Dataset

We evaluate our method on three datasets: ScanNetV2 [31], S3DIS [3], STPLS3D [18]. Following are descriptions of each dataset.

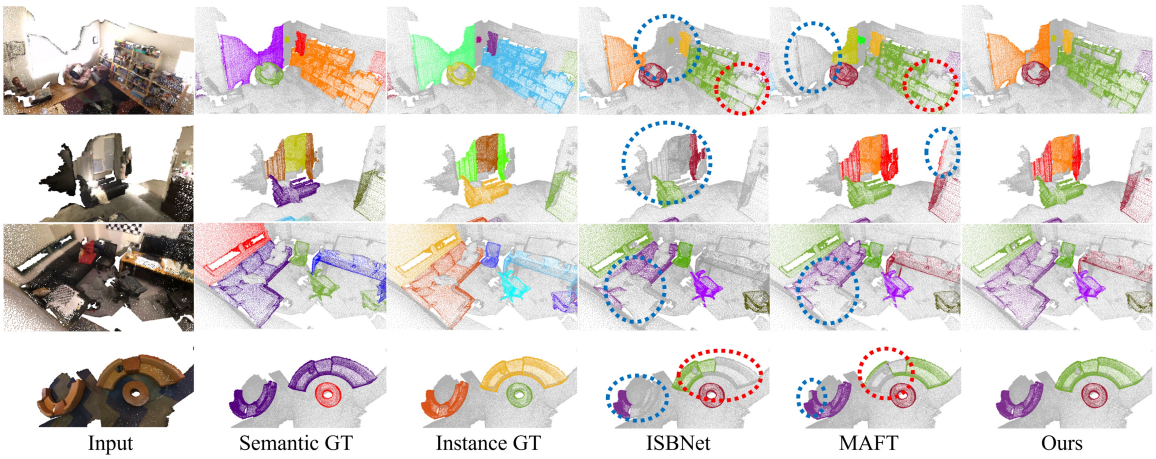



Figure 3.5: Qualitative comparison of ISBNet [158], MAFT [82], and ours on ScanNetV2 validation set. The coloured circles indicate wrong segmentation results.

**ScanNetV2** ScanNetV2 dataset consists of 1201, 312, and 100 scans with 18 object classes for training, validation, and testing, respectively. We report the evaluation results on the validation and hidden test sets as in the existing works.


Figure 3.6: Additional qualitative results on ScanNetV2 validation set, including TD3D [79]. The coloured circles indicate wrong segmentation results.

**S3DIS** S3DIS dataset contains 271 scenes from 6 areas with 13 categories. We report evaluations for both Area 5. Additional evaluation results with 6-fold cross-validation can be found in the supplementary material.

**STPLS3D** The STPLS3D dataset is an aerial photogrammetry point cloud dataset from real-world and synthetic environments. It includes 25 urban scenes of $6km^2$ and 14 instance categories. Following [19, 160, 158], we use scenes 5, 10, 15, 20, and 25 for validation and the rest for training.

### 3.4.2 Evaluation Metrics

We adopt average precision as our primary evaluation metric. Average precision is extensively used in vision tasks such as object detection and instance segmentation tasks. The metric calculates precisions by varying the IoU threshold. Following the existing works, we evaluate our model with three IoU thresholds: $AP$, $AP_{50}$, and $AP_{25}$. $AP_{50}$ and $AP_{25}$ stand for average precisions with IoU threshold as 25%

| Method | Venue | mAP | AP50 | AP25 |
|---|---|---|---|---|
| 3D-SIS(C)[59] | CVPR19 | - | 18.7 | 35.7 |
| GSPN(C)[187] | CVPR19 | - | 37.8 | 53.4 |
| TD3D(C)[79] | WACV23 | 47.3 | 71.2 | 81.9 |
| PointGroup(G)[71] | CVPR20 | 34.8 | 51.7 | 71.3 |
| SSTNet(G)[95] | ICCV21 | 49.4 | 64.3 | 74.0 |
| MaskGroup(G)[201] | ICME22 | 27.4 | 42.0 | 63.3 |
| SoftGroup(G)[160] | CVPR22 | 46.0 | 67.6 | 78.9 |
| RPGN(G)[35] | ECCV22 | - | 64.2 | - |
| PBNet(G)[198] | ICCV23 | 54.3 | 70.5 | 78.9 |
| PointInst3D(K)[57] | EECV22 | 45.6 | 63.7 | |
| DKNet(K)[173] | ECCV22 | 50.8 | 66.9 | 76.9 |
| ISBNet(K)[158] | CVPR23 | 54.5 | 73.1 | 82.5 |
| Mask3D(T)[141] | ICRA23 | 55.2 | 73.7 | 82.9 |
| 3IS-ESSS(T)[2] | ICCV23 | 56.1 | 75.0 | 83.7 |
| QueryFormer(T)[109] | ICCV23 | 56.5 | 74.2 | 83.3 |
| MAFT(T)[82] | ICCV23 | <u>58.4</u> | <u>75.6</u> | <u>84.5</u> |
| Ours(C) | - | **62.3** | **79.9** | **88.2** |

Table 3.2: Quantitative 3D instance segmentation results on ScanNetV2 validation set. (C),(G),(K), and (T) next to the names of the methods refer to coarse-to-fine, grouping, kernel, and Transformer based methods, respectively. The best results are in bold, and the second best ones are <u>underlined</u>.

and 50%, respectively. *AP* is an averaged score by varying IoU thresholds from 50% to 95% by increasing the threshold with step size 5%. For S3DIS, we also evaluate our model with mean precision (mPrec50) and mean recall with IoU threshold as 50%(mRec50).

### 3.4.3 Implementation Details

We build our model on PyTorch framework [122] and train it for 300 epochs with AdamW optimizer with a single NVIDIA A10 GPU. The batch size is set to 10. The learning rate and weight decay are initialized to 0.001 and 0.0001. Cosine annealing [191] is used for scheduling the learning rate. Following [160, 158], the voxel size is set to 0.02m for ScanNetV2 and S3DIS, and to 0.3m for STPLS3D. For the augmentation during training, we use random cropping for each scene with a maximum number of 250,000 points. During testing, a whole scene is used as input for the network.

Our backbone is similar to [160, 158], which outputs features $F_1$ with hidden dimension $D$ as 32 channels. For the voting, we use two set-abstraction [128, 126] layers with the ball query radius 0.2 and 0.4, respectively. The number of seeds and votes are set to 1024 and 256, respectively. The number of neighbours is set to 32 for both layers, similar to [158]. For *Point Migration Head*, we use two layers

of dynamic convolution[56], and their hidden dimensions are set to 32. $\lambda 1$, $\lambda 2$, $\lambda 3$, and $\lambda 4$ are set to 0.5, 0.5, 1, and 1, respectively. For training and inference, we set $N_\theta$ and $N_\varphi$ to 5 and 5, respectively. During inference, Non-Maximum-Suppression is applied to the $K$ binary masks to delete redundant masks using a confidence score of 0.2 as a threshold. Following [95, 173, 158, 109], we aggregate superpoints[84, 83] to align the final prediction masks on the ScanNetV2 dataset. Other details, such as the architecture of MLPs and runtime analysis, are following.

**Details of Voting:** Similar to [126, 158], we use two set-abstract layers with the following structures. The first set-abstraction layer has a ball-query radius of 0.2 and chooses 1024 points with farthest-point sampling (FPS). The maximum number of neighboring points each point considers is set to 32. The grouped features are the point locations($\mathbb{R}^{1024\times 3}$) and the corresponding features($\mathbb{R}^{1024\times D}$) from the encoder. Here, $D$ is 32, as specified in Section 3.4.3. The grouped features are then fed into MLP, which consists of 3 blocks, each with Convolution, BatchNorm and ReLU. The output is followed by max-pooling to produce 64 channels of features from the 1024 sub-sampled points. We also add a residual connection using the initial features from the encoder to prevent the gradient vanishing problem.

The second set-abstraction layer has 0.4 and 256 for ball-query radius and number of points for FPS, respectively. Following the same procedure as the set-abstraction layer from the first sampling, the MLP layer outputs 32 channels of features for 256 sub-sampled points, which corresponds to $F_2$ in Section 3.3.2. Here, the output feature dimension of the set-abstraction is reduced to 32 to minimize the memory consumption during training. Empirically, we have not found meaningful differences in performance by setting the feature dimensions higher, such as 64 and 128.

**Details of Dynamic Convolution:** As mentioned in the Implementation Detail, we implement Dynamic Convolution for the per-point offset prediction(RPM) with two layers with 32 hidden dimensions. Similar to [56, 158], an MLP consisting of two blocks with Convolution, BatchNorm, and ReLU as one block generates weights $W \in \mathbb{R}^{K\times H}$ for each instance by taking vote features $F_2 \in \mathbb{R}^{K\times D}$ as input. Here, $H$ is decided by the parameters such as hidden dimension and input feature dimension for dynamic convolution. $W$ is then used to convolve $F_2$ and the predicted instance center $F_{\text{center}} \in \mathbb{R}^{K\times 3}$, as follows:

$$F_\delta = Conv\left([F_2, F_{\text{center}}]; W\right), \tag{3.16}$$

where $Conv$ is implemented as two convolutional layers which have hidden feature dimensions of 32 from $W$. Here, $[,]$ refers to concatenation.

**Details MLP based Heads:** There are four MLPs used as final prediction heads: (1) *CenterHead* (2) *RayHead* (3) Classification and (4) Confidence. All of them take $F_2 \in \mathbb{R}^{K \times D}$ as input.

Each MLP consists of two blocks of Convolution, BatchNorm and ReLU. For the third block, we only use the convolution as an inference output. The momentum is set to 0.1 for all BatchNorm. All MLPs for heads have the same hyper-parameters, except for the output dimensions. For example, classification head outputs scores for each class ($\mathbb{R}^{K \times N_{\text{class}}}$), and confidence head outputs confidence score( $\mathbb{R}^{K \times 1}$). Here, $N_{\text{class}}$ stands for the number of classes.

**Details of Mask Assembly and Finding Misclassified Points:** As mentioned in Section *Mask Assembly*, the final mask is assembled by comparing if the addition of $p_s^{(i)}$ and $f_\delta^{(i)}$ is smaller than $f_{ray}^{(\tilde{i})}$, as in Equation 3.13. Here, the addition of $p_s^{(i)}$ and $f_\delta^{(i)}$ is performed by adding $f_\delta^{(i)}$ only to $r$ coordinate of $p_s^{(i)} \in \mathbb{R}^{N_p \times 3}$ among $(r, \theta, \varphi)$ because $\theta$ and $\varphi$ coordinates for each point remain constant within their respective sector. This operation is the same for finding the misclassified points in Equations 3.5, 3.6 and 3.9.

During the inference time, the same operation is applied to generate the binary masks for all $K$ votes, $F_{mask} \in \mathbb{R}^{K \times N_p}$, using $K$ point clouds in spherical coordinates, $P_s \in \mathbb{R}^{K \times N_P \times 3}$, centered to predicted instance centers $F_{\text{center}} \in \mathbb{R}^{K \times 3}$ in cartesian coordinates. The runtime analysis, including the finding of the index $\tilde{i}$ using $findSector(.)$, is illustrated in Section 3.4.3.

**Details of Training** In the method section 3.3.6, we illustrate how the loss for each vote is calculated. As mentioned in *Implementation Detail*, this loss is calculated for the number of votes mapped to the ground-truth instance with the batch size 10. Therefore, the final loss $L_{\text{total}}$ is the averaged value of:

$$L_{\text{total}} = \frac{1}{N_{\text{batch}} N_{\text{inst}} N_{\text{dupl}}} \sum_{b=1}^{N_{\text{batch}}} \sum_{k=1}^{N_{\text{inst}}^{(b)} N_{\text{dupl}}} L^{(b,k)}, \tag{3.17}$$

where $N_{\text{batch}}$, $N_{\text{inst}}$, and $N_{\text{inst}}^{(b)}$ stand for batch size, a total number of ground-truth instances in the batch, and a number of ground-truth instances in $b_{\text{th}}$ point cloud in the batch, respectively. $N_{\text{dupl}}$ is the number to duplicate ground-truth instances, and $L$ is the loss for each vote mapped to a ground-truth instance. As mentioned in Section 3.3.6 and 3.4.3, $N_{\text{batch}}$ and $N_{\text{dupl}}$ are set to 10 and 4 respectively.

**Implementation of $findSector(.)$:** The function $findSector(.)$ takes centered points $p_s \in \mathbb{R}^{N_p \times 3}$ in spherical coordinates and returns the index of the sectors that the points belong to as mentioned in Section 3.3.4 and Equations (3.5), (3.6), (3.9), (3.13). Alg. 1 shows the PyTorch implementation of $findSector$. $p_s$ for each vote

and batch(during training) are stacked and fed into $findSector$ at once for the efficient computation using GPU.

**Algorithm 1** $findSector$ in PyTorch

**Input:** $p_s$, $N_\theta$, $N_\varphi$
**Output:** Indices of sectors that each element of $p_s$ belong to

**Function** FINDSECTOR($p_s$,$N_\theta$,$N_\varphi$)
    $u_\theta$, $u_\varphi = 360/N_\theta$, $180/N_\varphi$ ▷ Horizontal and vertical unit angles for each spherical sector in degree
    $p_r, p_\theta, p_\varphi = p_s[:,0], p_s[:,1], p_s[:,2]$ ▷ Basis of spherical coordinates
    $\theta_{\text{angles}}$ = torch.arange(0,360, int($u_\theta$)) ▷ Finding the smallest positive value by subtraction
    $\varphi_{\text{angles}}$ = torch.arange(0,180, int($u_\varphi$))
    $\text{diff}_\theta = p_\theta$[:,:,None].repeat(1,1,len($\theta_{\text{angles}}$)) - $\theta_{\text{angles}}$[None,None,:]
    $\text{diff}_\varphi = p_\varphi$[:,:,None].repeat(1,1,len($\varphi_{\text{angles}}$)) - $\varphi_{\text{angles}}$[None,None,:]
    $\text{diff}_\theta[\text{diff}_\theta < 0] = u_\theta + 1$ ▷ Preventing negative values from being the smallest values
    $\text{diff}_\varphi[\text{diff}_\varphi < 0] = u_\varphi + 1$
    $\text{diff}_\theta = \text{diff}_\theta / u_\theta$
    $\text{diff}_\varphi = \text{diff}_\varphi / u_\varphi$
    $\theta_{\text{sector}}$ =torch.argmin($\text{diff}_\theta$,2) ▷ The index with the smallest positive value is the sector that each point belongs
    $\varphi_{\text{sector}}$ =torch.argmin($\text{diff}_\varphi$,2)
    **return** $\varphi_{\text{sector}} * N_\theta + \theta_{\text{sector}}$ ▷ Sector indices

**Runtime Analysis:** In this section, we report a runtime analysis of Spherical Mask. Figure 3.7 (a) shows the execution speed of each module with respect to the number of input points on ScanNetV2. Here, Mask Assembly includes the $findSector$ function and the binary mask generation using the inequality operator, as in Equation (3.13). As can be seen, the execution time gradually increases as the number of input points grows. The backbone takes the most time among the three due to the Furthest Point Sampling(FPS). The execution time of Mask Assembly also increases more rapidly compared to RID and RPM modules, as it includes more operations. By averaging the total time spent on each scan, Spherical Mask segments a point cloud in 167ms in ScanNetV2 validation set.

Figure 3.7 (b) compares the existing methods and Spherical Mask in terms of execution time and mAP on ScanNetV2 validation set. Spherical Mask achieves the second fastest execution speed while showing the strongest performance in mAP.

### 3.4.4 Main Results

**ScanNetV2** Table 3.1 and Table 3.2 show the quantitative result of instance segmentation on the test and validation sets. Our proposed method achieves the

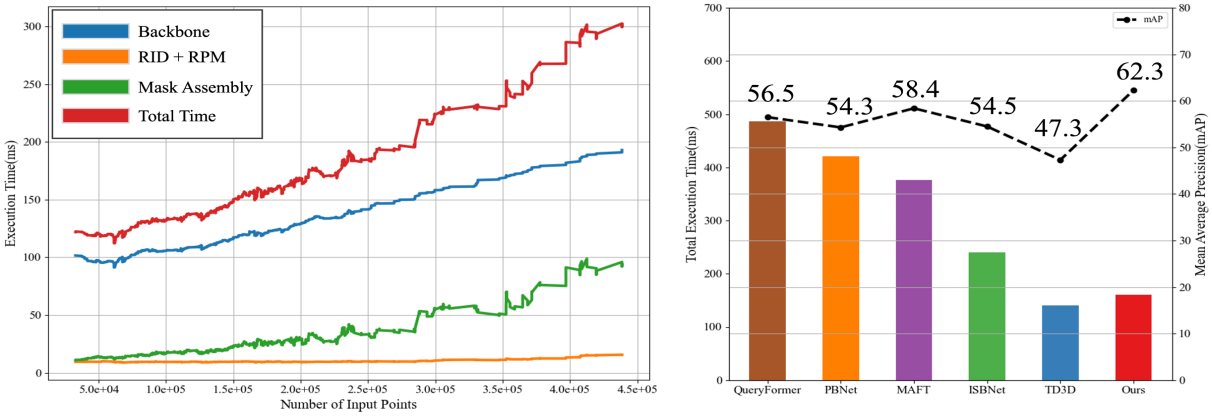



Figure 3.7: Runtime analysis on ScanNetV2 validation set. (a) The runtime of each component in Spherical Mask with respect to the number of input points using a NVIDIA A10 GPU. (b) Comparing the average execution time and the performance in mAP with recent works. On average, TD3D, ISBNet, MAFT, and *Ours* use 3947, 3264, 3723, and 3738 MiB on GPU for ScanNet, respectively.

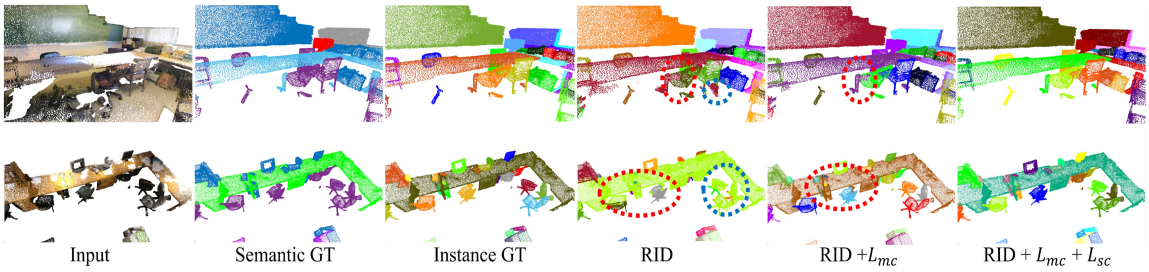



Figure 3.8: Visually comparing impacts of RID, $L_{mc}$, and $L_{sh}$. The coloured circles indicate wrong segmentation results.

highest AP and $AP_{50}$ surpassing the previous strongest method by the margin of 4.1% and 2.4% for the test set, and 6.7% and 5.7% for the validation set, respectively. Compared to the previous methods based on the coarse-to-fine approach, our method achieves 24.1% of the improvement in AP on the test set. In particular, for the test set, our method outperforms existing methods on instances that are typically located close to each other, such as pictures, desks, and bookshelves. This suggests that explicitly penalizing misclassified points around the edges of instances is helpful in RPM.

**S3DIS** Table 3.3 and Table 3.4 illustrate the quantitative result on S3DIS Area 5 and 6 fold cross-validation from published results. In Area 5, our proposed method outperforms the second-best-performing method with margins of 2.8, 2.39, and 4.09 in mAP, $AP_{50}$, and $mRec_{50}$, improving the performance of SOTA 4.8%, 3.4%, and 5.6% respectively. For the 6-fold cross-validation, our proposed method shows the improvement of 3.2, 1.7, and 0.6 in mAP, $AP_{50}$, and $mRec_{50}$, pushing the performance of SOTA for 5.3%, 2.4%, and 0.7% respectively. Figure 3.10 shows the

| Method | mAP | AP50 | $mPrec_{50}$ | $mRec_{50}$ |
|---|---|---|---|---|
| GSPN [187] | - | - | 36.0 | 28.7 |
| PointGroup [71] | - | 5.78 | 61.9 | 62.1 |
| HAIS [19] | - | - | 71.1 | 65.0 |
| SSTNet [95] | 42.7 | 59.3 | 65.6 | 64.2 |
| SoftGroup [160] | 51.6 | 66.1 | <u>73.6</u> | 66.6 |
| Mask3D [141] | 56.5 | 69.3 | 68.7 | 70.7 |
| TD3D [79] | 52.1 | 67.2 | - | - |
| RPGN [35] | - | - | 64.0 | 63.0 |
| PointInst3D [57] | - | - | 73.1 | 65.2 |
| DKNet [173] | - | - | 70.8 | 65.3 |
| ISBNet [158] | 56.3 | 67.5 | 70.5 | 72.0 |
| PBNet [198] | 53.5 | 66.4 | **74.9** | 65.4 |
| QueryFormer [109] | <u>57.7</u> | <u>69.9</u> | 70.5 | <u>72.2</u> |
| MAFT [82] | - | 69.1 | - | - |
| Ours | **60.5** | **72.3** | 71.3 | **76.3** |

Table 3.3: Quantitative 3D instance segmentation results on S3DIS Area 5. The best results are in bold, and the second best ones are <u>underlined</u>.

qualitative results on Area 5 of S3DIS.

**STPLS3D** Table 3.5 shows the quantitative comparison on the validation set of STPLS3D dataset. Our method outperforms all of the existing methods, improving SOTA performance in mAP and $AP_{50}$ for 3.0 and 4.3, respectively.

### 3.4.5 Qualitative Results

Fig.3.5 shows visual comparisons of ISBNet[158], MAFT[82], and our proposed Spherical Mask for challenging instances on ScanNetV2 validation set. Spherical Mask accurately segments large instances such as walls and bookshelves (row 1), curtains and a window between them(row 2), a large sofa(row 3) and a circular shape sofa(row 4).

ISBNet[158] struggles to segment large instances(wall and bookshelves in row 1) and a disconnected instance(curtains and a window between them in row 2). On the other hand, MAFT[82] shows better generalization capability for learning semantics(curtains and a window between them(row 2)). However, it struggles to segment some parts of an instance that look different, as shown in sofas(row 2,3) and over-segments the instance by considering physically further away points as the same instance(wall in row 2 and sofa in row 4), probably due to the local queries that overfit to certain semantics. More results are illustrated in Figure 3.6.

Figures 3.10 and 3.9 show the qualitative results on S3DIS and STPLS3D, respectively.

| Method | mAP | AP50 | $mPrec_{50}$ | $mRec_{50}$ |
|---|---|---|---|---|
| GSPN [187] | - | 54.4 | 38.2 | 31.2 |
| 3D-BoNet [181] | - | - | 65.6 | 47.7 |
| PointGroup [71] | - | 64.0 | 69.6 | 69.2 |
| OccuSeg [52] | - | - | 72.8 | 60.3 |
| HAIS [19] | - | - | 73.2 | 69.4 |
| SSTNet [95] | 54.1 | 67.8 | 73.5 | 73.4 |
| PointInst3D [57] | - | - | 76.4 | 74.0 |
| DKNet [173] | - | - | 75.3 | 71.1 |
| ISBNet [158] | 60.8 | 70.5 | 77.5 | 77.1 |
| PBNet [198] | 59.5 | 70.6 | **80.1** | 72.9 |
| Ours | **64.0** | **72.3** | 78.1 | **77.7** |

Table 3.4: Quantitative 3D instance segmentation results on S3DIS 6-fold cross-validation. The best results are in bold, and the second best ones are underlined.

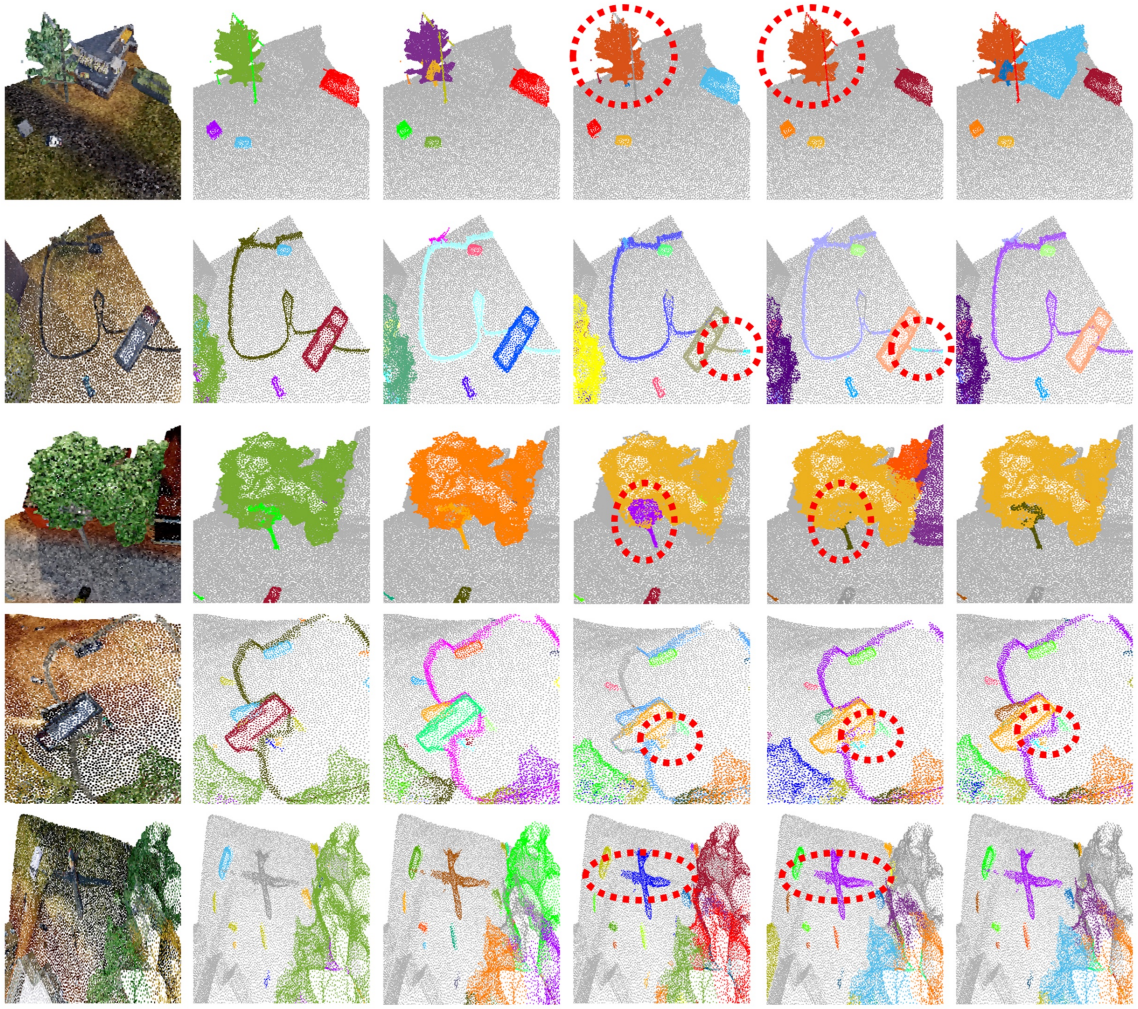



Figure 3.9: Qualitative results on STPLS3D. The coloured circles indicate wrong segmentation results.

| Method | mAP | $AP_{50}$ |
|---|---|---|
| HAIS [19] | 35.0 | 46.7 |
| PointGroup [71] | 23.3 | 38.5 |
| ISBNet [158] | 49.2 | 64.0 |
| Ours | **52.2** | **68.3** |

Table 3.5: Quantitative instance segmentation results on STPLS3D

| AABB | RID | $L_{mc}$ | $L_{sc}$ | mAP | $AP_{50}$ | $AP_{25}$ |
|---|---|---|---|---|---|---|
| ✓ | | | | 36.5 | 57.6 | 77.4 |
| | ✓ | | | 51.6 | 69.4 | 86.8 |
| | ✓ | ✓ | | 58.5 | 77.6 | 87.5 |
| | ✓ | | ✓ | 56.5 | 77.1 | 87.1 |
| | ✓ | ✓ | ✓ | **62.3** | **79.9** | **88.2** |

Table 3.6: Impact of each component on ScanNetV2 validation set

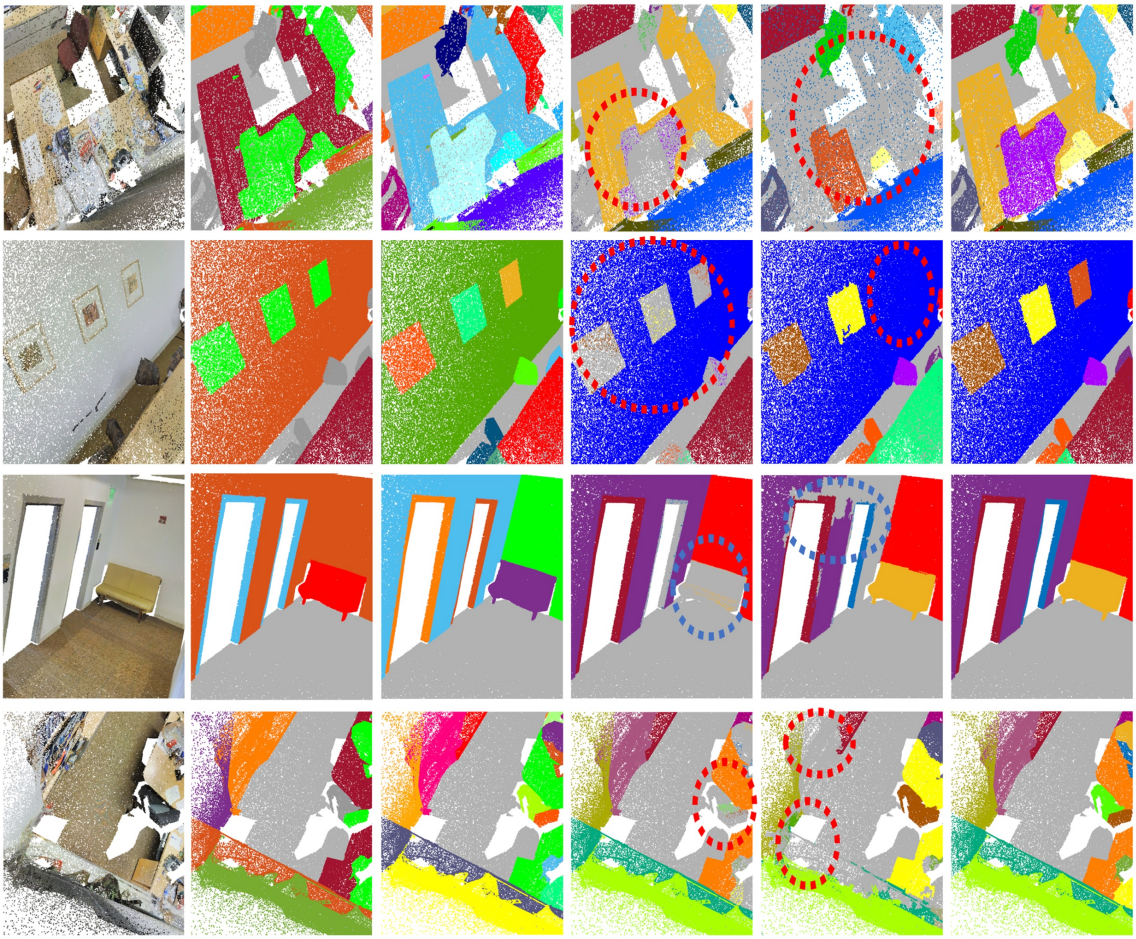



Figure 3.10: Qualitative results on S3DIS Area 5. The coloured circles indicate wrong segmentation results.

| $N_\theta / N_\varphi$ | mAP | $AP_{50}$ | $AP_{25}$ |
|---|---|---|---|
| 1/1 | 59.6 | 77.1 | 85.3 |
| 2/2 | 60.0 | 77.7 | 86.1 |
| 3/3 | 60.6 | 78.4 | 87.1 |
| 4/4 | 61.2 | **80.0** | **89.3** |
| 5/5 | **62.3** | 79.9 | 88.2 |
| 6/6 | 60.6 | 78.5 | 88.2 |

Table 3.7: Impact of $N_\theta$ and $N_\varphi$ on ScanNetV2 validation set

### 3.4.6 Ablation Study

In this section, we investigate Spherical Mask with ablation studies designed for its core components.

| $N_\theta / N_\varphi$ | mAP | $AP_{50}$ | $AP_{25}$ |
|---|---|---|---|
| 2/8 | 61.1 | 78.2 | 86.9 |
| 4/6 | 61.9 | 79.5 | 88.5 |
| 5/5 | 62.3 | 80.0 | 89.3 |
| 6/4 | **62.5** | **80.5** | **89.4** |
| 8/2 | 61.7 | 78.9 | 88.1 |

Table 3.8: Impact of $N_\theta$ and $N_\varphi$ on ScanNetV2 validation set

| Seed/Vote | mAP | $AP_{50}$ | $AP_{25}$ |
|---|---|---|---|
| 1024/128 | **62.3** | 79.2 | 87.8 |
| 1024/256 | **62.3** | **79.9** | **88.2** |
| 1024/512 | **62.3** | 79.4 | 87.6 |
| 2048/128 | 62.0 | 79.7 | 88.0 |
| 2048/256 | 61.8 | 79.6 | 87.9 |
| 2048/512 | 61.7 | 79.1 | 87.5 |

Table 3.9: Impact of seed and vote numbers on ScanNetV2 validation set

**Impact of $\theta$ and $\varphi$** is shown in Table 3.7 and 3.8. For this experiment, we fixed the backbone and the RPM with $L_{mc}$ and $L_{sc}$ and changed $N_\theta$ and $N_\varphi$. In theory, increasing $N_\theta$ and $N_\varphi$ should always produce better results. However, in practice, increasing them faces an issue of high complexity, as can be seen. Using 4/4 and 5/5 of $N_\theta$ and $N_\varphi$ improves the mAP for 0.9% and 1.7%, respectively. However, increasing them from 5/5 to 6/6 results in 2.8% of the performance drop, suggesting that too large $N_\theta$ and $N_\varphi$ make a negative impact on the performance due to increased complexity. For example, too large $N_\theta$ and $N_\varphi$ would result in many sectors without any foreground points, leading to biased target values for learning. Decreasing or increasing $N_\theta$ and $N_\varphi$ to an extreme, such as 2/8 and 8/2, results

in a performance drop. This is expected, as too few sectors on one axis increase complexity. Having 6/4 of $N_\theta$ and $N_\varphi$ shows a slight improvement of 0.2 in mAP, suggesting that increasing the number of sectors in the horizontal axis is slightly more helpful than increasing the number of sectors in the vertical axis.

Despite the variation, all configurations of $N_\theta$ and $N_\varphi$ still outperform existing methods in Table 3.2, implying that RID with RPM always improves the performance.

**Impact of Radial Point Migration** is illustrated in Table 3.6. Here, we investigate the impact of the RPM and each loss for it. The baseline is the model trained with only RID, excluding RPM. As the baseline model cannot refine the false positive points inside radial predictions or false negative points outside, as shown in Figure 3.8 (parts of chairs included as tables), its performance with high iou thresholds(AP, $AP_{50}$) are 17.1% and 13.1% worse than the full model. Including RPM with $L_{mc}$ leads to 13.3% of improvement in AP, suggesting refinement to push and pull misclassified points is crucial for the performance. $L_{sc}$ shows 9.4% improvement in AP from the baseline, suggesting that focusing on true positive samples also contributes significantly to learning fine granularity and common features shared within an instance. As shown in Figure 3.8(column 5,6), adding $L_{sc}$ improves the segmentation around the centre of the objects, which was expected as the true positive samples near the instance centres could be neglected from $L_{mc}$ as they are usually inside rays. As $L_{mc}$ and $L_{sc}$ target different samples, the full model combining both $L_{mc}$ and $L_{sc}$ shows 20.7% and 15.1% improvements for AP and $AP_{50}$, demonstrating the synergy of the two losses.

**Impact of Voting Parameters** is shown in Table 3.9. In this experiment, we change the seed and vote numbers inside the backbone while fixing RID and RPM. As can be seen, a seed number of 1024 produces more reliable results without variation than 2048, suggesting too many seed points negatively impact the system. Despite the slight difference, we observe that changing the seed and vote produces comparably small variations of ±0.6 in mAP for all settings.

### 3.4.7 Failure Cases

As can be seen in Table 3.1, The performance drop of Spherical Mask was observed in classes *Bed*, *Counter*, and *Cabinet* when compared with transformer-based SOTA methods, MAFT, and QueryFormer. These objects are usually very big and require larger receptive field architectures like transformers.

We also found a pattern of certain instances being located very close(< 20 cm) to each other, which can impose challenges in sharply segregating boundaries. For

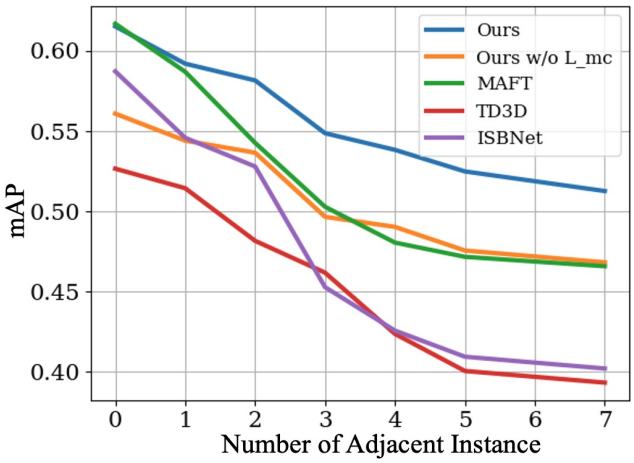



Figure 3.11

example, the top 3 categories found to have the same type of instances around the most are *Picture*, *Desk*, and *Bookshelf*. On average, a picture hangs with 2.05 pictures on a wall, a desk has 1.07 adjacent desks, usually in an office scene, and a bookshelf also has 1.29 bookshelves very close, usually in a library scene.

To further understand this, we plotted mAP conditioned on the number of close instances belonging to the same categories in Figure 3.11. The figure shows the drop in performance as the number of adjacent instances increases for all the methods.

**Comparison between RID and AABB:** The common characteristic of existing works with the coarse-to-fine approach is based on AABB. As one of the problems mentioned in Section 3.1, the excessive instance size estimation by AABB brings a burden to the refinement stage, as a bigger size box could include more background points to be refined. Therefore, tighter coarse detection is always advantageous for the refinement step, improving the overall performance of the system. As mentioned in Section (3.3.3), RID offers tighter boundaries for instances than AABB. To compare AABB and RID, we measure Signal-to-Noise Ratio for spaces occupied by foreground points compared to the spaces occupied by coarse detections. Specifically, we create a 3D grid, where each grid cell has $0.1m \times 0.1m \times 0.1m$ of space, and count two numbers: (1) $N_{\text{coarse}}$, the number of grid cells inside a coarse detection. (2) $N_{\text{foreground}}$, the number of grid cells occupied by foreground points. We then acquire SNR by dividing $N_{\text{foreground}}$ with $N_{\text{coarse}}$, so that the SNR effectively reflects how tight the boundary is. For example, the higher SNR indicates a tighter boundary, leading to better performance by making the refinement step simpler with fewer background points. Figure 3.12 shows the comparisons of AABB and RID with different $N_\theta$ and $N_\varphi$ for various instances in ScanNetV2 dataset. As can be seen, depending on the type of instance, the SNR increases from minimum


Figure 3.12: Visually comparing AABB and RID with different configurations of $N_\theta$ and $N_\varphi$ for various instances, such as bed, bathtub, chair, desk, sofa, toilet, and bookshelf. A larger signal-to-noise ratio(SNR) means fewer background points inside the coarse detection, leading to higher precision for the refinement stage and thus improving the overall system performance.

12% to maximum 50% from AABB when using 3/3 for $N_\theta/N_\varphi$(the second column), suggesting the advantage of RID over AABB. Increasing $N_\theta/N_\varphi$ from 3/3(column 2) to 5/5(column 3) leads to 22% higher SNR in average. The average SNR improves around 12% by increasing $N_\theta/N_\varphi$ from 5/5(column 3) to 7/7(column 4).

**Additional Discussion on Soft Margin Loss with Tanh** During our initial experiments, we found that using the original Soft Margin loss without tanh was found

to diverge approximately around 80 epochs. However, the proposed use of tanh stabilized the training and resulted in overall better results. When tested with the best-performing model(w/o tanh), it resulted in 57.7, 78.0, and 87.9 in mAP, $AP_{50}$, and $AP_{25}$, respectively on ScanNet, which is 4.6 lower in mAP compared to our final model.

**Additional Discussion on World Models** Spherical Mask employs Spherical coordinates for an individual object. More specifically, the centre of the object becomes the origin, and each object creates it's own local spherical coordinate. One of the practical advantages of Spherical coordinates over Cartesian coordinates is the reduced complexity for regression.

As we define radial sectors, our goal is to make the foreground points inside each sector move close to the origin (object center) while the background points move further away. The movement of each point can be simply performed by only changing the distance parameter in Spherical coordinates without considering the other two parameters for angles, reducing the regression complexity from 3 dimensions to 1 dimension for learning. On the other hand, in Cartesian coordinates, moving points always requires offset prediction in 3D (x,y,z).

## 3.5 Recent Works after Publication

After its publication with the title "Spherical Mask: Coarse-to-Fine 3D Point Cloud Instance Segmentation with Spherical Representation" in Conference on Computer Vision and Pattern Recognition (CVPR) 2024, there have been three relative works: OneFormer3D [80], EASE [137] and SGIFormer [186].

### 3.5.1 OneFormer3D: One Transformer for Unified Point Cloud Segmentation

OneFormer3D consistently performs instance and semantic segmentation with a group of kernels based on Transformer. This is different from previous methods that use a single kernel [158]. Compared to SphericalMask, OneFormer3D is provided with more semantic information using the explicit supervision for semantic segmentation. Also, using the whole input points without sampling combined with a larger receptive field of the transformer allows the model to learn detail information of the instances.

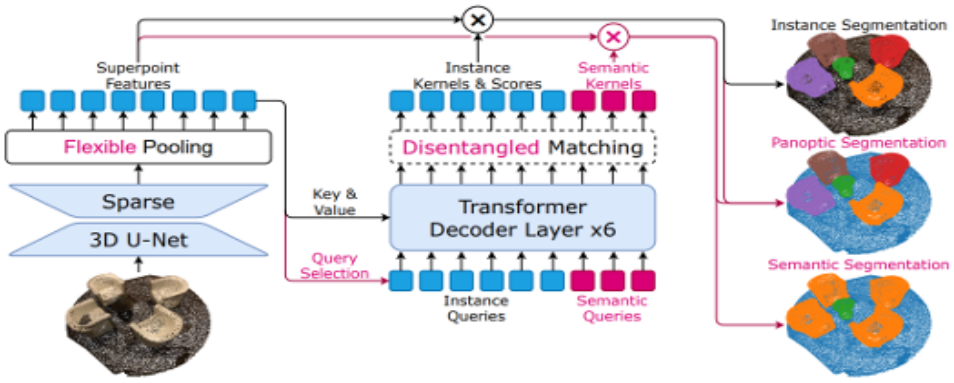



Figure 3.13: Flow diagram of OneFormer3D. Based on the transformer architecture, the model first extracts the Superpoint features in the point cloud, and learns Instance Segmentation, Panoptic Segmentation, and Semantic Segmentation simultaneously.

### 3.5.2 Edge-Aware 3D Instance Segmentation Network with Intelligent Semantic Prior (EASE)

EASE employs a powerful language model, CLIP [129], to feed the rich semantic knowledge as a prior. Additionally, motivated by the fact that existing methods often struggle to accurately segment edges, EASE introduces Edge Prediction Module. Compared to SphericalMask, EASE aims to focus more on learning the semantic information of each class by incorporating the embedding of each class from CLIP, as can be seen in Figure 3.13. Specifically, the embedding containing rich semantics from CLIP is fused with attention output inside the Mask Transformer's architecture [141] to encourage the learning of the semantics.

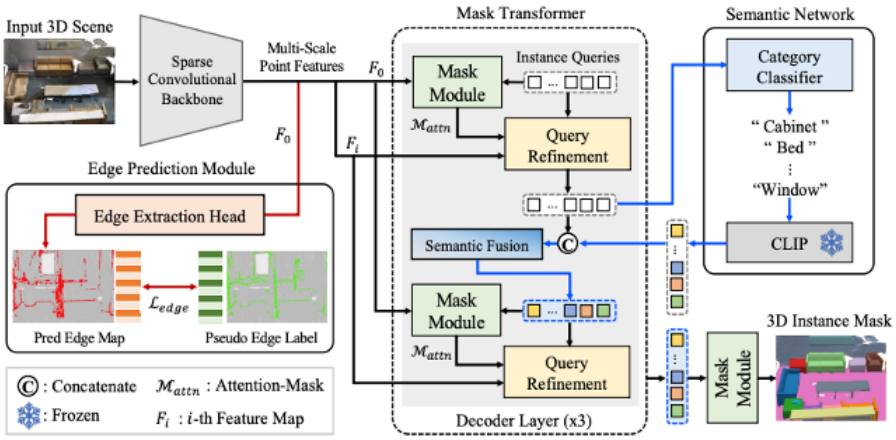



Figure 3.14: Flow diagram of EASE [137].

### 3.5.3 SGIFormer: Semantic-guided and Geometric-enhanced Interleaving Transformer for 3D Instance Segmentation

SGIFormer addresses the issue of query initialisation problem of Transformer-based method [141] by introducing an implicit scene aware query initialisation approach. More specifically, two new modules, such as Semantic-guided Mix Query (SMQ) initialization and the Geometric-enhanced Interleaving Transformer (GIT) decoder, are introduced. SMQ facilitates voxel-wise semantic label prediction, where the initial query set can be selected from voxels that have high class confidence scores. In GIT, the query points belonging to the same instances are optimized to have similar features, which can help the model catch geometric information of each instance. Although the architecture is based on the transformer and the focus of the work is on learning semantics, GIT has some similarities to SphericalMask. For instance, RPM module of SphericalMask and GIT of SGIFormer both learn to make the points inside each instance closer to each other. However, the difference is that RPM module is used as the final prediction mask, whereas GIT is used as a module before the final prediction to encourage the learning of the geometric understanding of instances.

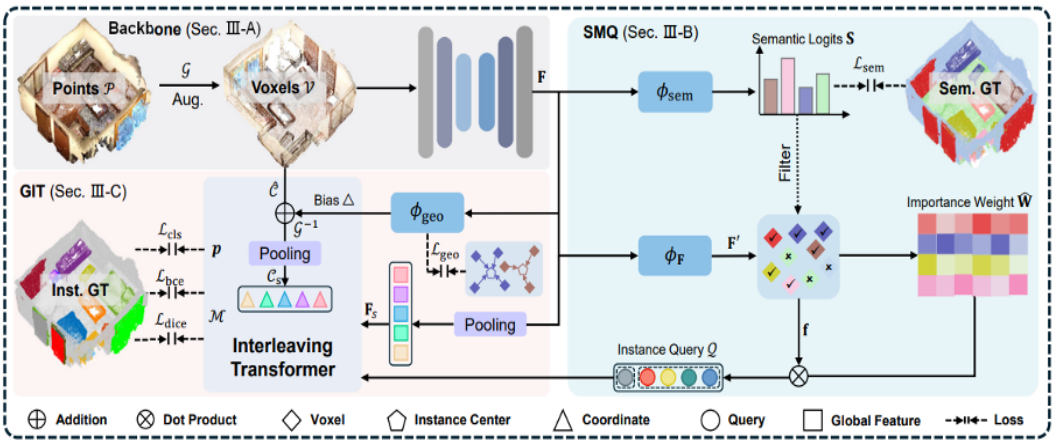



Figure 3.15: Flow diagram of SGIFormer [186]

## 3.6 Conclusion

This section presents Spherical Mask, a novel coarse-to-fine approach for 3D instance segmentation in point cloud. As a coarse detection, the RID module finds instances as 3D polygons defined with centre and rays. In contrast to existing coarse-to-fine approaches, the RPM module uses the polygons as soft references and migrates points efficiently in spherical coordinates to acquire final masks.

We demonstrate how each module contributes to the instance segmentation and achieves state-of-the-art performances on public benchmarks ScanNet-V2, S3DIS, and STPLS3D.

In the next chapter, we address the issue of high annotation cost for object localisation in changing environments. Specifically, we extend the environment of object localisation from per-frame to consecutive frames to utilise motion to find moving objects along with the technique that we have explored in this chapter, as shown in Figure 1.2. Then, the found objects can be used as pseudo-labels for training deep learning-based object detection models.

# Chapter 4

# Sample, Crop, Track: Self-Supervised Mobile 3D Object Detection for Urban Driving LiDAR

This chapter focuses on how to minimise the expensive annotation cost for object localisation based on deep learning. In particular, despite the superior performance of deep learning-based object detection, the limitation in the generalisation ability of the model hinders the real-world application of object detection models. We first start by motivating the problem, followed by introducing the pseudo-label collection approach using motion cues to minimise the cost.

In recent years, deep learning has led to great progress in the detection of mobile (i.e. movement-capable) objects in urban driving scenes. Supervised approaches typically require the annotation of large training sets; there has thus been great interest in leveraging weakly, semi- or self-supervised methods to avoid this, with much success. Whilst weakly and semi-supervised methods require some annotation, self-supervised methods have used cues such as motion to relieve the need for annotation altogether. However, a complete absence of annotation typically degrades their performance, and ambiguities that arise during motion grouping can inhibit their ability to find accurate object boundaries. In this chapter, we propose a new self-supervised mobile object detection approach called SCT. This uses both motion cues and expected object sizes to improve detection performance and predicts a dense grid of 3D-oriented bounding boxes to improve object discovery. We significantly outperform the state-of-the-art self-supervised mobile object detection method TCR on the KITTI tracking benchmark and achieve performance that is within 30% of the fully supervised PV-RCNN++ method for IoUs <= 0.5.

## 4.1 Introduction

Recent years have seen significant improvements in the accuracy and robustness of 3D object detectors, driven by the ever-improving capabilities of Deep Neural Networks (DNNs) and the commercial importance of tasks such as detecting vehicles, cyclists and pedestrians for autonomous driving. However, despite the success of DNNs, labelling the large datasets that fully supervised methods typically require has remained cumbersome and tedious.

Many recent semi-supervised [163] and weakly-supervised [207, 29, 179, 28, 197, 119, 46] methods, which annotate only part of the dataset or use only high-level labels, have been proposed to reduce this burden. By contrast, self-supervised approaches are less common, owing to the difficulty of detecting objects without any annotation. One recent work that does use self-supervision is TCR [53], which groups nearby points with similar motions and applies sophisticated data augmentation to learn to detect mobile (i.e. movement-capable) objects in both 2D and 3D space. However, this approach risks generating inaccurate pseudo-ground truth boxes when e.g. two nearby objects move in the same way. Moreover, without a prior notion of the expected object sizes, noise in the estimated motions can lead points to be incorrectly included/excluded from an object or clusters of background points with similar motions to be undesirably treated as objects of interest.

In this chapter, we address these issues by using the expected object sizes in the training set (which we require up-front) and temporal consistency in fitting boxes across multiple frames to filter out poor pseudo-ground truth boxes. By considering the expected object sizes when grouping points, we can avoid generating pseudo-ground truth boxes of inappropriate sizes, such as those that cover multiple nearby objects with similar motions. Considering the temporal consistency of fitted boxes across multiple frames helps to filter out further poor pseudo-ground truth boxes, such as those that are of an acceptable size but overlap more than one object. Lastly, to generate accurate pseudo-ground truth boxes for closely located objects and objects that exhibit limited movement during training, we allow our model to predict a dense output image in which each pixel denotes a candidate box (this contrasts with TCR [53], which only considers a subset of the output pixels as candidates).

**Contributions: 1)** We use the expected sizes of objects in the training set and temporal consistency in fitting boxes across multiple frames to filter out poor pseudo-ground truth boxes. **2)** We predict a dense output image in which each pixel denotes a candidate box, improving the accuracy of the pseudo-ground truth

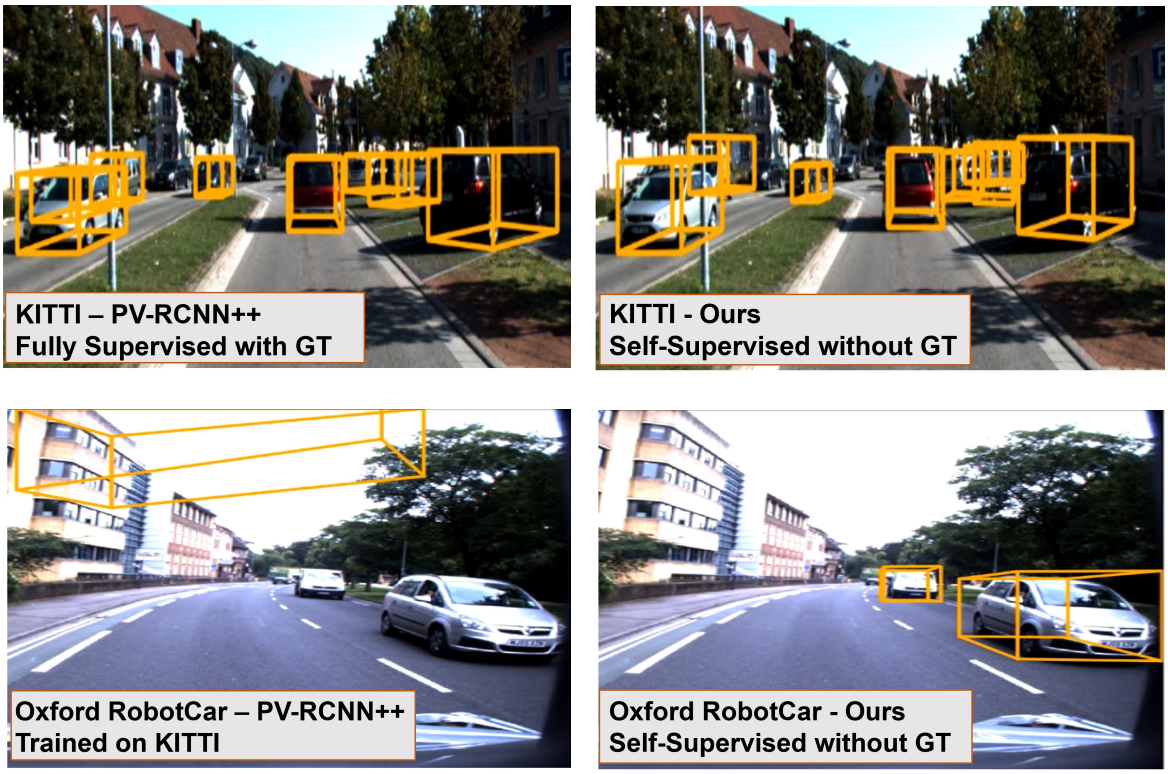



Figure 4.1: Comparing our method with PV-RCNN++ [147] on the KITTI [49] and Oxford RobotCar [110] datasets. For KITTI, where ground-truth annotations are available, the fully-supervised PV-RCNN++ outperforms our self-supervised approach. For RobotCar, no ground-truth annotations are available for training PV-RCNN++, and the KITTI-trained model fails to generalise to the different LiDAR setups used. By contrast, our self-supervised method does not use ground-truth annotations and can thus be trained on RobotCar, qualitatively outperforming PV-RCNN++ for this dataset.

boxes our model can generate. **3)** We propose a novel neural-guided sampling process to choose appropriate candidate pixels from the dense output image to both improve the efficiency of the training process, and guide the network towards dynamic objects during training.

## 4.2 Related Work

Many approaches have been proposed to tackle urban driving 3D object detection over the years. Our own method is entirely self-supervised, so for brevity, we focus only on the self-supervised literature in this chapter. For recent reviews that survey the broader field, see [4, 174].

Early work in this area focused on grouping regions in an image to find if they were the same type as an object of interest [139, 151, 170]. The recent success in deep learning has led to many new works such as AIR [40], MONet [10], GENE-

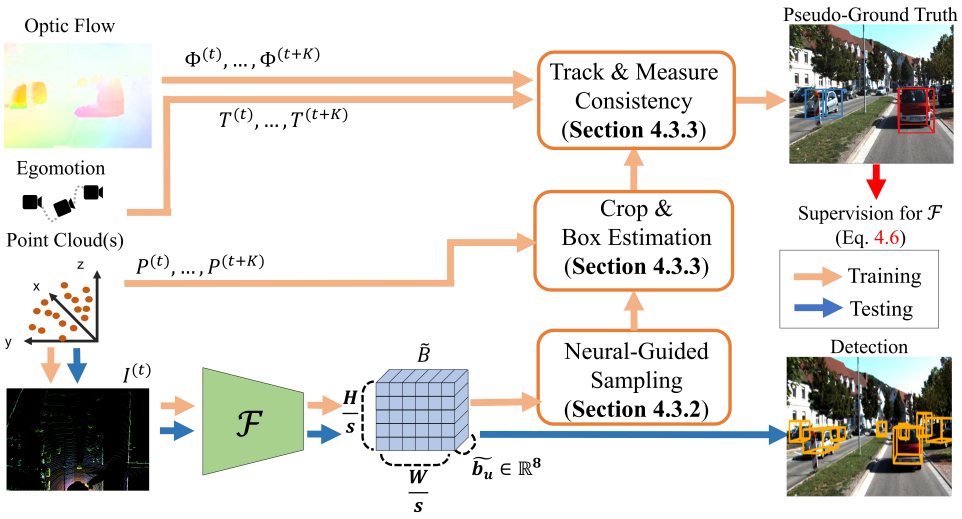



Figure 4.2: Our proposed approach (see §4.3). During training, we utilise cues from multiple successive frames to find potential moving objects. The input to $\mathcal{F}$ is PointPillar [85] based 3D LiDAR point cloud $P$. In order to find candidate regions that contain mobile objects, we first sample some regions in the dense grid that represent the voxelised point cloud. For efficient sampling, we utilised confidence scores from the detection model (Neural-Guided Sampling). The candidate regions are then cropped using box estimation. Lastly, to investigate the reliability of the box and filter out false positives, the box is tracked for successive frames to check its consistency. After the process, the survived box is used as a pseudo-label to train the detection model. For inference, only a point cloud is needed.

SIS [38], IODINE [51], Slot Attention [106] and SCALOR [70]. Most of these perform image reconstruction, in some cases using "slots" that are learned to represent an object or a part of an object using generative modelling. All these methods have proved to work well on simple datasets but have struggled to handle the complexities of real-world data [53, 5]. In very recent work, inspired by slot attention techniques, Bao et al. [5] used a weak motion segmentation algorithm to detect multiple mobile objects in synthesised photorealistic 2D images; however, the KITTI training code and models for this approach have not been publicly released.

Another line of research, which is more related to this work, uses motion cues to detect objects, assuming that the regions corresponding to each object should share the same motion. The early works in this direction combined the affinity matrix from point trajectories based on 2D motions in a video and standard clustering techniques to detect objects [8, 43]. These works assumed the camera to be static. In more recent work, Chen et al. [21] used range images from a moving camera and processed them through CNNs to detect moving regions by labelling range data. In another recent work, *Track, Check, Repeat* [53] (TCR) proposed an Expectation Maximisation (EM) approach for mobile object detection in the context of autonomous driving by using other information such as ego-motion, 2D

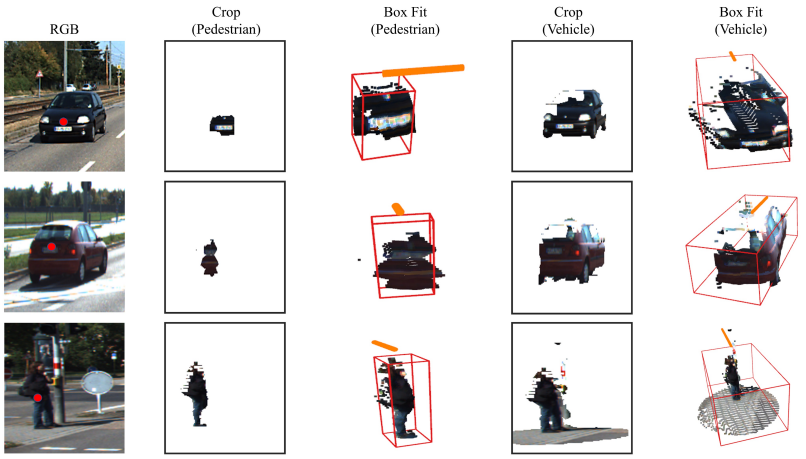



Figure 4.3: Cropping and box fitting examples. The red dot on each RGB image indicates a chosen pixel $\mathbf{u}$ in $\tilde{B}^{(t)}$. As per §4.3.3, we crop a set of points from $P^{(t)}$ around the centre $\tilde{\mathbf{c}}_{\mathbf{u}}^{(t)}$ of the bounding box denoted by $\mathbf{u}$, for each anchor, and then fit a 3D oriented bounding box to the cropped points. The pedestrian anchor is too small to capture the entirety of the cars, whereas the vehicle anchor performs far better; the converse is true for the pedestrian example, where the vehicle anchor captures too much background. Orange lines show the orientations of the fitted boxes.

optic flow and 3D LiDAR data. Compared to the proposed work in the thesis, TCR uses multiple stages of training both image and point based detection networks, requiring different parameters for each stage. This is because the detection networks are sensitive to false positive pseudo labels, which require different thresholds for detection confidence scores to be considered as pseudo labels. On the other hand, Sample Crop Track only requires one stage training by utilizing the additional cue of expected object sizes for each class.

## 4.3 Method

### 4.3.1 Overview

Our method attempts to use self-supervision to train a model $\mathscr{F}$[1] that can predict oriented bounding boxes around mobile objects in a 3D urban driving scene (see Figure 4.2) based on PointPillars [85][2]. The output of $\mathscr{F}$ is a (downsampled) image $\tilde{B} \in \mathbb{R}^{\frac{H}{s} \times \frac{W}{s} \times 8}$, in which $s = 4$ is the downsampling factor. Each pixel $\mathbf{u}$ in $\tilde{B}$ contains

[1] Our $\mathscr{F}$ uses an encoder-decoder architecture, in which the encoder is based on Feature Pyramid Network (FPN) [96] and ResNet-18 [55], and the decoder is based on CenterNet [36] (without any pre-trained weights).

[2] In practice, $H = W = 608$. The grid is axis-aligned, and corresponds to a 3D volume in LiDAR space covering points $(x, y, z) \in P$ that satisfy 2.5m $\leq x \leq$ 40m, -18m $\leq y \leq$ 18m and -2.73m $\leq z \leq$ 1.27m. The three values for each {pixel/grid cell} are based on the heights, intensities, and density of the points from $P$ in the corresponding grid cell.

an 8-tuple $\tilde{\mathbf{b}}_{\mathbf{u}} = (\tilde{\boldsymbol{\delta}}_{\mathbf{u}}, \tilde{\mathbf{d}}_{\mathbf{u}}, \tilde{r}_{\mathbf{u}}, \tilde{\kappa}_{\mathbf{u}})$ denoting a predicted 3D bounding box in KITTI [49] format. In this, $\tilde{\mathbf{d}}_{\mathbf{u}} \in \mathbb{R}^3$ denotes the (axis-aligned) dimensions of the box, $\tilde{r}_{\mathbf{u}} \in [-\pi, \pi]$ its rotation angle (yaw), and $\tilde{\kappa}_{\mathbf{u}} \in [0,1]$ its confidence score. The centre $\tilde{\mathbf{c}}_{\mathbf{u}}$ of the bounding box is specified indirectly as an offset $\tilde{\boldsymbol{\delta}}_{\mathbf{u}} \in \mathbb{R}^3$ from the centre of the vertical pillar in the grid that corresponds to $\mathbf{u}$.[3]

Note that at training time, our method also requires access to sparse depth images $\{D^{(t)} \in \mathbb{R}^{H' \times W'}\}$, optic flow images $\{\Phi^{(t)} \in \mathbb{R}^{H' \times W' \times 2}\}$ and 6DoF poses $\{T_t^w \in \mathbb{SE}(3)\}$, expressed as transformations from camera space at frame $t$ to world space $w$. These are typically available (or easily obtained) in the urban driving setting on which we focus.[4]

We draw inspiration from a recent study [53] that showcased the possibility of training a model using only the labels of actively moving objects and appropriate data augmentation to detect mobile objects with a similar shape, regardless of whether or not they are moving at the time. In our case, the set of shapes we expect objects to have must be specified in advance as a set of anchors $A$. Each anchor $\mathbf{a} = (a_x, a_y, a_z) \in A$ denotes an axis-aligned bounding box (AABB) of size $a_x \times a_y \times a_z$.[5] To train $\mathscr{F}$ to detect mobile objects without manual annotation, we propose a multi-step process that is run for each frame $(P^{(t)}, I^{(t)})$ in the training set to generate pseudo-ground truth 3D bounding boxes that can be used to train our model. First, we compute $\tilde{B}^{(t)} = \mathscr{F}(I^{(t)})$. Next, we use a neural-guided sampling process (see §4.3.2) to choose a subset of the pixels from $\tilde{B}^{(t)}$ to use for self-supervision. Each chosen pixel $\mathbf{u}$ denotes a predicted bounding box $\tilde{\mathbf{b}}_{\mathbf{u}}^{(t)}$. We then aim to generate (see §4.3.3) a pseudo-ground truth box $\mathbf{b}_{\mathbf{u}}^{(t)}$ corresponding to each $\tilde{\mathbf{b}}_{\mathbf{u}}^{(t)}$ that can be compared to it as part of the loss (see §4.3.4). We now describe this entire process in more detail.

### 4.3.2 Sampling

As mentioned in §4.3.1, the overall goal of our self-supervised approach is to generate pseudo-ground truth boxes for a subset of the pixels in $\tilde{B}^{(t)}$, which can then

[3] This is important, as it implicitly constrains the centres (indirectly) predicted by the network at the start of training to be within a reasonable distance of the relevant pillars.

[4] We constructed each $D^{(t)}$ by projecting the points in the corresponding point cloud $P^{(t)}$ down into the image plane (using first the fixed, pre-calibrated transformation $S_L^C$ from LiDAR to camera space, and then perspective projection based on the known camera intrinsics). To obtain the optic flow images, we followed TCR [53] in training a RAFT [156] model on MPI Sintel Flow [11]. For our experiments on the KITTI tracking benchmark [49], $H'$ was 375, and $W'$ was 1242. We computed the 6DoF poses from the OxTS GPS/IMU data provided by the benchmark.

[5] We used three anchors: pedestrian, cyclist and vehicle. Our pedestrian anchor $\mathbf{a}_p = (0.45\text{m}, 1.70\text{m}, 0.27\text{m})$ was obtained from [9]. Our cyclist anchor $\mathbf{a}_c = (0.54\text{m}, 1.90\text{m}, 1.75\text{m})$ was derived by adding half the height of the pedestrian anchor to the average dimensions from [107]. Our vehicle anchor $\mathbf{a}_v = (1.88\text{m}, 1.63\text{m}, 4.58\text{m})$ was obtained from [117].

be used to supervise the training of our model. Empirically, we found generating a pseudo-ground truth box for every pixel in $\tilde{B}^{(t)}$ to be prohibitively costly ($\approx$ 1 hour/frame); we thus propose a neural-guided approach to choose a subset of the pixels in $\tilde{B}^{(t)}$ to use for self-supervision. Our approach aims to select a mixture of some pixels whose boxes have high confidence scores predicted by the model and some whose boxes have lower confidence scores.[6] To achieve this, we first use a threshold $\tau = 0.08$ to divide the pixels into two sets – those whose confidence $\kappa_{\mathbf{u}}^{(t)} > \tau$, and those whose confidence $\kappa_{\mathbf{u}}^{(t)} \leq \tau$ – and then uniformly sample 15 pixels from each set, resulting in total 30 pixels. The confidence score $\tau$ was selected by referring to the implementation of TCR [53], and selected the value that shows the best performance among 0.08, 0.25, and 0.5. We chose 30 pixels for subsampling as this is the maximum size that we could choose with the restriction in the computation resource (i.e. GPU memory).

Rather than directly using the confidences predicted by the network for these boxes, we then compute more robust confidence for each box by averaging the confidences of the box itself and the 8 boxes from $\tilde{B}^{(t)}$ whose centres are the nearest neighbours of the box's centre in 3D. This makes the confidence values for the boxes more robust to local noise.

### 4.3.3 Pseudo-Ground Truth Box Generation

Each pixel $\mathbf{u}$ from $\tilde{B}^{(t)}$ chosen as described in §4.3.2 denotes a predicted bounding box $\tilde{\mathbf{b}}_{\mathbf{u}}^{(t)}$ centred on the 3D point $\tilde{\mathbf{c}}_{\mathbf{u}}^{(t)}$. For each such $\mathbf{u}$, we now seek to generate a corresponding pseudo-ground truth box $\mathbf{b}_{\mathbf{u}}^{(t)}$ that can be compared to $\tilde{\mathbf{b}}_{\mathbf{u}}^{(t)}$ as part of the loss (see §4.3.4). The sizes of the pseudo-ground truth boxes we generate will be guided by the initial set of anchors (expected object sizes) provided by the user (see §*Cropping*).

To generate the boxes, we proceed as follows. For each $\mathbf{u}$ and each anchor $\mathbf{a}$, we crop a set of points $Q_{\mathbf{u},\mathbf{a}}^{(t)}$ from $P^{(t)}$ around $\tilde{\mathbf{c}}_{\mathbf{u}}^{(t)}$ (see §*Cropping*). We then track the points in $Q_{\mathbf{u},\mathbf{a}}^{(t)}$ forwards for $K$ frames, resulting in $K+1$ sets of points $\{Q_{\mathbf{u},\mathbf{a}}^{(t\rightsquigarrow k)} : 0 \leq k \leq K\}$ for each $\mathbf{u}$ and $\mathbf{a}$ (see §*Tracking*). Next, we fit a 3D oriented bounding box (OBB) to each $Q_{\mathbf{u},\mathbf{a}}^{(t\rightsquigarrow k)}$, and then use these boxes to choose the best anchor $\mathbf{a}^\star$ (if any) for each $\mathbf{u}$. For any pixel $\mathbf{u}$ with a valid best anchor $\mathbf{a}^\star$, we treat the box fitted to $Q_{\mathbf{u},\mathbf{a}^\star}^{(t\rightsquigarrow 0)}$ as the pseudo-ground truth for $\mathbf{u}$, convert it into the correct space and denote it as $\mathbf{b}_{\mathbf{u}}^{(t)}$ (see §*Box Fitting and Selection*).

[6] Sampling high-confidence boxes allows us to continually learn how to predict boxes for the same objects over time; conversely, sampling low-confidence boxes allows us to discover new objects in the scene.

**Cropping:** For a chosen pixel $\mathbf{u}$ in $\tilde{B}^{(t)}$, denoting a predicted box with centre $\tilde{\mathbf{c}}_{\mathbf{u}}^{(t)} = (c_x, c_y, c_z)$, we first crop a set of points $Q_{\mathbf{u},\mathbf{a}}^{(t)}$ from $P^{(t)}$ in a vertical cylinder[7] around $\tilde{\mathbf{c}}_{\mathbf{u}}^{(t)}$ for each user-provided anchor $\mathbf{a} = (a_x, a_y, a_z) \in A$. The points in $P^{(t)}$ are in the space defined by the LiDAR at frame $t$; for convenience, we convert them into camera space at frame $t$ by applying the (fixed, pre-calibrated) transformation $S_L^C \in \mathbb{SE}(3)$ from LiDAR to camera space. Formally, $Q_{\mathbf{u},\mathbf{a}}^{(t)}$ can then be defined as

$$\begin{aligned} Q_{\mathbf{u},\mathbf{a}}^{(t)} = \{ & S_L^C \mathbf{p} : \mathbf{p} = (p_x, p_y, p_z) \in P^{(t)} \\ & \text{and } |p_y - c_y| < \frac{a_y}{2} \\ & \text{and } \left\| (p_x - c_x, p_z - c_z) \right\|_2 < \frac{\|(a_x, a_z)\|_2}{2} \Big\}. \end{aligned} \tag{4.1}$$

**Tracking:** We next track the points in $Q_{\mathbf{u},\mathbf{a}}^{(t)}$ forwards for $K$ frames to help us to determine the best anchor (if any) for each $\mathbf{u}$ (see §4.3.3). We denote the point track for an individual point $\mathbf{q} \in Q_{\mathbf{u},\mathbf{a}}^{(t)}$ as $\{\chi_0(\mathbf{q}), \ldots, \chi_K(\mathbf{q})\}$. To calculate it, we first set $\chi_0(\mathbf{q})$ to $\mathbf{q}$, and then iteratively compute each $\chi_{k+1}(\mathbf{q})$ from the preceding $\chi_k(\mathbf{q})$ as follows. First, we project $\chi_k(\mathbf{q})$, which is in camera space, down into the image plane via $\mathbf{u}_k = \pi(\chi_k(\mathbf{q}))$, in which $\pi$ denotes perspective projection. Then we compute the corresponding pixel in frame $t+k+1$, namely $\mathbf{u}'_{k+1} = \mathbf{u}_k + \Phi^{(t+k)}$, find the nearest pixel $\mathbf{u}''_{k+1}$ to it that has a valid depth in $D^{(t+k+1)}$, and back-project this into camera space to obtain $\chi_{k+1}(\mathbf{q}) = \pi^{-1}\left(\mathbf{u}''_{k+1}, D^{(t+k+1)}\right)$.[8] We then use the point tracks for all the points in $Q_{\mathbf{u},\mathbf{a}}^{(t)}$ to project $Q_{\mathbf{u},\mathbf{a}}^{(t)}$ forwards into the $K$ subsequent frames, via

$$Q_{\mathbf{u},\mathbf{a}}^{(t \rightsquigarrow k)} = \begin{cases} Q_{\mathbf{u},\mathbf{a}}^{(t)} & \text{if } k = 0, \\ \{T_{t+k}^t \chi_k(\mathbf{q}) : \mathbf{q} \in Q_{\mathbf{u},\mathbf{a}}^{(t)}\} & \text{if } 0 < k \leq K. \end{cases} \tag{4.2}$$

Note that we transform all of the points into frame $t$ here using $T_{t+k}^t = (T_t^w)^{-1} T_{t+k}^w$ since we want all of the $Q_{\mathbf{u},\mathbf{a}}^{(t \rightsquigarrow k)}$ point sets to be in the same frame of reference.

**Box Fitting and Selection:** Having constructed the point sets as described in §4.3.3, we next fit to each point set $Q_{\mathbf{u},\mathbf{a}}^{(t \rightsquigarrow k)}$ a 3D oriented bounding box $\mathbf{b}_{\mathbf{u},\mathbf{a}}^{(t \rightsquigarrow k)}$ that has centre $\mathbf{c} \in \mathbb{R}^3$, dimensions $\mathbf{d} \in \mathbb{R}^3$ and rotation angle (yaw) $r \in [-\pi, \pi]$. We compute $\mathbf{c}$ as the mean of the points in $Q_{\mathbf{u},\mathbf{a}}^{(t \rightsquigarrow k)}$. To find $r$, we first apply Principal Component Analysis (PCA) to $Q_{\mathbf{u},\mathbf{a}}^{(t \rightsquigarrow k)}$, and let $\mathbf{e} = (e_x, e_y, e_z) \in \mathbb{R}^3$ denote its first eigenvector. We then compute $r = \arctan(e_z / e_x)$, as our camera coordinate system has $x$ right, $y$ down and $z$ forward. We approximate the desired dimensions $\mathbf{d}$ as those of the smallest axis-aligned bounding box (AABB) that encloses the set of points $S = \{\mathcal{T}\mathbf{q} : \mathbf{q} \in Q_{\mathbf{u},\mathbf{a}}^{(t \rightsquigarrow k)}\}$, where $\mathcal{T} \in \mathbb{SE}(3)$ denotes a rigid transformation that translates $\mathbf{c}$ to the origin and rotates by $-r$ around the $y$ axis. Then $\mathbf{d}$ can be trivially calculated by first

[7] Thus obviating the need to know the yaw of the object *a priori*.

[8] This approach can in some cases yield a $\chi_{k+1}(\mathbf{q})$ that is not a good match for $\chi_k(\mathbf{q})$, e.g. if $\mathbf{u}''_{k+1}$ is far from $\mathbf{u}'_{k+1}$. However, since only sets of boxes based on correct point tracks will survive the consistency scoring in §4.3.3, we did not observe this to cause a problem in practice.

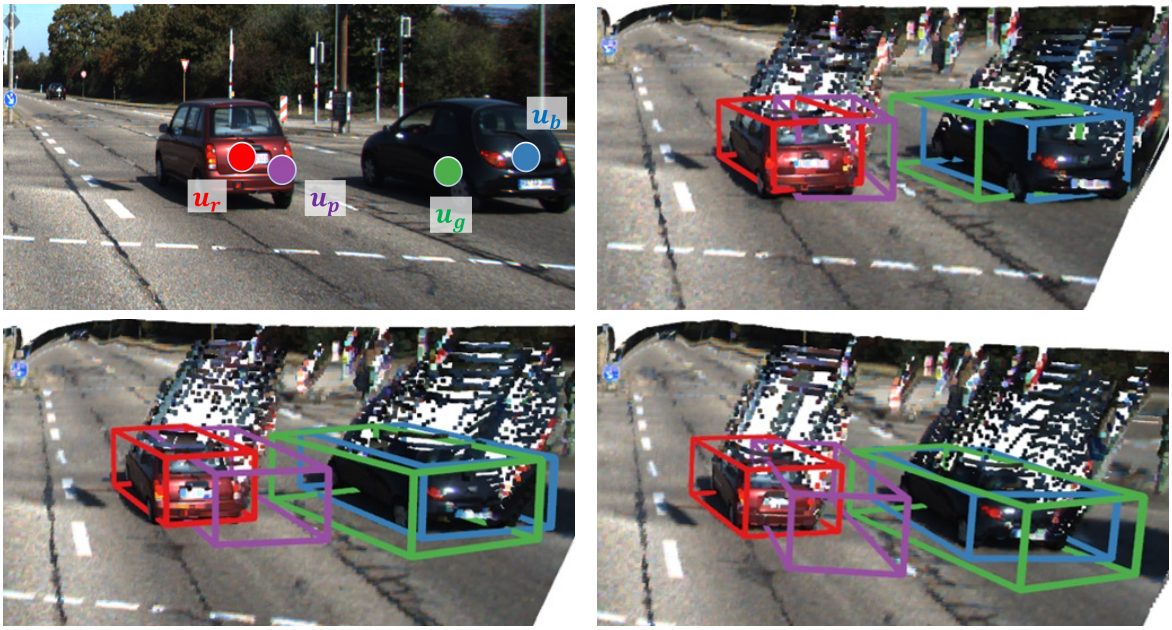



Figure 4.4: The use of temporal consistency to decide between pseudo-ground truth box candidates for objects. Left-to-right, top-to-bottom: choosing candidate pixels $U = \{\mathbf{u}_r, \mathbf{u}_p, \mathbf{u}_g, \mathbf{u}_b\}$; the fitted boxes $\mathbf{b}_{\mathbf{u},\mathbf{a}_v}^{(t \rightsquigarrow k)}$ for $\mathbf{u} \in U$, the vehicle anchor $\mathbf{a}_v$ and $0 < k \leq 3$. The shapes of the red and blue boxes are more consistent over time than those of the purple and green ones, and so $\mathbf{b}_{\mathbf{u}_r}^{(t)}$ and $\mathbf{b}_{\mathbf{u}_b}^{(t)}$ will be used as the pseudo-ground truths for the cars.

defining $min_i = \min_{\mathbf{s} \in S} s_i$ and $max_i = \max_{\mathbf{s} \in S} s_i$ for all $i \in \{x, y, z\}$, where $s_i$ denotes the $i^{th}$ component of $\mathbf{s}$, and then computing $\mathbf{d} = (max_x - min_x, max_y - min_y, max_z - min_z)$.

For each chosen pixel $\mathbf{u} \in \tilde{B}^{(t)}$, we will then have constructed a set of 3D oriented bounding boxes $\{\mathbf{b}_{\mathbf{u},\mathbf{a}}^{(t \rightsquigarrow k)} : 0 \leq k \leq K\}$ for each anchor $\mathbf{a} \in A$. We now seek to choose the best anchor $\mathbf{a}^\star$ (if any) to associate with each $\mathbf{u}$, which will in turn allow us to define a pseudo-ground truth bounding box $\mathbf{b}_{\mathbf{u}}^{(t)}$ to associate with $\mathbf{u}$ for training purposes. We will denote the set of chosen pixels for which best anchors can be found as $U_+^{(t)}$, and the remaining chosen pixels as $U_-^{(t)}$. To determine $\mathbf{b}_{\mathbf{u}}^{(t)}$, we initially score the set of boxes associated with each $\mathbf{u}$ and $\mathbf{a}$ using two criteria: (i) the extent to which the set of boxes indicates a moving object, and (ii) the extent to which the dimensions of the boxes in the set remain consistent over time. (Note that we want to generate pseudo-ground truth boxes that correspond to moving objects and are temporally consistent.) The first criterion can be specified as

$$moving_{\mathbf{u},\mathbf{a}}^{(t)} = \sum_{k=1}^{K} \left\| \mathbf{c}_{\mathbf{u},\mathbf{a}}^{(t \rightsquigarrow k)} - \mathbf{c}_{\mathbf{u},\mathbf{a}}^{(t \rightsquigarrow (k-1))} \right\|_2, \tag{4.3}$$

in which $\mathbf{c}_{\mathbf{u},\mathbf{a}}^{(t \rightsquigarrow k)}$ denotes the centre of $\mathbf{b}_{\mathbf{u},\mathbf{a}}^{(t \rightsquigarrow k)}$. This computes the distance moved by the object over the $K+1$ frames considered. The second criterion can be specified

| | Eval. | mAP@IoU | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | *0.1* | *0.2* | *0.3* | *0.4* | *0.5* | *0.6* | *0.7* |
| **Self-Supervised** | | | | | | | | |
| Slot Attention [106] | 2D | 0.07 | 0.03 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 |
| SCALOR [70] | 2D | 0.11 | 0.07 | 0.02 | 0.00 | 0.00 | 0.00 | 0.00 |
| TCR [53] | 2D | 0.43 | 0.40 | 0.37 | 0.35 | 0.33 | 0.30 | 0.21 |
| TCR$^\star$ [53] | 2D | 0.60 | 0.59 | 0.59 | 0.56 | 0.49 | 0.40 | 0.23 |
| Ours | 2D | **0.79** | **0.77** | **0.74** | **0.72** | **0.66** | **0.59** | **0.41** |
| TCR [53] | BEV | 0.40 | 0.38 | 0.35 | 0.33 | 0.31 | 0.23 | 0.06 |
| TCR$^\star$ [53] | BEV | 0.58 | 0.58 | 0.54 | 0.48 | 0.42 | 0.29 | 0.10 |
| Ours | BEV | **0.79** | **0.79** | **0.76** | **0.74** | **0.69** | **0.62** | **0.33** |
| **Semi-Supervised** | | | | | | | | |
| *3DIoUMatch [163]* | *BEV* | *0.84* | *0.84* | *0.84* | *0.84* | *0.84* | *0.84* | *0.83* |
| **Fully Supervised** | | | | | | | | |
| *PV-RCNN++ [147]* | *BEV* | *0.98* | *0.98* | *0.98* | *0.97* | *0.97* | *0.95* | *0.95* |

Table 4.1: Comparing our method's ability to discover mobile objects to that of other methods on the KITTI tracking benchmark [49]. We compare to a number of recent methods, of which three are self-supervised, one is semi-supervised (with annotations for 10% of the dataset), and one is fully supervised. TCR denotes the original version described in [53]; TCR$^\star$ denotes the improved version published on GitHub. (We italicise the semi-supervised and fully supervised methods, which are not directly comparable as their access to annotations gives them a significant advantage.)

as

$$inconsistency^{(t)}_{\mathbf{u},\mathbf{a}} = \sum_{k=1}^{K} \left\| \mathbf{d}^{(t\rightsquigarrow k)}_{\mathbf{u},\mathbf{a}} - \mathbf{d}^{(t\rightsquigarrow 0)}_{\mathbf{u},\mathbf{a}} \right\|_2, \tag{4.4}$$

in which $\mathbf{d}^{(t\rightsquigarrow k)}_{\mathbf{u},\mathbf{a}}$ denotes the dimensions of $\mathbf{b}^{(t\rightsquigarrow k)}_{\mathbf{u},\mathbf{a}}$. This sums variations from the initial box in the set. We combine both criteria into a single confidence score for each set of boxes via

$$\kappa^{(t)}_{\mathbf{u},\mathbf{a}} = \lambda_1 \times moving^{(t)}_{\mathbf{u},\mathbf{a}} - \lambda_2 \times inconsistency^{(t)}_{\mathbf{u},\mathbf{a}}, \tag{4.5}$$

empirically setting $\lambda_1 = 0.4$ and $\lambda_2 = 0.15$.

To try to choose a best anchor $\mathbf{a}^\star$ for $\mathbf{u}$, we now first disregard any anchor $\mathbf{a}$ such that $\kappa^{(t)}_{\mathbf{u},\mathbf{a}} < \eta$, where $\eta$ is a threshold empirically set to 0.08. We then examine the number of surviving candidate anchors for $\mathbf{u}$: if no anchors have survived, we add $\mathbf{u}$ to $U^{(t)}_-$; otherwise, if at least one anchor has survived, we add $\mathbf{u}$ to $U^{(t)}_+$. If exactly one anchor has survived, this is chosen as the best anchor; if multiple anchors have survived, we select the one with the largest 3D volume on the basis that whilst smaller anchors may yield slightly higher confidence, owing to the lesser amount

| Classes | Eval. | Method | Accuracy@IoU | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | *0.1* | *0.2* | *0.3* | *0.4* | *0.5* | *0.6* | *0.7* |
| Vehicle | 2D | TCR⋆ | 0.76 | 0.73 | 0.71 | 0.67 | 0.59 | 0.49 | 0.29 |
| | | Ours | **0.91** | **0.88** | **0.84** | **0.81** | **0.75** | **0.67** | **0.47** |
| | BEV | TCR⋆ | 0.69 | 0.68 | 0.64 | 0.57 | 0.50 | 0.35 | 0.12 |
| | | Ours | **0.86** | **0.86** | **0.84** | **0.82** | **0.77** | **0.69** | **0.39** |
| Pedestrian | 2D | TCR⋆ | 0.28 | 0.08 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | | Ours | **0.48** | **0.25** | **0.09** | **0.01** | 0.00 | 0.00 | 0.00 |
| | BEV | TCR⋆ | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | | Ours | **0.28** | **0.20** | **0.13** | **0.08** | **0.04** | **0.01** | **0.01** |
| Cyclist | 2D | TCR⋆ | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | | Ours | **0.90** | **0.90** | **0.90** | **0.90** | **0.30** | 0.00 | 0.00 |
| | BEV | TCR⋆ | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | | Ours | **0.90** | **0.90** | **0.60** | **0.10** | 0.00 | 0.00 | 0.00 |

Table 4.2: Comparing the per-class accuracies of our method to those of TCR⋆ (the improved version of TCR [53] published on GitHub) on the KITTI tracking benchmark [49].

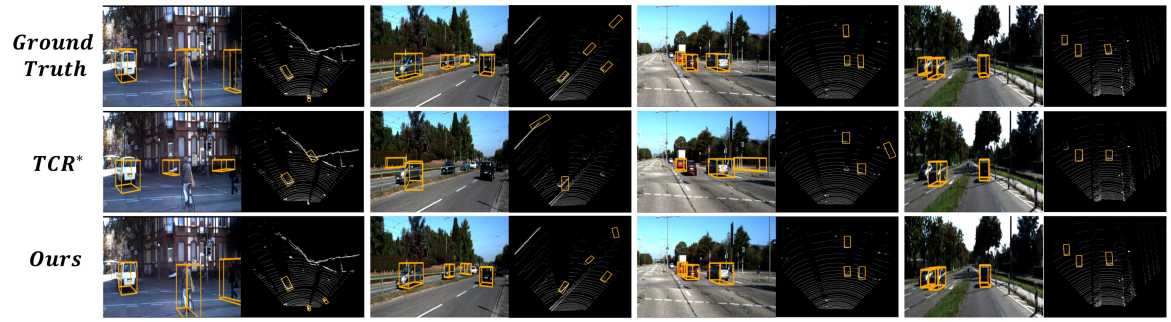



Figure 4.5: Qualitatively comparing our method to the state-of-the-art TCR⋆ method (the improved version of TCR [53]) for self-supervised mobile object detection. See main text.

of background they contain, the largest anchor has met the threshold and is the one most likely to be correct in practice.

We can now define the pseudo-ground truth bounding box $\mathbf{b}_{\mathbf{u}}^{(t)}$ for each pixel $\mathbf{u} \in U_{+}^{(t)}$ with best anchor $\mathbf{a}^{\star}$ as the box that results from transforming $Q_{\mathbf{u},\mathbf{a}^{\star}}^{(t \rightsquigarrow 0)}$ back into LiDAR space using the (fixed, pre-calibrated) transformation $S_C^L \in \mathbb{SE}(3)$. (Note that the pseudo-ground truth boxes must be in the same space as the predicted ones to compute the loss.)

### 4.3.4 Loss Formulation

Our loss $L^{(t)}$ for frame $t$ can be calculated as a sum of terms, each of which compares an individual predicted box $\tilde{\mathbf{b}}_{\mathbf{u}}^{(t)} = (\tilde{\boldsymbol{\delta}}_{\mathbf{u}}^{(t)}, \tilde{\mathbf{d}}_{\mathbf{u}}^{(t)}, \tilde{r}_{\mathbf{u}}^{(t)}, \tilde{\kappa}_{\mathbf{u}}^{(t)})$ with its corresponding fitted box $\mathbf{b}_{\mathbf{u}}^{(t)} = (\mathbf{c}_{\mathbf{u}}^{(t)}, \mathbf{d}_{\mathbf{u}}^{(t)}, r_{\mathbf{u}}^{(t)}, \kappa_{\mathbf{u}}^{(t)})$. As in §4.3.1, recall that the centre $\tilde{\mathbf{c}}_{\mathbf{u}}^{(t)}$ of $\tilde{\mathbf{b}}_{\mathbf{u}}^{(t)}$ can be computed from $\tilde{\boldsymbol{\delta}}_{\mathbf{u}}^{(t)}$, the predicted offset from the centre of the pillar corresponding to $\mathbf{u}$. We distinguish between pixels in $U_{+}^{(t)}$ and $U_{-}^{(t)}$, supervising only the confidence scores of the latter. Specifically, we compute

$$\begin{aligned} L^{(t)} = \sum_{\mathbf{u} \in U_{+}^{(t)}} \big( & BL_1(\tilde{\mathbf{c}}_{\mathbf{u}}^{(t)}, \mathbf{c}_{\mathbf{u}}^{(t)}) + BL_1(\tilde{\mathbf{d}}_{\mathbf{u}}^{(t)}, \mathbf{d}_{\mathbf{u}}^{(t)}) + BL_1(\tilde{r}_{\mathbf{u}}^{(t)}, r_{\mathbf{u}}^{(t)}) + \\ & \left\| \tilde{\kappa}_{\mathbf{u}}^{(t)} - \kappa_{\mathbf{u}}^{(t)} \right\|_2 \big) + \sum_{\mathbf{u} \in U_{-}^{(t)}} \left\| \tilde{\kappa}_{\mathbf{u}}^{(t)} - \kappa_{\mathbf{u}}^{(t)} \right\|_2, \end{aligned} \tag{4.6}$$

in which $BL_1$ denotes the balanced $L_1$ loss from [120].

## 4.4 Experiments

Like TCR [53], we experiment on the KITTI tracking benchmark [49], and use sequences 0000-0009 for training and 0010 and 0011 for testing. We train our model for 300 epochs, using Adam optimisation with batch size 8, an initial learning rate of $10^{-3}$ and cosine learning rate annealing on Nvidia Titan X GPU. We then compare with three recent self-supervised methods that can detect objects in 2D or 3D without requiring ground-truth annotations: (i) Slot Attention [106] uses reconstruction error to learn a multi-block representation of the scene that can be used to detect objects within it; (ii) SCALOR [70] is a generative tracking model that learns to capture object representations from video and enables the detection of a large number of objects in a complex scene; (iii) TCR [53] first tries to find candidate regions using RANSAC, then, throughout multiple training stages, uses 2D-3D correspondence scores for possible object regions as a supervision signal. To clarify the current performance drop of state-of-the-art self-supervised methods in comparison to annotated methods, we also compare to a recent semi-supervised method [163] and a recent fully supervised one [147].

### 4.4.1 Object Discovery

Our first set of experiments compares our method's ability to discover mobile objects to that of the other methods we consider. We evaluate both the 3D bounding boxes predicted by each method (which we compare to the 3D KITTI ground truth boxes) and their projections into the 2D camera images (which we compare to the 2D KITTI ground truth). We refer to the first of these evaluation types as BEV and

the second as 2D, in line with [53]. To project each predicted 3D bounding box into 2D, we project its 8 vertices individually into the image plane and then use the minimum and maximum components of the projected vertices to fit the tightest possible 2D axis-aligned bounding box around them.

As shown in Table 4.1, our method significantly outperforms the state-of-the-art self-supervised method TCR⋆ (the improved version of TCR [53] published on GitHub) in both the 2D and BEV settings. It is also interesting to compare our performance to the semi-supervised 3DIoUMatch [163], and the fully supervised PV-RCNN++ [147]. For PV-RCNN++, the model is trained with all of the 16,648 annotated boxes available in our training set (sequences 0000-0009 of the KITTI tracking benchmark [49]). For 3DIoUMatch, we instead randomly select 1,664 (i.e. $\approx 10\%$) of these annotated boxes for training.

For IoUs up to 0.5, our BEV mAP is within 20% of that of 3DIoUMatch [163] and 30% of PV-RCNN++ [147], which is encouraging given that our method is unable to leverage any ground-truth annotations. Moreover, whilst both do greatly outperform us at an IoU of 0.7, our BEV mAP, in this case, is still over $3\times$ higher than that of the previous state-of-the-art TCR⋆ [53].

### 4.4.2 Per-class Accuracies

Our second set of experiments compares our method's per-class accuracies (at different IoUs) to those of the state-of-the-art self-supervised TCR⋆ approach (see Table 4.2). Both methods are self-supervised and do not predict class labels for the objects they detect; however, we can still calculate per-class accuracies by associating each predicted box with the (labelled) ground-truth box with which it has the greatest IoU. Both methods achieve much higher accuracies for vehicles than for pedestrians/cyclists at all IoU thresholds. However, the accuracies achieved by our method for pedestrians/cyclists are also much higher than those of TCR⋆, which almost completely fails for these classes. We hypothesise that this may be partly a result of the non-rigid motions exhibited by pedestrians/cyclists, which can be hard to group together without additional cues. Moreover, the pedestrians/cyclists are smaller than the vehicles and appear much less frequently in the KITTI training set, making these classes harder for networks to learn. Nevertheless, as our method takes advantage of additional cues in the form of expected object sizes and box-level consistency scores, it is able to achieve at least moderate accuracies even on these more difficult classes (see also Figure 4.5).

| Object Representation | Eval. | mAP@IoU | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 |
| Box | 2D | 0.79 | 0.77 | 0.74 | 0.72 | 0.66 | 0.59 | 0.41 |
| | BEV | 0.79 | 0.79 | 0.76 | 0.74 | 0.69 | 0.62 | 0.33 |
| $\theta$=3, $\varphi$=3 | 2D | 0.79 | 0.77 | 0.74 | 0.74 | 0.69 | 0.60 | 0.41 |
| | BEV | 0.79 | 0.77 | 0.77 | 0.74 | 0.71 | 0.63 | 0.34 |
| $\theta$=4, $\varphi$=4 | 2D | 0.81 | 0.79 | 0.75 | 0.75 | 0.71 | 0.62 | 0.43 |
| | BEV | 0.80 | 0.80 | 0.77 | 0.75 | 0.72 | 0.64 | 0.36 |
| $\theta$=5, $\varphi$=5 | 2D | 0.78 | 0.77 | 0.73 | 0.72 | 0.69 | 0.60 | 0.42 |
| | BEV | 0.78 | 0.78 | 0.74 | 0.73 | 0.70 | 0.63 | 0.34 |

Table 4.3: Quantitatively comparing the performance in mAP when replacing box representation with Radial Instance Detection (RID). Here, box refers to the performance when using the existing bounding Box as the object representation as in Chapter 3. $\theta$ and $\varphi$ stand for the number of horizontal and vertical preset angles used.

### 4.4.3 Impact of Spherical Representation

In this experiment, we investigate how spherical representation could contribute to the system. To see the effect, we replace the Cartesian-based bounding box with Radial Instance Detection (RID) introduced in Chapter 3, while all other modules of the system remain the same. We provide two experimental results. Table 4.3 shows the performance in mAP. Here, in order to get the box from RID, we first extract points inside each radial detection and fit the box as described in Chapter 3.

Interestingly, when using ($\theta$=3, $\varphi$=3), the performance in 2D does not improve compared to the box representation. This is partly because of the box fitting process that estimates excessive size. For instance, if there are not many preset angles, each spherical sector needs a long ray that can include all the points inside the sector. This leads to big box estimation when RID is converted into a box, which leads to 0.01 mAP lower in 0.1 IoU than using box representation for BEV detection. However, as the number of preset angles increases, this problem is alleviated, as can be seen in the cases of ($\theta$=4,$\varphi$=4) and ($\theta$=5,$\varphi$=5). In particular, RID shows superior performance in relatively high precision metrics ( IoU >= 0.5), suggesting that using RID enables more precise detection by learning each part of an object rather than a whole object as the box. However, as expected, the learning complexity grows as the number of parameters increases. This leads to performance degrade in ($\theta$=5,$\varphi$=5) compared to ($\theta$=4,$\varphi$=4). The adoption of more capable architecture, such as Transformer, would improve the performance of a larger number of parameters.

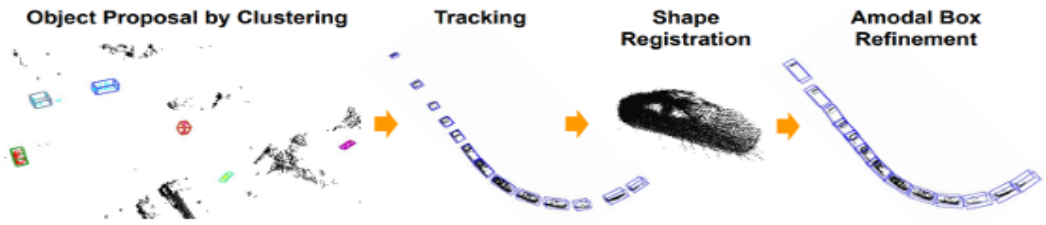



Figure 4.6: Flow diagram of DBSCAN++.

### 4.4.4 Failure Cases

Table 4.2 shows significant performance drops for non-rigid objects, such as pedestrians and cyclists. This is because the foreground points of these objects are not considered moving objects due to the unstable motions of foreground points. For example, when a pedestrian is walking, their hands and legs naturally move back and forth, which gives a less clear signal than the movement of rigid objects, leading to false negative pseudo labels of these classes. Another limitation of Sample Crop Track is the use of expected object size as a prior, which has been reported to be different depending on the regional preference [165].

## 4.5 Recent Works after the Publication

After its publication with the title "Sample, Crop, Track: Self-Supervised Mobile 3D Object Detection for Urban Driving LiDAR" in International Conference on Robotics and Automation (ICRA) 2023, there have been several works in the similar field: DBSCAN++ [114], LISO [6] and SeMoLi [142].

### 4.5.1 Motion Inspired Unsupervised Perception and Prediction in Autonomous Driving (DBSCAN++)

DBSCAN++ formulates the problem into open-set moving object detection from self-learned scene flow. The scene flow is then used for DBSCAN-based clustering to acquire pseudo labels. Figure 4.6 shows the steps to acquire pseudo labels. Similar to Sample Crop Track, DBSCAN++ aims to distinguish static and dynamic points. However, unlike Sample Crop Track, DBSCAN++ utilizes self-supervised scene flow estimation, which could perform better points tracking on the given domain.

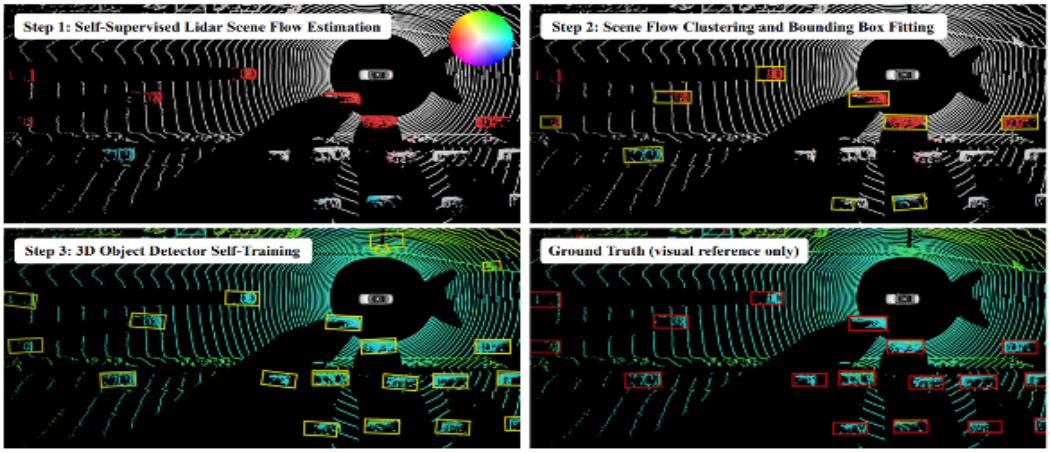



Figure 4.7: Flow diagram of LISO.

### 4.5.2 LISO: Lidar-only Self-Supervised 3D Object Detection

Similarly, LISO employes self-supervised LiDAR scene-flow estimation and ego-motion estimation to get pseudo labels. Figure 4.7 shows the steps to train the object detector without labels. Compared to Sample Crop Track, LISO employs self-supervised scene flow estimation for LIDAR. In addition to DBSCAN++, it includes accurate egomotion estimation steps to acquire dynamic points as the candidate of pseudo labels. In order to get reliable signals for only rigid moving objects, LISO introduces a track optimization technique based on a quadratic regularizer term to find reliable pseudo labels.

### 4.5.3 SeMoLi: What Moves Together Belongs Together

SeMoLi leverages scene flow and message-passing networks for class-agnostic correlation grouping to get pseudo labels, as shown in Figure 4.8. Specifically, SeMoLi consider point cloud as a graph where each point is a node with motion feature. The aim of correlation clustering algorithm is then to cut edges to obtain a set of connected components in successive frames of point cloud for finding moving objects. Although the approach is similar to DBSCAN++, SeMoLi achieves the state of the art result by introducing the correlation-based clustering with the graph. The proposed clustering approach combined with self-supervised scene flow estimation leads to less false positive pseudo labels compared to Sample Crop Track.

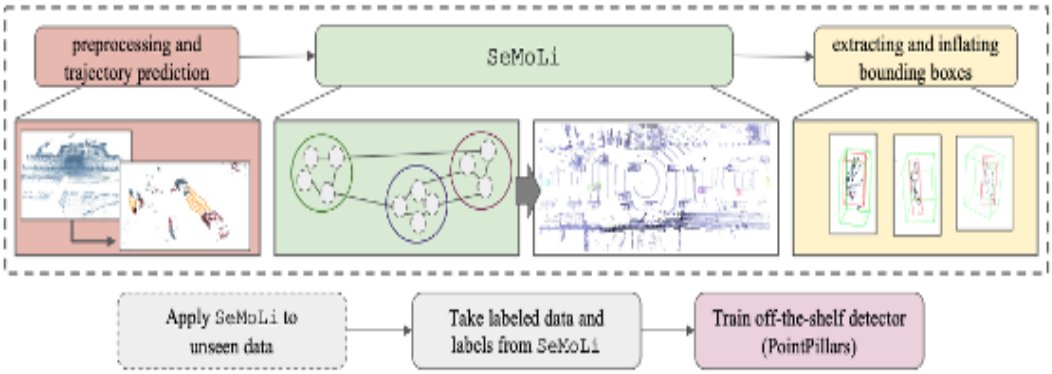



Figure 4.8: Flow diagram of SEMOLI.

## 4.6 Conclusion and Discussion

In this chapter, we propose SCT, a method for self-supervised mobile object detection in urban driving scenes. SCT uses additional cues, such as the expected sizes of target objects, to estimate pseudo-ground truth boxes and outputs a possible box for each pixel to improve object discovery. Experimentally, our object discovery results show improvements (for all IoUs) of more than 36% in BEV mAP and more than 25% in 2D mAP over the state-of-the-art self-supervised TCR⋆ method (the improved version of TCR [53] published on GitHub) on the KITTI tracking benchmark [49]. Indeed, for IoUs up to 0.5, we achieve BEV mAP scores that are within 30% of the fully supervised PV-RCNN++ [147] method, without requiring any manual annotation. Furthermore, unlike TCR⋆, our method is able to cope at least to some extent with trickier classes such as pedestrians and cyclists, although further work is needed to improve the accuracies for these classes at high IoUs.

Collecting pseudo-labels in this chapter addresses the issue of high-cost annotation when the operation environment continually changes, which reflects the real world. In the real world, the object localisation model is likely to encounter diverse environments while accumulating pseudo labels of the encountered environments. In order to achieve the best performance of the object localisation model, the accumulated pseudo-labels need a careful strategy for their usage to train the model. For example, in dry regions, the number of pseudo labels collected on rainy days would be notably smaller than the pseudo labels collected during sunny days. Directly using all of the pseudo labels for training leads to an imbalance problem, where the model would not perform optimally on rainy days. The next chapter will focus on how to best distribute the newly collected pseudo labels along with already existing labels to minimise these domain gaps caused by region-wise differences for object detection.

# Chapter 5

# Towards Learning Group-Equivariant Features for Domain Adaptive 3D Detection

This chapter explores strategies to distribute pseudo labels and existing labels for training object detection models. We start by addressing the limitations of the previous works and motivate an improved strategy in the context of domain adaptive 3D detection.

The performance of 3D object detection in large outdoor point clouds deteriorates significantly in an unseen environment due to the inter-domain gap. To address these challenges, most existing methods for domain adaptation harness self-training schemes and attempt to bridge the gap by focusing on a single factor that causes the inter-domain gap, such as objects' sizes, shapes, and foreground density variation. However, the resulting adaptations suggest that there is still a substantial inter-domain gap left to be minimized. We argue that this is due to two limitations: 1) Biased pseudo-label collection from self-training. 2) Multiple factors jointly contribute to how the object is perceived in the unseen target domain. In this section of the thesis, we propose a grouping-exploration strategy framework to address those two issues. Specifically, our grouping divides the available label sets into multiple clusters and ensures all of them have equal learning attention with the group-equivariant spatial feature, avoiding dominant types of objects causing imbalance problems. Moreover, grouping learns to divide objects by considering inherent factors in a data-driven manner, without considering each factor separately as existing works. On top of the group-equivariant spatial feature that selectively detects objects similar to the input group, we additionally introduce an explorative group update strategy that reduces the false negative detection in the target domain, further reducing the inter-domain gap. During inference, only the learned group features are necessary for making the group-equivariant spatial

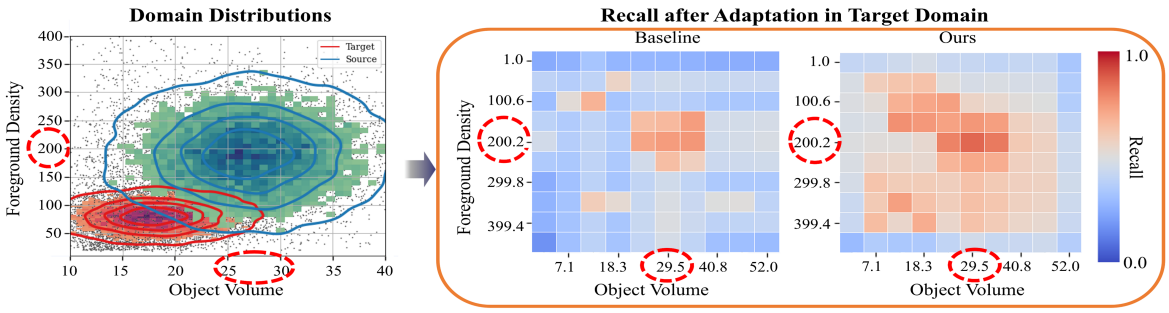



Figure 5.1: Several factors causing the inter-domain gap, such as the point density and object volume in NuScenes (Target) and Waymo (Source) datasets, are illustrated with the fitted multivariate Gaussian distribution (left). The baseline [183] adaptation primarily detects objects having features near the mean of the distributions indicated by red circles. On the other hand, the proposed adaptation first groups objects and explores the target domain to reduce the false negative. The heatmaps show average recall (right). Objects with extreme sparsity are excluded for clear visualization.

feature, placing our method as a simple add-on that can be applicable to most existing detectors. We show how each module contributes to substantially bridging the inter-domain gaps compared to existing works across large urban outdoor datasets such as NuScenes, Waymo, and KITTI.

## 5.1 Introduction

Learning 3D object detection in large outdoor point cloud scenes is of increasing importance due to its wide range of applications, such as autonomous driving [121] and Augmented Reality (AR) [89]. However, annotating such 3D data is expensive and labour-intensive. This limits the applicability and generalization of off-the-shelf models to diverse real-world applications. To mitigate the dependency on large-scale datasets with labels, studies for Domain Adaptation (DA) have gained considerable attention from the community [165, 140, 178, 188, 184, 183, 123, 94, 171, 61, 24]. In the context of 3D detection, DA is motivated by utilizing knowledge learned from a source domain to adapt to a target domain using a pseudo-label set. Typically, in the widely used self-training scheme [184, 183], a detection model is pre-trained on the source domain and constructs an initial pseudo-label set for retraining in the target domain. The pseudo-label set progressively expands as more pseudo-labels are collected after each retraining. Here, the pseudo-label set consists of detections with high confidence scores. Based on the principle of self-training, recent existing works focus on specific factors that cause the inter-domain gap. These can be broadly categorized into domain variations in object sizes [165], density [61], or geometric structure [94, 123]. Despite good progress in bridging the

inter-domain gap, two challenges remain unsolved: 1) Biased pseudo-labels due to the conservative collection strategy of the self-training scheme 2) Strict separation of multiple factors that jointly contribute to the creation of inter-domain gap. Typically, in one domain, a group of objects sharing common features outnumber other objects having different features, as addressed in existing works [165, 61, 94, 123]. While this may not cause a significant performance deterioration in the source domain, where the environment is similar, it could cause a bias under the self-training scheme. For example, detection with high confidence scores typically comes from objects belonging to dominant groups, as they are the ones that the detector learns the most of in the source domain. As shown in Figure 5.1 (a), the recall of objects in the target domain is significantly higher when the objects contain similar features as the dominant objects in the source domain, ignoring other objects. As a partial solution to this problem, existing works focus on a single factor only to address the inter-domain gap, e.g. size or point cloud density. However, as Figure 5.1 shows, a single factor cannot explain all the domain variation as an object's appearance is a result of multiple factors jointly influencing the foreground points.

In this section of the thesis, we address the aforementioned two issues of the current domain adaptive 3D detection. Our core intuition comes from the fact that the factors causing inter-domain gaps often already exist in the source domain to an extent. For example, the density of points often varies due to the distance and viewing angle of the sensor, even inside a domain. Object sizes and shapes would also vary even in one single domain. Based on this observation, we aim to reduce the bias of the detection model to dominant objects by finding groups and evenly distributing the detector's attention during learning to all groups that would be otherwise largely neglected due to less dominant features. Nevertheless, finding optimal groups that represent the intra-domain gap is not straightforward because multiple factors jointly contribute to variations in objects' appearances. To address this issue, we introduce a data-driven grouping method that finds object groups with different characteristics. The groups are then progressively updated for adaptation, redistributing the available labels according to each group to learn the different characteristics found for each group in the target domain. Our contributions can be summarized as follows:

1. We introduce a new alternative approach for domain adaptation, Group Explorer Domain Adaptation, **GroupExp-DA**, which reflects on the available labels in order to understand the target domain, bridging the inter-domain gap.

2. To ensure each object group receives equal attention for learning, we introduce the Group-Equivariant spatial feature, which is learned for selectively detecting

objects similar to the input group, preventing dominant types of objects from causing imbalance problems.

3. To make the best use of the Group-Equivariant feature's selective detection ability depending on the input group, we propose an exploration strategy that encourages the groups to reflect the target domain by redistributing available labels, leading to fewer false negatives for the adaptation.

4. Extensive adaptation experiments on KITTI, Waymo, and NuScenes datasets show the effectiveness of our approach for bridging inter-domain gaps.

## 5.2 Related Work

**Point Cloud-based 3D Detection for Outdoor Scene.** Existing methods for 3D point cloud-based object detection can be divided into two main categories: point-based and voxel-based. Based on PointNet series [125, 128], point-based methods [127, 149, 149, 126, 185] propose to extract features from unordered and unstructured raw point-cloud directly. Voxel-based methods divide the unstructured point clouds into regular voxel grids and are fed into encoders either based on sparse convolution [180, 204, 85, 150, 145, 22, 74, 192] or Transformer [154, 93, 42, 162, 92]. The encoded spatial features, which are also called Bird's Eyes View (BEV) features, are then fed into Regional Proposal Networks (RPN) for the final detection. There are also a few methods that combine point- and voxel-based methods [146, 147, 168]. Currently, in terms of performance, voxel-based detectors dominate the point-cloud-based 3D detection for outdoor scenes. Therefore, we employ Second-IoU [180] and PointPillars [85] as the base detectors for our experiments, as they are the most widely used detectors that existing works are built on.

**Domain Adaptive Detection.** The target of domain adaptive 3D detection is to mitigate the inter-domain gap between the source and the target by focusing on several factors, such as object size [165, 140], point cloud deterioration [178], and the encoder separation for domains [188]. ST3D series [184, 183] propose a self-training scheme that progressively collects pseudo-label sets in the target domain for retraining. Built on self-training scheme, DA for 3D detection has been extensively studied to acquire higher quality pseudo-labels by addressing the inter-domain gap caused by geometric shape with prototype learning [123, 94], dense-to-sparse density variation using knowledge distillation [171], a general density variation with beam augmentation and knowledge distillation [61], cross-domain examination to measure the consistency of pseudo-labels [24] and focusing on specific architecture [194]. A considerable number of existing works focus on a

single factor, such as density, shape, and size, to bridge the inter-domain gap. Instead, we attempt to see the inter-domain gap as a result of multiple factors combined and bridge the gap by finding inherent groups.

**Summary** Domain adaptive 3D detection methods focus on finding pseudo labels for self-training. The primary importance of these works is finding reliable pseudo labels, which will directly affect the adaptation performance in the target domain. The reliable pseudo labels are typically detections with high confidence scores acquired by the model only trained on the source domain. Although this approach has seen great progress, there exists a risk of biased pseudo label collection, as the detection (pseudo labels) with high confidence scores can be from dominant features of objects in the source environment, which is minor in the target domain, leading to large false negative problems.

## 5.3 Method

### 5.3.1 Framework Overview

Following the self-training-based unsupervised domain adaptation scheme [184, 183] for 3D detection, we are given point clouds $X = X_s \cup X_t$ and labels $Y = Y_s \cup Y_t$ as the initial set. Here, $X_s$ and $Y_s$ are point could and box labels of the known source domain. On the other hand, $X_t$ and $Y_t$ are the point cloud and initial pseudo label set in the unlabeled target domain. $Y_t$ is collected by the detector trained only on the source domain using $X_s$ and $Y_s$. A label in $Y$ consists of seven parameters defining a 3D box with three parameters for centre $(x, y, z)$, three parameters for size $(l, w, h)$, and one parameter for vertical rotation $\theta$. Our goal is to improve the detector's inter-domain adaptation by focusing on the pseudo-label collection. In particular, instead of treating all available objects in $Y$ equivalently, we aim to understand objects in $Y$ better by grouping.

Specifically, as depicted in Figure 5.2, we first extract foreground points from available boxes and learn to encode objects into the object descriptor from the points (Sec.5.3.2). The descriptors are then used to progressively group objects (Sec. 5.3.3). Our Group-Region Correlation (Sec 5.3.4) takes the group features as input and fuses with RPN to selectively detect objects similar to the individual group.

### 5.3.2 Object Descriptor Extraction

Given a pair of point cloud $x \in \mathbb{R}^{n_p \times 3}$ and a label set $y \in \mathbb{R}^{n_b \times 7}$ consisting of $n_p$ points and $n_b$ boxes from $X$ and $Y$, respectively, this module aims to produce a

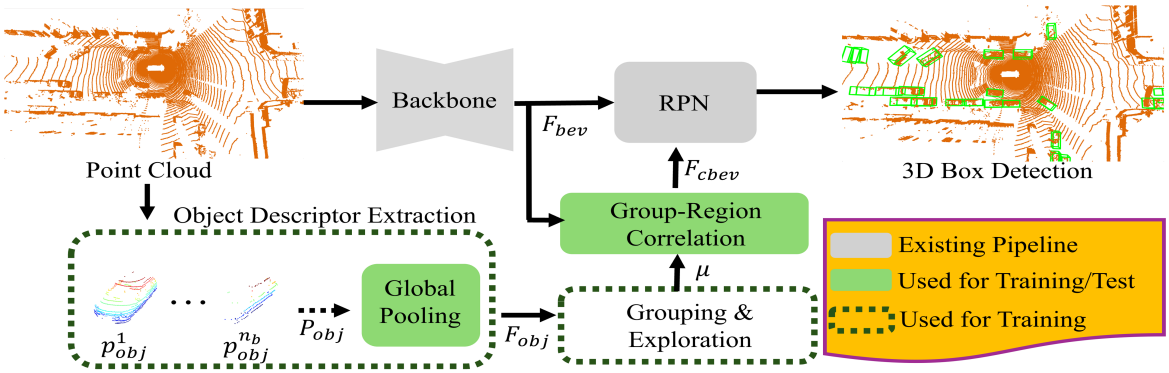



Figure 5.2: The overall pipeline of our proposed method. During training, we extract foreground points from existing 3D box labels and feed them to the Object Descriptor Extraction module to acquire object descriptors. The descriptors are used for grouping & exploration and fed into the Group-Correlation module to generate RPN to detect objects similar to each group.

learnable object descriptor $F_{obj} \in \mathbb{R}^{n_b \times d_{obj}}$ during training. First, foreground points, $P_{obj} = \{p^k_{obj}\}^{n_b}_{k=1}$ are extracted using $y$ from the input point cloud $x$. Our object descriptor extraction module then takes $P_{obj}$ as input and outputs object descriptors $F_{obj}$ using neural networks consisting of MLP and global max pooling, which are adopted from [125, 86]. The motivation behind the architectural choice is two-fold: (1) An arbitrary number of object points, $p^k_{obj}$, can be processed efficiently without involving comparably slow sampling techniques, such as further point sampling. (2) Global max pooling offers permutation invariant features, which helps $F_{obj}$ less prone to overfitting by certain permutations from viewing angles, *etc.* During training, $F_{obj}$ serves as the input for the progressive grouping process and is not used during inference.

### 5.3.3 Grouping & Exploration

Determining similarities between objects is a complex problem as intra-object variations are created by various factors, such as density, shape, size, *etc.* To find groups that, when combined, explain this intra-object variation, we utilize Gaussian Mixture Model (GMM) based grouping for the following advantages: (1) GMM better captures the heterogeneity of data using only a few more parameters, such as covariance and weights, compared to the proximity-based methods. (2) The parameters that define groups can be efficiently updated with weighted linear combinations and are also differentiable for learning the groups in a data-driven manner. In the following sections, we explain the details of how the groups are initialized, determined for each sample, and updated.

**Initialization:** Given all available pairs of scans and labels, X and Y, we first extract $F_{obj}$ from each pair and stack them. All of the stacked $\{F^o_{obj}\}^{n_{total}}_{o=1}$ are then

used for the initialization. Here, $n_{total}$ stands for the total number of pairs in X and Y. Specifically, we initialize $n_g$ groups using K-Means clustering on $\{F^o_{obj}\}^{n_{total}}_{o=1}$, where each cluster forms a group. After this, Maximum Likelihood Estimation is performed for each group to acquire parameters, such as mean $\mu \in \mathbb{R}^{n_g \times d_{obj}}$, covariance $\sigma \in \mathbb{R}^{n_g \times d_{obj} \times d_{obj}}$, and weight $\phi \in \mathbb{R}^{n_g}$ that define a GMM-based group. It is worth noting that this initialization is required only once before the training, which does not slow down the whole training process. In the following, all procedures are based on a single pair of scan and label, $x$ and $y$, which can be extended to batch-wise operation.

**Determining Group for New Sample:** The probability $P^{k\to i}$ of $k$-th sample belonging to $i$-th group is estimated as :

$$P^{k\to i} = \frac{\phi^i \mathcal{N}(F^k_{obj}|\mu^i, \phi^i)}{\sum_{t=1}^{n_g} \phi^t \mathcal{N}(F^k_{obj}|\mu^t, \phi^t)}, \tag{5.1}$$

where $\mathcal{N}(|,)$ is the probability density function of multivariate Gaussian distribution that outputs the likelihood of a sample $F^k_{obj}$ given mean $\mu$ and covariance $\phi$. The group labels $G \in \mathbb{R}^{n_b}$ for all the $n_b$ samples in $F_{obj}$ are then acquired using $P$. That is, the group label $G^k$ for $k$-th sample is specifically determined as:

$$G^k = \arg\max_i P^{k\to i} \tag{5.2}$$

**Explorative Update during Training:** Typically, pseudo-label sets from the target domain are incorporated into the existing label set from the source domain and considered as the same label set for training [184, 183], as shown in Figure 5.3 (a). However, we argue that for the domain adaptation, the pseudo-label set should be given more weight on its usage than the source label sets because they contain inherent features of the target domain, which can be used to understand the target domain in terms of objects' appearances. To address this, we introduce an explorative group update strategy using pseudo-labels. For $i$-th group, its mean $\hat{\mu}^i \in \mathbb{R}^{d_{obj}}$, covariance $\hat{\sigma}^i \in \mathbb{R}^{d_{obj} \times d_{obj}}$, and the group weight $\hat{\phi}^i$ are acquired as follows:

$$\hat{\mu}^i = \frac{1}{n_{b,i}} \sum_{k=1}^{n_{b,i}} F^k_{b,i}, \quad \hat{\sigma}^i = \frac{1}{n_{b,i}} \sum_{k=1}^{n_{b,i}} (F^k_{obj,i} - \hat{\mu}^i)(F^k_{obj,i} - \hat{\mu}^i)^T, \quad \hat{\phi}^i = \frac{n_{b,i}}{n_b}, \tag{5.3}$$

where $F_{obj,i}$ is a subset of $F_{obj}$ that belong to $i$-th group using $G$ in Eqn. 5.2 as:

$$F_{obj,i} = \{F^k_{obj}|G^k = i\}, \tag{5.4}$$

and $n_{b,i}$ is the number of samples in $F_{obj,i}$. Similar to the update rule for prototype learning [123], each group parameters are updated with the linear combination as:

$$\mu = \alpha\mu + (1-\alpha)\hat{\mu}, \quad \sigma = \alpha\sigma + (1-\alpha)\hat{\sigma}, \quad \phi = \alpha\phi + (1-\alpha)\hat{\phi}, \tag{5.5}$$

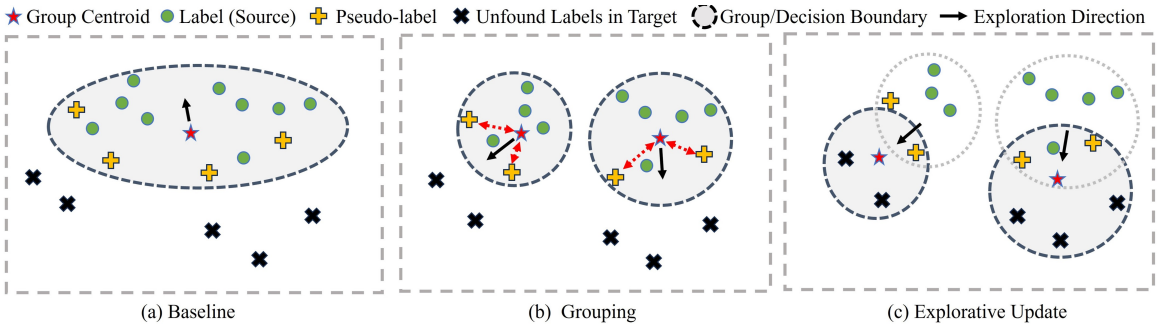



Figure 5.3: Conceptual diagram comparing pseudo-label generation processes of (a) baseline [184, 183] with our (b) grouping followed by (c) the explorative update using pseudo-labels.

where $\alpha$ is a coefficient that affects how conservatively each parameter is updated. After the update, $\mu^i$ is considered as a representative of $i$-th group and utilized as a query input for generating group-equivariant spatial features. Accordingly, during training, the samples that belong to the $i$-th group are used as the foreground boxes. Intuitively, this process distributes source labels according to the groups found in the target domain so that the source labels are used for learning to find similar objects in the target domain, as shown in Figure 5.3 (b) and (c).

For training, to ensure that the groups learn to be distinctive enough, we adopt inter-group repel [94] based on the contrastive loss.

$$L_{rep} = \sum_{i=1}^{n_g-1} \sum_{j=i+1}^{n_g} max(0, cos(\mu^i, \mu^j)), \tag{5.6}$$

where $cos(,)$ is a function that calculates cosine-similarity between two inputs. In addition, to encourage similar features for group cohesion, intra-group attraction loss, $L_{att}$ is used:

$$L_{att} = \sum_{i=1}^{n_g} \sum_{k=1}^{n_b} (1 - cos(F_{obj}^k, \mu^i)) \mathbb{1}[G^k = i], \tag{5.7}$$

where $\mathbb{1}[G^k = i]$ is an indicator function that is 1 if $G^k = i$ and 0 otherwise. During the inference, only the learned $\mu$ is necessary for the following Group-Region Correlation.

### 5.3.4 Group-region Correlation

In a typical voxel-based 3D object detector's pipeline, voxelized points are fed into the backbone, which outputs the spatial feature $F_{bev} \in \mathbb{R}^{H \times W \times d_{bev}}$, as shown in Figure 5.2. $F_{bev}$ is then fed into RPN to make final box predictions. Given input group queries $\mu = \{\mu^i\}_{i=1}^{n_g}, \mu^i \in \mathbb{R}^{d_{obj}}$ and $F_{bev}$, Our Group-Region Correlation aims at producing the spatial features $F_{cbev} \in \mathbb{R}^{n_g \times H \times W \times d_{bev}}$ that are equivariant to the

group query so that $F_{cbev}$ provide selective features for detecting objects similar to each group. In the following sections, we explain how each group and spatial feature are correlated and then used by RPN to detect the corresponding objects.

**Group Equivariant Spatial Feature:** The aim of the Group-Region Correlation module is to make spatial features $F_{bev}$ selectively attend to objects that are similar to the query group. The following RPN then detects only certain objects that are similar to the query group. To achieve this, we utilize cross-attention with $\mu$ as query and $F_{bev}$ as key and value to encourage features from $\mu$ and $F_{bev}$ to cross-attend to generate necessary features. Intuitively, $\mu$ is compared to $F_{bev}$ to find the object that is similar to each group. For $i$-th group, the attended group-equivariant spatial features $F^i_{cbev} \in \mathbb{R}^{H\times W\times d_{bev}}$ are acquired as:

$$F^i_{cbev} = cAttn(\mu^i, F_{bev}, F_{bev}), \tag{5.8}$$

where $cAttn(.)$ refers to the cross attention [159] that takes query, key, and value as input and outputs the cross-attended feature. Here, $\mu^i$ is the $i$-th query in $\mu$. The group-equivariant spatial features $F_{cbev}$ can then be fed into any existing RPN strcutures [180, 37, 85] to detect foreground objects for each group. The ground-truth boxes in $y$ that belong to the $i$-th group are utilized as the foreground boxes for $F^i_{cbev}$ to train the following RPN.

**Regional Proposal Network (RPN):** Following the general architecture of RPN [180, 85], our objectness and box regression heads take $F_{cbev} \in \mathbb{R}^{n_g\times H\times W\times D}$ as input and predict objectness scores $F_{cls} \in \mathbb{R}^{n_g\times H\times W\times 1}$ and box parameters $F_{box} \in \mathbb{R}^{n_g\times H\times W\times 7}$ to form 3D boxes on the dense spatial grid corresponding to $F_{cbev}$.

For the training of $i$-th group, standard training losses for RPN based detection, $L^i_{det}$, are applied as existing works [180, 85, 150, 145, 22, 74, 192]. Specifically, given the box labels $y^i$, $F^i_{cls}$ and $F^i_{box}$ are used for calculating first-stage box detection training loss, $L_{det1}$, as:

$$L^i_{det1} = L^i_{focal} + L^i_{box}, \tag{5.9}$$

where $L_{focal}$ stands for Focal Loss [97] and $L_{box}$ is box regression loss. Here, the foreground labels and regression targets for $L_{focal}$ and $L_{box}$ are calculated using $y^i$ depending on the individual base detectors' configurations. Similarly, for the second-stage box refinement training, $F_{cbev}$ is used with $F_{cls}$ and $F_{box}$ for RoI Pooling. Then, the pooled features are fed into the classification or box regression head for refinement with architectures depending on the detectors to calculate the second stage loss $L^i_{det2}$. The detection loss for all groups, $L_{det}$, is then acquired by iterating over all groups as:

$$L_{det} = \frac{1}{n_g}\sum_{i=1}^{n_g} L^i_{det1} + L^i_{det2}. \tag{5.10}$$

### 5.3.5 Overall Training

Apart from $L_{rep}$ and $L_{att}$ for grouping, our overall training losses are defined the same as the general RPN learning for detection.

$$L = \lambda_1 L_{rep} + \lambda_2 L_{att} + \lambda_3 L_{det} \tag{5.11}$$

Using $L$, we train our system following the self-training scheme [184, 183].

## 5.4 Experiments

### 5.4.1 Datasets

We evaluate our methods against various baselines across three different datasets, such as KITTI [49], NuScenes [12], and Waymo [153]. KITTI contains 7481 frames of point clouds for training and validation, and all the data is collected with 64-beam Velodyne LiDAR. NuScenes dataset contains 28130 training and 6019 validation point clouds collected with a 32-beam roof LiDAR. Waymo dataset contains 122000 training and 30407 validation frames of point clouds collected with five LiDAR sensors, i.e., one 64-beam LiDAR and four 200-beam LiDAR.

### 5.4.2 Implementation Details

For a fair comparison with existing domain adaptive 3D detection methods, we build our model on two base detectors, Second IoU [180] and PointPillars [85], following [184, 183, 94, 61], that are widely used and applicable to most recent detectors with the implementation based on OpenPCDet [155] and parameters from ST3D [184]. Following [94], we first train each detector for 30 epochs with batch-size 12 as a pretraining step using a single NVIDIA A10 GPU. In the self-training stage, we train 30 more epochs for the tuning to adapt to the target domain. The learning rate is set to $1\times10^{-3}$ using Adam optimizer with Cosine annealing [191] for scheduling the learning rate. The feature dimensions $d_{bev}$ and $d_{obj}$ for $F_{bev}$ and $\mu$ are both set to 512 for $cAttn$. $\alpha$ for updating the group parameters is empirically set to 0.8. $\lambda_1$, $\lambda_2$, and $\lambda_3$ are set to 0.5, 0.5, and 1.0, respectively.

**Details of Self-Training Implementation:** For each cycle of the self-training, we train our system for 2 epochs using $L$ in Eqn. 5.11 and collect the pseudo-label sets to merge into $Y$ using the model's inference. Here, the collection of the pseudo-labels takes place in the training set of the target domain. Utilizing the new $Y$, we progressively train the detector for $n_{se}$ epochs again and iterate the same process. The confidence score of the detection to be considered as the pseudo label is set to 0.5, and $n_{se}$ is set to 2 epochs.

### 5.4.3 Comparing Methods

We compare recent existing 3D domain adaptive detection methods, such as SN [165], 3D-CoCo [188], ST3D [184, 183], GPA-3D [94], REDB [24], and DTS [61] with our proposed method. As our method is based on self-training, we set ST3D [184] as our baseline and show experimental results by comparing with more recent methods. Additionally, we also illustrate the performance of the **oracle** models, which refer to a fully supervised model on the target domain directly as an upper bound. Following the most recent works [94, 61], all methods are compared in three adaptation scenarios focusing on the "car" class: (1) Waymo → NuScenes (2) NuScenes → KITTI (3) Waymo → KITTI.

### 5.4.4 Evaluation Metric

Following [184, 183, 94, 61], we adopt Average Precision (AP) as our primary evaluation metric and evaluate our model on Bird-Eyes-View (BEV) IoU, $AP_{\text{BEV}}$, and 3D Box IoU, $AP_{\text{3D}}$, with 40 varying recall points and 0.7 as the IoU threshold.

**Quantitative Results** •**Waymo → KITTI** Table 5.1 (first task) shows the quantitative results of 3D detection in $AP_{\text{BEV}}$ and $AP_{\text{3D}}$. When using Second-IoU as detector [180], our proposed method outperforms the baseline ST3D series for 4.75/11.88 in $AP_{\text{BEV}}/AP_{\text{3D}}$, respectively. Compared with the SOTA method, DTS [61], 1.14/2.21 improvements are made. When using PointPillars as the base detector [85], our approach gains 2.34/3.91 improvements compared to the best-performing method, GPA-3D [94].

•**NuScenes → KITTI** As shown in Table 5.1 (second task), our approach shows 2.28/5.53 performance gains in terms of $AP_{\text{BEV}}/AP_{\text{3D}}$ with Second-IoU [180] as the base detector. Compared with SOTA DTS [61], 1.38/1.33 improvements are acquired. With PointPillars [85] as the base detector, our approach exceeds the baseline and DTS by 21.49/41.74 and 2.39/1.04, respectively.

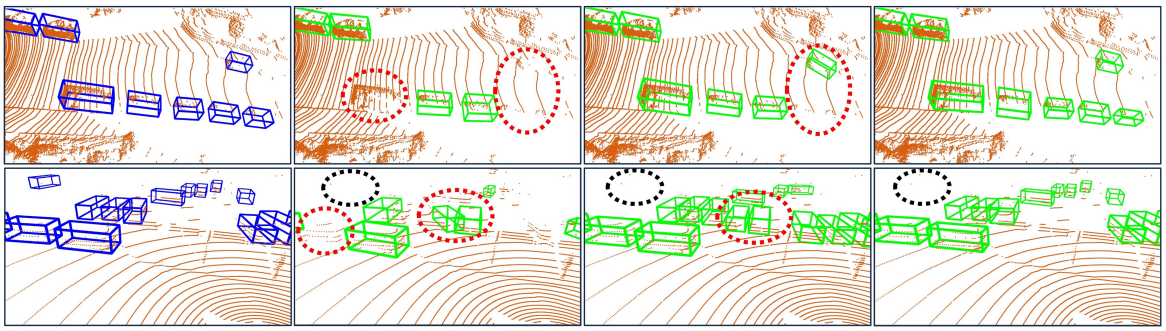
•**Waymo → NuScenes** Table 5.1 (third task) illustrates the adaptation results. Our approach outperforms the baseline and the best-performing method by 8.55/5.72 and 3.27/2.91 in $AP_{\text{BEV}}/AP_{\text{3D}}$, respectively, with Second-IoU as the detector. Similarly, with PointPillars as the detector, 14.89/7.84 and 32.8/1.94 improvements are gained compared with the baseline and SOTA in terms of $AP_{\text{BEV}}/AP_{\text{3D}}$.

**Qualitative Results** Figure 5.4 compares the 3D detection results of the baseline [184], DTS [61] and our methods. Due to the conservative pseudo-label selection policy and absence of methods addressing variations in object size or foreground density, the baseline struggles to detect objects with comparably less common sizes or further away with different densities (red circles in 1st row (b)).

Table 5.1: Quantitative comparisons of the recent domain adaptive 3D detection methods on three adaptation scenarios. The best-performing method is in **bold**, and the second best-performing method is underlined.

| Task | Methods | SECOND-IOU | | PointPillars | |
|---|---|---|---|---|---|
| | | $AP_{BEV}/AP_{3D}$ | Closed Gap | $AP_{BEV}/AP_{3D}$ | Closed Gap |
| Waymo → KITTI | Source Only | 67.64/27.48 | - | 47.8/11.5 | - |
| | SN [165] | 78.96/59.20 | 72.33/69.00 | 27.4/6.4 | -55.14/-8.49 |
| | 3D-CoCo [188] | - | - | 76.1/42.9 | 76.49/52.25 |
| | ST3D [184] | 82.19/61.83 | 92.97/74.72 | 58.1/23.2 | 27.84/19.47 |
| | ST3D++ [183] | 80.78/65.64 | 83.96/83.01 | - | - |
| | GPA-3D [94] | 83.79/70.88 | 103.19/94.41 | 77.29/50.84 | 79.70/65.46 |
| | REDB [24] | 80.37/54.12 | - | - | - |
| | DTS [61] | 85.80/71.50 | 115.9/95.7 | 76.1/50.2 | 76.50/64.4 |
| | Ours | **86.94/73.70** | **123.2/100.4** | **78.44/54.11** | **82.81/71.0** |
| | Oracle | 83.3/73.5 | - | 84.8/71.6 | - |
| NuScenes → KITTI | Source Only | 51.8/17.9 | - | 22.8/0.5 | - |
| | SN [165] | 59.7/37.6 | 25.1/35.4 | 39.3/2.0 | 26.6/2.1 |
| | 3D-CoCo [188] | - | - | 77.0/47.2 | 87.4/65.7 |
| | ST3D [184] | 75.9/54.1 | 76.6/59.5 | 60.4/11.1 | 60.6/14.9 |
| | ST3D++ [183] | 80.5/62.4 | 91.1/80.0 | - | - |
| | REDB [24] | 74.23/51.31 | - | - | - |
| | DTS [61] | 81.4/66.6 | 94.0/87.6 | 79.5/51.8 | 91.5/72.2 |
| | Ours | **82.78/67.93** | **98.3/90.0** | **81.89/52.84** | **95.3/73.6** |
| | Oracle | 83.3/73.5 | - | 84.8/71.6 | - |
| Waymo → NuScenes | Source Only | 32.91/17.24 | - | 27.8/12.1 | - |
| | SN [165] | 33.23/18.57 | 1.69/7.54 | 28.1/12.98 | 2.41/4.58 |
| | 3D-CoCo [188] | - | - | 33.1/20.7 | 25.00/44.79 |
| | ST3D [184] | 35.92/20.19 | 15.87/16.73 | 30.6/15.6 | 13.21/18.23 |
| | ST3D++ [183] | 35.73/20.90 | 14.87/20.76 | - | - |
| | GPA-3D [94] | 37.25/22.54 | 22.88/30.06 | 35.47/21.01 | 36.18/46.41 |
| | REDB [24] | 30.12/18.56 | - | - | - |
| | DTS [61] | 41.2/23.0 | 43.7/32.80 | 42.2/21.5 | 67.9/49.0 |
| | Ours | **43.84/24.42** | **57.56/40.66** | **44.31/22.15** | **77.88/52.34** |
| | Oracle | 51.9/34.9 | - | 49.0/31.3 | - |

Moreover, some dominant object shapes make the detector overfit, leading to false positive detection of road structure (red circles in 2nd row (b)). DTS [61] improves the inter-domain density variance problem presented in the baseline. However, it still encounters the overfitting problem to certain geometric shapes in the source domain, leading to the same false positive detection of the road structure as the baseline (red circles in 2nd row (c)). Moreover, despite the training for foreground density invariant, the objects appearing sparse due to the distance remain false negative (red circles in 1st row (c)) in DTS. The observation suggests that another factor, in addition to the foreground density, causes the inter-domain gap. On the other hand, our proposed method improves both false positive and negative detections, demonstrating a more robust adaptation ability than DTS, which is the best-performing method. Figure 5.7 shows more visualizations of Waymo → KITTI adaptation.



(a) Ground-Truth (b) Baseline (c) DTS (d) Ours

Figure 5.4: Qualitative comparison of Baseline ST3D [184], DTS [61] and ours on NuScenes to KITTI adaptation scenario (top) and Waymo to NuScenes(bottom) adaptation scenarios.

Table 5.2: Impact of each component in $AP_{\text{BEV}}$ and $AP_{\text{3D}}$ on Waymo to NuScenes adaptation.

| | Group | $L_{att}$ | $L_{repel}$ | Exp. Up. | $AP_{\text{BEV}}$ | $AP_{\text{3D}}$ |
|---|---|---|---|---|---|---|
| (a) | | | | | 35.92 | 20.19 |
| (b) | ✓ | | | | 39.48 | 22.66 |
| (c) | ✓ | ✓ | | | 40.82 | 23.14 |
| (d) | ✓ | | ✓ | | 41.14 | 23.61 |
| (e) | ✓ | ✓ | ✓ | | 41.64 | 24.55 |
| (f) | ✓ | ✓ | ✓ | ✓ | **44.47** | **25.91** |

Table 5.3: Impact of grouping methods and the group parameter updating coefficient $\alpha$ on NuScenes to KITTI adaptation.

### 5.4.5 Ablation

**Impact of Each Component** Table 5.2 shows the impact of each component in Waymo→NuScenes adaptation using Second IoU [180] in terms of $AP_{\text{BEV}}/AP_{\text{3D}}$. For this experiment, we progressively add each core component to the baseline (a) to see how they affect the performance. As grouping ensures every group has similar attention during training and prevents only a few dominant object types from having high confidence scores, it significantly improves the performance 9.91%/12.23% from the baseline. From this result, $L_{att}$ (c) and $L_{repel}$ (d) improve 3.39%/2.11% and 4.20%/4.19% respectively, suggesting that making the groups distinctive has more impact than making the intra-group cohesive. This is expected because, during the group-equivariant RPN training, samples inside each group are already learned to be cohesive against the background, and samples from other groups are indirectly supervised to be close to each other. Nevertheless, when combined in (e), $L_{att}$ and $L_{repel}$ improves 15.92%/21.59% from the baseline, demonstrating the synergy of the two losses. On top of this, the explorative update consid-

| | Proximity-based | | GMM-based | |
|---|---|---|---|---|
| | $AP_{\text{BEV}}$ | $AP_{\text{3D}}$ | $AP_{\text{BEV}}$ | $AP_{\text{3D}}$ |
| $\alpha$=0.4 | 79.82 | 64.77 | 81.37 | 66.95 |
| $\alpha$=0.6 | 81.02 | 65.63 | 82.29 | 67.52 |
| $\alpha$=0.8 | 81.61 | 66.20 | **82.78** | **67.93** |
| $\alpha$=0.99 | 81.95 | 66.83 | 81.80 | 66.86 |

Table 5.4: Impact of grouping methods and $\alpha$ on NuScenes to KITTI adaptation.

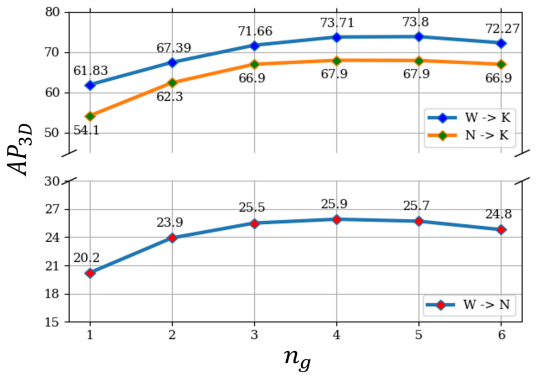



Figure 5.5: Impact of $n_{\text{g}}$ on three adaptation scenarios in $AP_{\text{3D}}$. Here $W \rightarrow K$, $N \rightarrow K$, and $W \rightarrow N$ refer to Waymo→KITTI, NuScenes→KITTI, and Waymo→NuScenes adaptations.

erably boosts the performance by 23.80%/28.33%, proving that using samples found in target (pseudo-labels) for the update further reduces the inter-domain gap, as can also be seen in Figure 5.6 (a group of FN in the middle disappears in the right figure).

**Impact of the number of groups** $n_g$ is illustrated in Figure 5.5 in $AP_{\text{3D}}$ for all adaptation scenarios presented in Section 5.4.4. For this experiment, we include all components with the same hyper-parameters and only change $n_g$ to see the impact. As can be seen, increasing $n_g$ improves the performance, reaching the top around when $n_g$ is set to 4. In principle, having many groups would make the group-equivariant RPN discover more object types while ensuring the same attention for each group during learning. However, in practice, the performances start decreasing when $n_g$ is set around 6, suggesting that having too many groups increases the risks of overfit problems due to a small number of objects in each group or underfit problems due to high complexity for learning $F_{cbev}$.

**Effect of GMM based Grouping and the group parameter updating coefficient** $\alpha$ are illustrated in Table 5.4 to compare: (1) Proximity-based grouping and (2) GMM-based grouping with varying $\alpha$. For the proximity-based grouping, we discard $\sigma$ and $\phi$ from GMM, and determine the group of each sample as the closest $\mu$ from the samples using $L2$ distance while all other configurations stay the same.

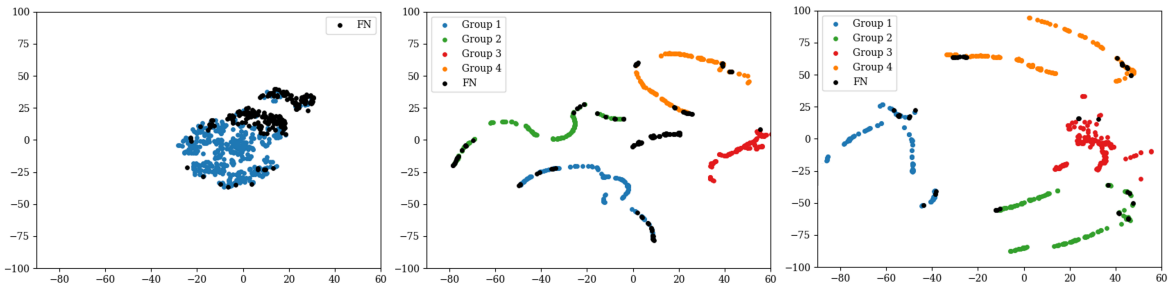



Figure 5.6: Comparison of DTS [61] (left), ours without explorative update (middle), and ours with explorative update (right) in Waymo → NuScenes with t-SNE visualization. Here, the foreground features are extracted using ground-truth boxes using $F_{\text{bev}}$ for DTS and $F_{\text{cbev}}$ for ours.

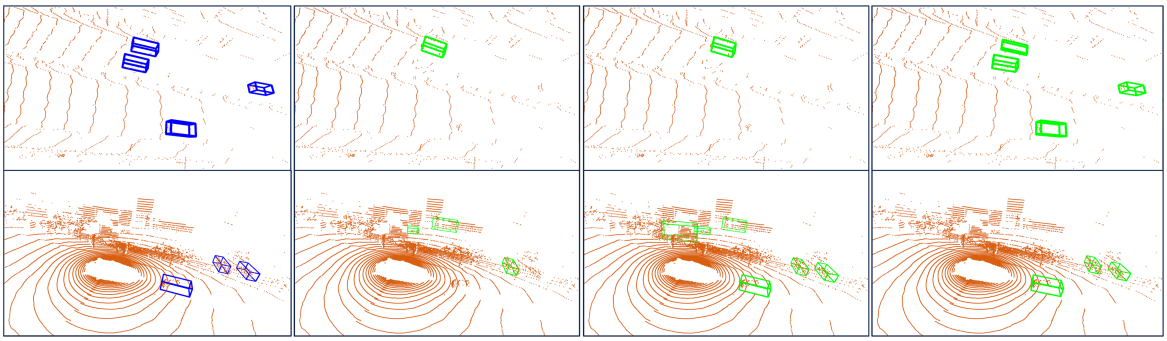



Figure 5.7: Qualitative comparison of Baseline ST3D [184], DTS [61] and ours on Waymo to KITTI adaptation scenario (top) and Waymo to NuScenes(bottom) adaptation scenarios.

As can be seen, GMM-based grouping constantly shows better performances in both $AP_{\text{BEV}}$/$AP_{\text{3D}}$ in all $\alpha$. This is due to $\sigma$ and $\phi$ that preserve the characteristics of each group more compared to only $\mu$ based on previously seen samples, improving the stability during learning with similar objects grouped together. Additionally, setting $\alpha$ too large or small degrades the performances, as small $\alpha$ discourages the explorative update, while too large $\alpha$ could result in instability as the input $\mu$ for training RPN constantly changes more. Also, setting $\alpha = 0.6$ results in 0.49/0.66 higher in $AP_{\text{BEV}}$/$AP_{\text{3D}}$ compared to setting $\alpha = 0.99$, suggesting the advantage from exploration surpasses the stability. Interestingly, unlike GMM-based grouping, setting higher $\alpha$ constantly shows increasingly better performance with the proximity-based grouping, proving inherent instability without $\sigma$ and $\phi$. Nevertheless, in nearly all the configurations of $\alpha$, the GMM-based grouping persistently outperforms the best-performing method, DTS [61].

| Task | Methods | Second-IOU | | PointPillars | |
|---|---|---|---|---|---|
| | | $AP_{BEV}$ | $AP_{3D}$ | $AP_{BEV}$ | $AP_{3D}$ |
| Waymo → KITTI | Box | 86.94 | 73.70 | 78.44 | 54.11 |
| | $\theta$=3, $\varphi$=3 | 87.11 | 73.81 | 78.98 | 54.33 |
| | $\theta$=4, $\varphi$=4 | 87.04 | 73.78 | 78.73 | 54.24 |
| | $\theta$=5, $\varphi$=5 | 87.96 | 73.67 | 78.65 | 54.22 |
| NuScenes → KITTI | Box | 82.78 | 67.93 | 81.98 | 52.84 |
| | $\theta$=3, $\varphi$=3 | 82.84 | 68.10 | 82.14 | 52.96 |
| | $\theta$=4, $\varphi$=4 | 83.02 | 68.15 | 82.20 | 52.99 |
| | $\theta$=5, $\varphi$=5 | 82.91 | 68.07 | 82.01 | 52.81 |
| Waymo → NuScenes | Box | 43.84 | 24.42 | 44.31 | 22.15 |
| | $\theta$=3, $\varphi$=3 | 43.87 | 24.43 | 44.30 | 22.14 |
| | $\theta$=4, $\varphi$=4 | 43.81 | 24.39 | 44.30 | 22.14 |
| | $\theta$=5, $\varphi$=5 | 43.75 | 24.31 | 44.25 | 22.07 |

Table 5.5: Quantitative comparisons of Box and Spherical representation, Radial Instance Detection (RID), for the domain adaptation tasks. Here, Box in Methods indicates the performance using the existing bound box as object representation. $\theta$ and $\varphi$ stand for the number of preset angles (spherical sectors) used for RID.

### 5.4.6 Failure Cases

The proposed method struggles to detect objects with extremely sparse foreground points (black circles in 2nd row), as shown in Figure 5.4, because those objects do not contain distinctive features to being well-learned as groups due to extreme sparsity, which remains as one of the challenges to be explored. A potential approach would be to use a generative model, such as Diffusion [58], for artificially densifying the points of the objects by imitating the ranging sensor's scanning pattern.

### 5.4.7 Applying Spherical Representation

In this experiment, we investigate how Radial Instance Detection (RID) based on spherical representation affects the adaptation system. We replace the Cartesian-based bounding box with RID in Chapter 3 while all other modules of the system stay the same. Table 5.5 shows the performance of domain adaptation performance in mAP. Here, similar to Chapter 4, we first extract points inside each detection from RID and fit the box. Notably, RID contributes more to source environments that are less complex. For example, when the source domain is Waymo, where the variation in objects is larger, the improvement in mAP quickly stops at ($\theta$=4,$\varphi$=4), as more parameters lead to higher learning complexity. However, in a comparably less complex source environment, such as NuScenes, the improvement is made with more parameters ($\theta$=4,$\varphi$=4). Despite the fact that the degree

of improvement depends on the source environment, using RID, in particular for ($\theta$=3,$\varphi$=3), leads to performance improvement in almost all adaptation scenarios, suggesting that RID improves the model's understanding of object for localization.

## 5.5 Conclusion and Discussion

In this chapter, we present *GroupExp-DA* that learns object groups, which can be used to bridge the inter-domain gap with (1) less bias by the dominant objects in the available label sets and (2) consideration of multiple factors for creating inter-domain gaps in a data-driven manner. This is achieved by utilising the group equivariant spatial features that connect the group feature and spatial features to be learned together with the existing detection loss function.

This chapter incorporates the improved object localisation system from Chapter 3 and the concept of collecting pseudo labels for unlabeled environments in Chapter 4 to address the continual object localisation in the formulation of the domain adaptation. The focus of this chapter is to explore the strategy to make the best use of newly added pseudo labels along with existing labels to avoid imbalance problems caused by dominant types of objects. Furthermore, to get a step closer to real-world applications with diverse environments, the scale of environments in this chapter is expanded to different regions.

# Chapter 6

# Conclusions and Future Works

This thesis has explored learning-based object localisation, focusing on overcoming the limitations of continual learning for real-world applications.

In contrast to the image-based localisation techniques, object localisation in 3D has inherent challenges due to the data type (point cloud), as mentioned in Chapter 2, which makes it hard for the DNNs model to generalise. Chapter 3 addresses this issue to first improve the performance of the object localisation system by adopting the spherical coordinate system as opposed to a commonly used Cartesian system, which provides two critical benefits for the coarse-to-fine object localisation approach. We propose Spherical Mask, addressing the two limitations of existing object localisation methods: 1) Excessive box size estimation and 2) Error accumulation from the coarse detection to the refinement stage. Spherical Mask demonstrates its performance in several challenging public benchmarks that require precise object localisation, such as point cloud instance segmentation. In addition to the precision, Spherical Mask also shows competitive execution speed and memory usage compared to other relevant works. Despite the progress, the challenges for point cloud-based object localisation remain clear. First, the current state-of-the-art deep learning-based point cloud instance segmentation methods suffer from high complexity when applied to dynamic outdoor scenes due to their query-based inference architectures. The superiority of spherical representation is also presented in the following studies in Chapters 4 and 5. For example, their common assumption is that, given a point cloud, the initial query points have to be densely located somewhere near objects are located. However, in complex and large outdoor point clouds, there are many objects, such as cars and pedestrians, having very sparse points on their surfaces due to the distance from the sensor, viewing angles, or occlusions. This will lead the initial query points to be either not located near them or sparsely located, leading to large false negative problems. A potential promising direction to address this issue is the coarse-to-fine architecture that combines detection and segmentation. For example, the commonly

used voxel grid-based feature map is known to handle the outdoor scene better for coarse localisation, such as detection, by learning the surrounding information, although it is unable to provide detailed features for fine-grained segmentation masks. The coarse boxes can then be refined using deeper features with the careful design choice to be able to cut the error accumulation from the boxes as addressed in Chapters 4.

Chapter 4 explores methods for acquiring annotations from motion cues to improve continual object localisation. In particular, it addresses the issue of unsustainable human annotation in a continuously changing real-world road environment. The proposed method utilises optical flow, ego-motion, and expected object sizes of the moving vehicle to find moving objects. More specifically, to find candidates for moving objects, scene flow is used to compare motions with the egomotion to find independent motions. Each region with independent motions in the point cloud is then extracted and used for box fitting. The fitted boxes are tracked for multiple frames to check consistency. The surviving boxes are used to train an object detection model with heavy augmentation for colour and size in order to detect the same type of objects that are both static and moving. Despite its applicability, a remaining challenge to be addressed is that its performance is upper bounded by the motion estimation models, which has clear advantages and disadvantages due to the inherent characteristic of the sensor that they use, *i.e.* camera. More specifically, the dense pixel-based features that cameras produce normally show robust performance in different shapes and colours of objects. However, it is unable to deliver the same robust features during the night when the light condition changes. This challenge can be potentially addressed by combining camera-based motion estimation with LiDAR-based motion estimation, which shows robust performance regardless of the light condition when the environment is not complex and does not have many moving objects.

Chapter 5 studies a domain adaptation method to effectively use existing labels and pseudo labels from Chapter 4 for continuous object localisation. In order to learn the domain gaps caused by different object sizes, shapes, and densities, the proposed method first divides existing labels into groups that have different features that cause domain gaps in a data-driven manner. Since each group contains objects of different sizes, shapes, and densities, it encourages the learning of internal variation from available annotations that reduce false negatives in the target domain. Furthermore, the pseudo-labels collected from the target domain are used to update groups that can best represent the changing environment. The performance of the proposed method is evaluated in challenging outdoor datasets such as Waymo, NuScenes, and KITTI. A lesson learned during the project is the

importance of continuous learning for domain adaptative detection. Although the current domain adaptation technique is advancing, the methods require the initial inter-domain gap to be in tangible distance. For example, even in the same region, the change in weather creates challenges if the system is directly given the dataset under sunny weather as the source domain and the dataset under rainy weather as the target domain, as objects in similar categories could exhibit substantially different features due to the change in light condition or object surface's reflectance. However, when the system is given sunny weather as the source and the transition period from sunny weather to rainy weather as the target, the domain adaptive detection shows robust performance, as the two environments exhibit higher similarities, which minimises the false negatives during the pseudo-label collection process. Then, the self-trained system can be adapted to the rainy weather as the target domain. This highlights the importance of continuous learning for the real-world deployment of the point cloud-based object detection system.

## 6.1 Potential Future Research Directions

The work of my DPhil opens up several interesting directions.

### 6.1.1 Multi-modal Object Localisation

Sensors have their own characteristics, which means one type of sensor can produce distinctive object features while other sensors cannot, depending on the environment. For instance, LiDAR produces good features for precise depth but may struggle in adverse weather conditions due to adverse reflection, while cameras can capture detailed visual information but may falter in low visibility conditions.

In order to achieve reliable and long-term object localization, it is crucial to integrate multiple sensor modalities, leveraging their complementary strengths to enhance overall robustness. An expected challenge is to determine which sensors to rely on in different scenarios, as some sensors could produce noisy signals under certain conditions. This noise can adversely impact the performance of detection and segmentation methods, leading to inaccuracies.

Another interesting aspect would be to quantitatively assess how each sensor and its specific characteristics contribute to performance in different environments. This involves conducting detailed studies to measure the impact of each sensor modality on the accuracy and reliability of object localization under various conditions. Such research can inform the development of strategies for optimal

sensor fusion, where the strengths of different sensors can be systematically maximized, and their weaknesses are minimized. This quantitative analysis has been relatively underexplored but holds the potential to significantly advance the field of long-term multi-modal object localization. Moreover, the findings from this study could significantly contribute to the development of more effective sensor fusion algorithms and next-generation sensors.

### 6.1.2 Tracking and Reidentification

One of the challenges in long-term object localization is outdoor reidentification, which requires a robust feature for the association of objects that regularly disappear in the scene or occasionally due to occlusion. Unlike state-of-the-art methods that have been effectively deployed for reidentification in controlled environments, such as indoor face recognition, long-term reidentification in dynamic outdoor environments poses significant challenges for real-world applications due to several challenges.

First, the outdoor environment is inherently dynamic and unpredictable compared to the relatively controlled settings for face recognition. For instance, variables such as changing lighting conditions and weather make it difficult to consistently capture and utilize distinctive features for identifying individuals.

Another challenge lies in the necessity to selectively ignore certain features that are not reliable for reidentification. For example, in pedestrian identification, an individual's clothing can vary significantly day by day. Such changes can alter the visual perception of the individual in RGB features and modify the pattern readings from range sensors, leading to potential false reidentifications. To address this challenge, exploring novel features, such as behavioural patterns of individuals interacting with the environments, could provide a robust feature for reidentification. This research direction promises the way for future developments in autonomous systems, surveillance, and various applications for outdoor scenarios.

# Bibliography


[1] Lucas R Agostinho, Nuno M Ricardo, Maria I Pereira, Antoine Hiolle, and Andry M Pinto. A practical survey on visual odometry for autonomous driving in challenging scenarios and conditions. *IEEE Access*, 10:72182–72205, 2022.

[2] Salwa Al Khatib, Mohamed El Amine Boudjoghra, Jean Lahoud, and Fahad Shahbaz Khan. 3d instance segmentation via enhanced spatial and semantic supervision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 541–550, October 2023.

[3] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017.

[4] Eduardo Arnold, Omar Y Al-Jarrah, Mehrdad Dianati, Saber Fallah, David Oxtoby, and Alex Mouzakitis. A Survey on 3D Object Detection Methods for Autonomous Driving Applications. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3782–3795, 2019.

[5] Zhipeng Bao*, Pavel Tokmakov*, Allan Jabri, Yu-Xiong Wang, Adrien Gaidon, and Martial Hebert. Discovering Objects that Can Move. In *CVPR*, 2022.

[6] Stefan Baur, Frank Moosmann, and Andreas Geiger. Liso: Lidar-only self-supervised 3d object detection. *arXiv preprint arXiv:2403.07071*, 2024.

[7] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[8] Thomas Brox and Jitendra Malik. Object Segmentation by Long Term Analysis of Point Trajectories. In *ECCV*, 2010.

[9] BS-EN-ISO. Basic human body measurements for technological design: Worldwide and regional design ranges for use in product standards. *British Standards Institute*, 7250-3, 2015.

[10] Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. MONet: Unsupervised Scene Decomposition and Representation. *arXiv:1901.11390*, 2019.

[11] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A Naturalistic Open Source Movie for Optical Flow Evaluation. In *ECCV*, 2012.

[12] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[13] Mark Campbell, Magnus Egerstedt, Jonathan P How, and Richard M Murray. Autonomous driving in urban environments: approaches, lessons and challenges. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1928):4649–4672, 2010.

[14] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[15] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[16] Chaoqi Chen, Weiping Xie, Wenbing Huang, Yu Rong, Xinghao Ding, Yue Huang, Tingyang Xu, and Junzhou Huang. Progressive feature alignment for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 627–636, 2019.

[17] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4974–4983, 2019.

[18] Meida Chen, Qingyong Hu, Zifan Yu, Hugues Thomas, Andrew Feng, Yu Hou, Kyle McCullough, Fengbo Ren, and Lucio Soibelman. Stpls3d: A large-scale synthetic and real aerial photogrammetry 3d point cloud dataset. *arXiv preprint arXiv:2203.09065*, 2022.

[19] Shaoyu Chen, Jiemin Fang, Qian Zhang, Wenyu Liu, and Xinggang Wang. Hierarchical aggregation for 3d instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15467–15476, 2021.

[20] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2147–2156, 2016.

[21] Xieyuanli Chen, Shijie Li, Benedikt Mersch, Louis Wiesmann, Jürgen Gall, Jens Behley, and Cyrill Stachniss. Moving Object Segmentation in 3D LiDAR Data: A Learning-Based Approach Exploiting Sequential Data. 6(4):6529–6536, 2021.

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

[23] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia. Voxelnext: Fully sparse voxelnet for 3d object detection and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21674–21683, June 2023.

[24] Zhuoxiao Chen, Yadan Luo, Zheng Wang, Mahsa Baktashmotlagh, and Zi Huang. Revisiting domain-adaptive 3d object detection by reliable, diverse and class-balanced pseudo-labeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3714–3726, 2023.

[25] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1290–1299, 2022.

[26] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *Advances in Neural Information Processing Systems*, 34:17864–17875, 2021.

[27] Tianheng Cheng, Xinggang Wang, Lichao Huang, and Wenyu Liu. Boundary-preserving mask r-cnn. In *Computer Vision–ECCV 2020: 16th European*

*Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 660–676. Springer, 2020.

[28] Junsuk Choe*, Seong Joon Oh*, Seungho Lee, Sanghyuk Chun, Zeynep Akata, and Hyunjung Shim. Evaluating Weakly Supervised Object Localization Methods Right. In *CVPR*, 2020.

[29] Junsuk Choe and Hyunjung Shim. Attention-based Dropout Layer for Weakly Supervised Object Localization. In *CVPR*, 2019.

[30] Spconv Contributors. Spconv: Spatially sparse convolution library. `https://github.com/traveller59/spconv`, 2022.

[31] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[32] Xiyang Dai, Yinpeng Chen, Jianwei Yang, Pengchuan Zhang, Lu Yuan, and Lei Zhang. Dynamic detr: End-to-end object detection with dynamic attention. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2988–2997, 2021.

[33] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 1201–1209, 2021.

[34] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018.

[35] Shichao Dong, Guosheng Lin, and Tzu-Yi Hung. Learning regional purity for instance segmentation on 3d point clouds. In *European Conference on Computer Vision*, pages 56–72. Springer, 2022.

[36] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6569–6578, 2019.

[37] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. CenterNet: Keypoint Triplets for Object Detection. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[38] Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. Genesis: Generative Scene Inference and Sampling with Object-Centric Latent Representations. In *ICLR*, 2020.

[39] Emeç Erçelik, Ekim Yurtsever, Mingyu Liu, Zhijie Yang, Hanzhen Zhang, Pınar Topçam, Maximilian Listl, Yılmaz Kaan Caylı, and Alois Knoll. 3d object detection with a self-supervised lidar scene flow backbone. In *European Conference on Computer Vision*, pages 247–265. Springer, 2022.

[40] SM Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Koray Kavukcuoglu, and Geoffrey E Hinton. Attend, Infer, Repeat: Fast Scene Understanding with Generative Models. In *NeurIPS*, 2016.

[41] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338, 2010.

[42] Lue Fan, Ziqi Pang, Tianyuan Zhang, Yu-Xiong Wang, Hang Zhao, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Embracing single stride 3d object detector with sparse transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8458–8468, 2022.

[43] Katerina Fragkiadaki, Geng Zhang, and Jianbo Shi. Video Segmentation by Tracing Discontinuities in a Trajectory Embedding. In *CVPR*, 2012.

[44] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[45] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[46] Wei Gao, Fang Wan, Xingjia Pan, Zhiliang Peng, Qi Tian, Zhenjun Han, Bolei Zhou, and Qixiang Ye. TS-CAM: Token Semantic Coupled Attention Map for Weakly Supervised Object Localization. In *ICCV*, 2021.

[47] Yulu Gao, Yifan Sun, Xudong Ding, Chuyang Zhao, and Si Liu. Easedetr: Easing the competition among object queries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17282–17291, June 2024.

[48] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[49] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, 2012.

[50] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018.

[51] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Chris Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-Object Representation Learning with Iterative Variational Inference. 2019.

[52] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. Occuseg: Occupancy-aware 3d instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2940–2949, 2020.

[53] Adam W Harley, Yiming Zuo*, Jing Wen*, Ayush Mangal, Shubhankar Potdar, Ritwick Chaudhry, and Katerina Fragkiadaki. Track, Check, Repeat: An EM Approach to Unsupervised Tracking. In *CVPR*, 2021.

[54] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

[56] Tong He, Chunhua Shen, and Anton Van Den Hengel. Dyco3d: Robust instance segmentation of 3d point clouds through dynamic convolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 354–363, 2021.

[57] Tong He, Wei Yin, Chunhua Shen, and Anton van den Hengel. Pointinst3d: Segmenting 3d instances by points. In *European Conference on Computer Vision*, pages 286–302. Springer, 2022.

[58] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[59] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4421–4430, 2019.

[60] Xiuquan Hou, Meiqin Liu, Senlin Zhang, Ping Wei, and Badong Chen. Salience detr: Enhancing detection transformer with hierarchical salience filtering refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17574–17583, 2024.

[61] Q. Hu, D. Liu, and W. Hu. Density-Insensitive Unsupervised Domain Adaption on 3D Object Detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[62] Qianjiang Hu, Daizong Liu, and Wei Hu. Density-insensitive unsupervised domain adaption on 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17556–17566, 2023.

[63] Xin Huang, Xinxin Wang, Wenyu Lv, Xiaying Bai, Xiang Long, Kaipeng Deng, Qingqing Dang, Shumin Han, Qiwen Liu, Xiaoguang Hu, et al. Pp-yolov2: A practical object detector. *arXiv preprint arXiv:2104.10419*, 2021.

[64] Yuhao Huang, Sanping Zhou, Junjie Zhang, Jinpeng Dong, and Nanning Zheng. Voxel or pillar: Exploring efficient point cloud representation for 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 2426–2435, 2024.

[65] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[66] Zhaoyang Huang, Xiaoyu Shi, Chao Zhang, Qiang Wang, Ka Chun Cheung, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer: A transformer architecture for optical flow. In *European conference on computer vision*, pages 668–685. Springer, 2022.

[67] Junhwa Hur and Stefan Roth. Self-supervised monocular scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7396–7405, 2020.

[68] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.

[69] Jisoo Jeong, Hyojin Park, and Nojun Kwak. Enhancement of ssd by concatenating feature maps for object detection. *arXiv preprint arXiv:1705.09587*, 2017.

[70] Jindong Jiang*, Sepehr Janghorbani*, Gerard de Melo, and Sungjin Ahn. SCALOR: Generative World Models with Scalable Object Representations. In *ICLR*, 2020.

[71] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and Pattern recognition*, pages 4867–4876, 2020.

[72] Xiaohui Jiang, Shuailin Li, Yingfei Liu, Shihao Wang, Fan Jia, Tiancai Wang, Lijin Han, and Xiangyu Zhang. Far3d: Expanding the horizon for surround-view 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 2561–2569, 2024.

[73] Yanqin Jiang, Li Zhang, Zhenwei Miao, Xiatian Zhu, Jin Gao, Weiming Hu, and Yu-Gang Jiang. Polarformer: Multi-camera 3d object detection with polar transformer. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 37, pages 1042–1050, 2023.

[74] Xin Jin, Kai Liu, Cong Ma, Ruining Yang, Fei Hui, and Wei Wu. Swiftpillars: High-efficiency pillar encoder for lidar-based 3d detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 2625–2633, 2024.

[75] Glenn Jocher, Alex Stoken, Ayush Chaurasia, Jirka Borovec, Yonghye Kwon, Kalen Michael, Liu Changyu, Jiacong Fang, Piotr Skalski, Adam Hogan, et al. ultralytics/yolov5: v6. 0-yolov5n'nano'models, roboflow integration, tensorflow export, opencv dnn support. *Zenodo*, 2021.

[76] Rahima Khanam and Muhammad Hussain. Yolov11: An overview of the key architectural enhancements. *arXiv preprint arXiv:2410.17725*, 2024.

[77] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[78] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9799–9808, 2020.

[79] Maksim Kolodiazhnyi, Danila Rukhovich, Anna Vorontsova, and Anton Konushin. Top-down beats bottom-up in 3d instance segmentation. *arXiv preprint arXiv:2302.02871*, 2023.

[80] Maxim Kolodiazhnyi, Anna Vorontsova, Anton Konushin, and Danila Rukhovich. Oneformer3d: One transformer for unified point cloud segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20943–20953, 2024.

[81] Jason Ku, Alex D Pon, and Steven L Waslander. Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11867–11876, 2019.

[82] Xin Lai, Yuhui Yuan, Ruihang Chu, Yukang Chen, Han Hu, and Jiaya Jia. Mask-attention-free transformer for 3d instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3693–3703, October 2023.

[83] Loic Landrieu and Mohamed Boussaha. Point cloud oversegmentation with graph-structured deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7440–7449, 2019.

[84] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4558–4567, 2018.

[85] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019.

[86] Itai Lang, Asaf Manor, and Shai Avidan. Samplenet: Differentiable point cloud sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7578–7588, 2020.

[87] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)*, pages 734–750, 2018.

[88] Jungwook Lee, Sean Walsh, Ali Harakeh, and Steven L Waslander. Leveraging pre-trained 3d object detection models for fast ground truth generation. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2504–2510. IEEE, 2018.

[89] Ville V Lehtola, Harri Kaartinen, Andreas Nüchter, Risto Kaijaluoto, Antero Kukko, Paula Litkey, Eija Honkavaara, Tomi Rosnell, Matti T Vaaja, Juho-Pekka Virtanen, et al. Comparison of the selected state-of-the-art 3d indoor scanning and point cloud generation methods. *Remote sensing*, 9(8):796, 2017.

[90] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 163–168. IEEE, 2011.

[91] Feng Li, Ailing Zeng, Shilong Liu, Hao Zhang, Hongyang Li, Lei Zhang, and Lionel M. Ni. Lite detr: An interleaved multi-scale encoder for efficient detr. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18558–18567, June 2023.

[92] Jianan Li, Shaocong Dong, Lihe Ding, and Tingfa Xu. Mssvt++: Mixed-scale sparse voxel transformer with center voting for 3d object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2023.

[93] Yanwei Li, Yilun Chen, Xiaojuan Qi, Zeming Li, Jian Sun, and Jiaya Jia. Unifying voxel-based representation with transformer for 3d object detection. *Advances in Neural Information Processing Systems*, 35:18442–18455, 2022.

[94] Ziyu Li, Jingming Guo, Tongtong Cao, Bingbing Liu, and Wankou Yang. GPA-3D: Geometry-aware Prototype Alignment for Unsupervised Domain Adaptive 3D Object Detection from Point Clouds. In *International Conference on Computer Vision (ICCV)*, 2023.

[95] Zhihao Liang, Zhihao Li, Songcen Xu, Mingkui Tan, and Kui Jia. Instance segmentation in 3d scenes using semantic superpoint tree networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2783–2792, 2021.

[96] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *CVPR*, 2017.

[97] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[98] Hong Liu, Mingsheng Long, Jianmin Wang, and Yu Wang. Learning to adapt to evolving domains. *Advances in Neural Information Processing Systems*, 33:22338–22348, 2020.

[99] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[100] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[101] Xianpeng Liu, Ce Zheng, Ming Qian, Nan Xue, Chen Chen, Zhebin Zhang, Chen Li, and Tianfu Wu. Multi-view attentive contextualization for multi-view 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16688–16698, 2024.

[102] Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petr: Position embedding transformation for multi-view 3d object detection. In *European Conference on Computer Vision*, pages 531–548. Springer, 2022.

[103] Yingfei Liu, Junjie Yan, Fan Jia, Shuailin Li, Aqi Gao, Tiancai Wang, and Xiangyu Zhang. Petrv2: A unified framework for 3d perception from multi-camera images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3262–3272, 2023.

[104] Youquan Liu, Lingdong Kong, Jun Cen, Runnan Chen, Wenwei Zhang, Liang Pan, Kai Chen, and Ziwei Liu. Segment any point cloud sequences by distilling vision foundation models. *Advances in Neural Information Processing Systems*, 36, 2024.

[105] Zechen Liu, Zizhang Wu, and Roland Tóth. Smoke: Single-stage monocular 3d object detection via keypoint estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 996–997, 2020.

[106] Francesco Locatello*, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf*. Object-Centric Learning with Slot Attention. 2020.

[107] The Best Bike Lock. What size shed do I need for my bikes? Available online at `https://thebestbikelock.com/bike-storage-ideas/best-bike-storage-shed/what-size-shed-for-bikes` (as of 12th September 2022).

[108] Xiang Long, Kaipeng Deng, Guanzhong Wang, Yang Zhang, Qingqing Dang, Yuan Gao, Hui Shen, Jianguo Ren, Shumin Han, Errui Ding, et al. Pp-yolo: An effective and efficient implementation of object detector. *arXiv preprint arXiv:2007.12099*, 2020.

[109] Jiahao Lu, Jiacheng Deng, Chuxin Wang, Jianfeng He, and Tianzhu Zhang. Query refinement transformer for 3d instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 18516–18526, October 2023.

[110] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. 36(1):3–15, 2017.

[111] Anas Mahmoud, Jordan S. K. Hu, and Steven L. Waslander. Dense voxel fusion for 3d object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 663–672, January 2023.

[112] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 922–928. IEEE, 2015.

[113] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017.

[114] Mahyar Najibi, Jingwei Ji, Yin Zhou, Charles R Qi, Xinchen Yan, Scott Ettinger, and Dragomir Anguelov. Motion inspired unsupervised perception and prediction in autonomous driving. In *European Conference on Computer Vision*, pages 424–443. Springer, 2022.

[115] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *18th international conference on pattern recognition (ICPR'06)*, volume 3, pages 850–855. IEEE, 2006.

[116] Jing Nie, Rao Muhammad Anwer, Hisham Cholakkal, Fahad Shahbaz Khan, Yanwei Pang, and Ling Shao. Enriched feature guided refinement network for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9537–9546, 2019.

[117] NimbleFins. What Are The Average Dimensions Of A Car In The UK? Available online at `https://www.nimblefins.co.uk/cheap-car-insurance/average-car-dimensions` (as of 12th September 2022).

[118] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[119] Xingjia Pan*, Yingguo Gao*, Zhiwen Lin*, Fan Tang, Weiming Dong, Haolei Yuan, Feiyue Huang, and Changsheng Xu. Unveiling the Potential of Structure Preserving for Weakly Supervised Object Localization. In *CVPR*, 2021.

[120] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra R-CNN: Towards Balanced Learning for Object Detection. In *CVPR*, 2019.

[121] Kyeong-Beom Park, Minseok Kim, Sung Ho Choi, and Jae Yeol Lee. Deep learning-based smart task assistance in wearable augmented reality. *Robotics and Computer-Integrated Manufacturing*, 63:101887, 2020.

[122] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[123] Xidong Peng, Xinge Zhu, and Yuexin Ma. Cl3d: Unsupervised domain adaptation for cross-lidar 3d detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 2047–2055, 2023.

[124] Yifan Pu, Weicong Liang, Yiduo Hao, Yuhui Yuan, Yukang Yang, Chao Zhang, Han Hu, and Gao Huang. Rank-detr for high quality object detection. *Advances in Neural Information Processing Systems*, 36, 2024.

[125] Charles Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

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

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

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

[129] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[130] Abdullah Rashwan, Agastya Kalra, and Pascal Poupart. Matrix nets: A new deep architecture for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

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

[132] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[133] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[134] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[135] Thomas Roddick, Alex Kendall, and Roberto Cipolla. Orthographic feature transform for monocular 3d object detection. *arXiv preprint arXiv:1811.08188*, 2018.

[136] Byungseok Roh, JaeWoong Shin, Wuhyun Shin, and Saehoon Kim. Sparse detr: Efficient end-to-end object detection with learnable sparsity. *arXiv preprint arXiv:2111.14330*, 2021.

[137] Wonseok Roh, Hwanhee Jung, Giljoo Nam, Jinseop Yeom, Hyunje Park, Sang Ho Yoon, and Sangpil Kim. Edge-aware 3d instance segmentation network with intelligent semantic prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20644–20653, 2024.

[138] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[139] Bryan C Russell, Alexei A Efros, Josef Sivic, William T Freeman, and Andrew Zisserman. Using Multiple Segmentations to Discover Objects and their Extent in Image Collections. In *CVPR*, 2006.

[140] Cristiano Saltori, Stéphane Lathuiliére, Nicu Sebe, Elisa Ricci, and Fabio Galasso. Sf-uda 3d: Source-free unsupervised domain adaptation for lidar-based 3d object detection. In *2020 International Conference on 3D Vision (3DV)*, pages 771–780. IEEE, 2020.

[141] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3d for 3d semantic instance segmentation. *arXiv preprint arXiv:2210.03105*, 2022.

[142] Jenny Seidenschwarz, Aljosa Osep, Francesco Ferroni, Simon Lucey, and Laura Leal-Taixé. Semoli: What moves together belongs together. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14685–14694, 2024.

[143] Xing Shen, Jirui Yang, Chunbo Wei, Bing Deng, Jianqiang Huang, Xian-Sheng Hua, Xiaoliang Cheng, and Kewei Liang. Dct-mask: Discrete cosine transform mask representation for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8720–8729, June 2021.

[144] Yaqi Shen, Le Hui, Jin Xie, and Jian Yang. Self-supervised 3d scene flow estimation guided by superpoints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5271–5280, 2023.

[145] Guangsheng Shi, Ruifeng Li, and Chao Ma. Pillarnet: Real-time and high-performance pillar-based 3d object detection. In *European Conference on Computer Vision*, pages 35–52. Springer, 2022.

[146] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10529–10538, 2020.

[147] Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo, Jianping Shi, Xiaogang Wang, and Hongsheng Li. PV-RCNN++: Point-Voxel Feature Set Abstraction With Local Vector Representation for 3D Object Detection. *arXiv:2102.00463*, 2021.

[148] Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. *International Journal of Computer Vision*, 131(2):531–551, 2023.

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

[150] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware

and part-aggregation network. *IEEE transactions on pattern analysis and machine intelligence*, 43(8):2647–2664, 2020.

[151] Josef Sivic, Bryan C Russell, Alexei A Efros, Andrew Zisserman, and William T Freeman. Discovering objects and their location in images. In *ICCV*, 2005.

[152] Jiahao Sun, Chunmei Qing, Junpeng Tan, and Xiangmin Xu. Superpoint transformer for 3d scene instance segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 2393–2401, 2023.

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

[154] Pei Sun, Mingxing Tan, Weiyue Wang, Chenxi Liu, Fei Xia, Zhaoqi Leng, and Dragomir Anguelov. Swformer: Sparse window transformer for 3d object detection in point clouds. In *European Conference on Computer Vision*, pages 426–442. Springer, 2022.

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

[156] Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *ECCV*, 2020.

[157] Zachary Teed and Jia Deng. Raft-3d: Scene flow using rigid-motion embeddings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8375–8384, 2021.

[158] Khoi Nguyen Tuan Duc Ngo, Binh-Son Hua. Isbnet: a 3d point cloud instance segmentation network with instance-aware sampling and box-aware dynamic convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[159] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[160] Thang Vu, Kookhoi Kim, Tung M Luu, Thanh Nguyen, and Chang D Yoo. Softgroup for 3d instance segmentation on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2708–2717, 2022.

[161] Haiyang Wang, Chen Shi, Shaoshuai Shi, Meng Lei, Sen Wang, Di He, Bernt Schiele, and Liwei Wang. Dsvt: Dynamic sparse voxel transformer with rotated sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13520–13529, June 2023.

[162] Haiyang Wang, Chen Shi, Shaoshuai Shi, Meng Lei, Sen Wang, Di He, Bernt Schiele, and Liwei Wang. Dsvt: Dynamic sparse voxel transformer with rotated sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13520–13529, 2023.

[163] He Wang, Yezhen Cong, Or Litany, Yue Gao, and Leonidas J Guibas. 3DIoUMatch: Leveraging IoU Prediction for Semi-Supervised 3D Object Detection. In *CVPR*, 2021.

[164] Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19795–19806, October 2023.

[165] Yan Wang, Xiangyu Chen, Yurong You, Li Erran Li, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. Train in germany, test in the usa: Making 3d object detectors generalize. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11713–11723, 2020.

[166] Yingming Wang, Xiangyu Zhang, Tong Yang, and Jian Sun. Anchor detr: Query design for transformer-based detector. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 2567–2575, 2022.

[167] Yue Wang, Vitor Campagnolo Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *Conference on Robot Learning*, pages 180–191. PMLR, 2022.

[168] Zhenyu Wang, Ya-Li Li, Xi Chen, Hengshuang Zhao, and Shengjin Wang. Uni3detr: Unified 3d detection transformer. *Advances in Neural Information Processing Systems*, 36, 2024.

[169] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Prisacariu, and Min Chen. Flownet3d++: Geometric losses for deep scene flow estimation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 91–98, 2020.

[170] Markus Weber, Max Welling, and Pietro Perona. Towards Automatic Discovery of Object Categories. In *CVPR*, 2000.

[171] Yi Wei, Zibu Wei, Yongming Rao, Jiaxin Li, Jie Zhou, and Jiwen Lu. Lidar distillation: Bridging the beam-induced domain gap for 3d object detection. In *European Conference on Computer Vision*, pages 179–195. Springer, 2022.

[172] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4840–4851, June 2024.

[173] Yizheng Wu, Min Shi, Shuaiyuan Du, Hao Lu, Zhiguo Cao, and Weicai Zhong. 3d instances as 1d kernels. In *European Conference on Computer Vision*, pages 235–252. Springer, 2022.

[174] Yutian Wu, Yueyu Wang, Shuwei Zhang, and Harutoshi Ogai. Deep 3D Object Detection Networks Using LiDAR data: A Review. *IEEE Sensors Journal*, 21(2):1152–1171, 2020.

[175] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single shot instance segmentation with polar representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12193–12202, 2020.

[176] Yiming Xie, Huaizu Jiang, Georgia Gkioxari, and Julian Straub. Pixel-aligned recurrent queries for multi-view 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18370–18380, 2023.

[177] Jenny Xu and Steven L Waslander. Hypermodest: Self-supervised 3d object detection with confidence score filtering. In *2023 20th Conference on Robots and Vision (CRV)*, pages 217–224. IEEE, 2023.

[178] Qiangeng Xu, Yin Zhou, Weiyue Wang, Charles R Qi, and Dragomir Anguelov. Spg: Unsupervised domain adaptation for 3d object detection via semantic point generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15446–15456, 2021.

[179] Haolan Xue, Chang Liu, Fang Wan, Jianbin Jiao, Xiangyang Ji, and Qixiang Ye. DANet: Divergent Activation for Weakly Supervised Object Localization. In *ICCV*, 2019.

[180] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[181] Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham, and Niki Trigoni. Learning object bounding boxes for 3d instance segmentation on point clouds. *Advances in neural information processing systems*, 32, 2019.

[182] Jiao Yang and Liqi Wang. Feature fusion and enhancement for single shot multibox detector. In *2019 Chinese automation congress (CAC)*, pages 2766–2770. IEEE, 2019.

[183] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. St3d++: Denoised self-training for unsupervised domain adaptation on 3d object detection. *arXiv preprint arXiv:2108.06682*, 2021.

[184] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. St3d: Self-training for unsupervised domain adaptation on 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10368–10378, 2021.

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

[186] Lei Yao, Yi Wang, Moyun Liu, and Lap-Pui Chau. Sgiformer: Semantic-guided and geometric-enhanced interleaving transformer for 3d instance segmentation. *arXiv preprint arXiv:2407.11564*, 2024.

[187] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3947–3956, 2019.

[188] Zeng Yihan, Chunwei Wang, Yunbo Wang, Hang Xu, Chaoqiang Ye, Zhen Yang, and Chao Ma. Learning transferable features for point cloud detection via 3d contrastive co-training. *Advances in Neural Information Processing Systems*, 34:21493–21504, 2021.

[189] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[190] Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and Steven Lovegrove. Star: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13144–13152, 2021.

[191] Biao Zhang and Peter Wonka. Point cloud instance segmentation using probabilistic embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8883–8892, 2021.

[192] Gang Zhang, Chen Junnan, Guohuan Gao, Jianmin Li, and Xiaolin Hu. Hednet: A hierarchical encoder-decoder network for 3d object detection in point clouds. *Advances in Neural Information Processing Systems*, 36, 2024.

[193] Gang Zhang, Xin Lu, Jingru Tan, Jianmin Li, Zhaoxiang Zhang, Quanquan Li, and Xiaolin Hu. Refinemask: Towards high-quality instance segmentation with fine-grained features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6861–6869, June 2021.

[194] Jingyi Zhang, Jiaxing Huang, Zhipeng Luo, Gongjie Zhang, Xiaoqin Zhang, and Shijian Lu. Da-detr: Domain adaptive detection transformer with information fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23787–23798, 2023.

[195] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Single-shot refinement neural network for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4203–4212, 2018.

[196] Shilong Zhang, Xinjiang Wang, Jiaqi Wang, Jiangmiao Pang, Chengqi Lyu, Wenwei Zhang, Ping Luo, and Kai Chen. Dense distinct query for end-to-end object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7329–7338, June 2023.

[197] Xiaolin Zhang, Yunchao Wei, and Yi Yang. Inter-Image Communication for Weakly Supervised Localization. In *ECCV*, 2020.

[198] Weiguang Zhao, Yuyao Yan, Chaolong Yang, Jianan Ye, Xi Yang, and Kaizhu Huang. Divide and conquer: 3d point cloud instance segmentation with point-wise binarization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 562–571, October 2023.

[199] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. Detrs beat yolos on real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16965–16974, June 2024.

[200] Liwen Zheng, Canmiao Fu, and Yong Zhao. Extend the shallow part of single shot multibox detector via convolutional neural network. In *Tenth International Conference on Digital Image Processing (ICDIP 2018)*, volume 10806, pages 287–293. SPIE, 2018.

[201] Min Zhong, Xinghao Chen, Xiaokang Chen, Gang Zeng, and Yunhe Wang. Maskgroup: Hierarchical point grouping and masking for 3d instance segmentation. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2022.

[202] Dingfu Zhou, Jin Fang, Xibin Song, Liu Liu, Junbo Yin, Yuchao Dai, Hongdong Li, and Ruigang Yang. Joint 3d instance segmentation and object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1839–1849, 2020.

[203] Kaichen Zhou, Jia-Xing Zhong, Sangyun Shin, Kai Lu, Yiyuan Yang, Andrew Markham, and Niki Trigoni. Dynpoint: Dynamic neural point for view synthesis. *Advances in Neural Information Processing Systems*, 36, 2024.

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

[205] Yuchen Zhou, Jiayuan Gu, Tung Yen Chiang, Fanbo Xiang, and Hao Su. Pointsam: Promptable 3d segmentation model for point clouds, 2024.

[206] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

[207] Yi Zhu, Yanzhao Zhou, Huijuan Xu, Qixiang Ye, David Doermann, and Jianbin Jiao. Learning Instance Activation Maps for Weakly Supervised Instance Segmentation. In *CVPR*, 2019.

[208] Zhuofan Zong, Guanglu Song, and Yu Liu. Detrs with collaborative hybrid assignments training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6748–6758, October 2023.