UNIVERSITÀ
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

Handling Domain Shift in 3D Point Cloud Perception

Cristiano Saltori

Advisor

Prof. Elisa Ricci
Fondazione Bruno Kessler
Università degli Studi di Trento

Co-Advisors

Prof. Nicu Sebe
Università degli Studi di Trento

Prof. Fabio Galasso
Sapienza Università di Roma

April 2024

Abstract

This thesis addresses the problem of domain shift in 3D point cloud perception. In the last decades, there has been tremendous progress in within-domain training and testing. However, the performance of perception models is affected when training on a source domain and testing on a target domain sampled from different data distributions. As a result, a change in sensor or geo-location can lead to a harmful drop in model performance. While solutions exist for image perception, addressing this problem in point clouds remains unresolved. The focus of this thesis is the study and design of solutions for mitigating domain shift in 3D point cloud perception. We identify several settings differing in the level of target supervision and the availability of source data. We conduct a thorough study of each setting and introduce a new method to solve domain shift in each configuration. In particular, we study three novel settings in domain adaptation and domain generalization and propose five new methods for mitigating domain shift in 3D point cloud perception. Our methods are used by the research community, and at the time of writing, some of the proposed approaches hold the state-of-the-art. In conclusion, this thesis provides a valuable contribution to the computer vision community, setting the groundwork for the development of future works in cross-domain conditions.

Keywords

[3D Point Cloud Perception, Domain Adaptation, Domain Generalization, Unsupervised Learning, Semi-Supervised Learning]

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Context

As humans, we possess an innate ability to perceive and understand our surrounding environment, utilizing our sensory inputs [12, 76]. Replicating this process in machines, such as robots and computers, has long been a focal point in Artificial Intelligence (AI) research, particularly within the Computer Vision (CV) field [43, 63]. Whether navigating a vehicle or walking along a street, our subconscious planning involves a detailed visual scene analysis. This includes understanding the scene by localizing objects, analyzing the scene, and tracking moving entities. In computer vision, these tasks include semantic segmentation, object detection, and multi-object tracking. Semantic segmentation is the task of assigning a specific class to each pixel or object within the input image [22, 170]. The objective is to generate a dense pixel-wise segmentation map where every pixel is associated with a class or object. For instance, in the context of an autonomous vehicle, relevant classes might include *vehicle*, *road*, *pedestrian*, and *sidewalk*. Differently, an indoor robot might segment images into classes such as *table*, *floor*, *chairs*, and *wardrobe*. Object detection differs from segmentation as it focuses on detecting and localizing instances of objects within the input image, disregarding classes associated with the

scene background [24, 16], such as *road* and *vegetation*. This process involves retrieving the boundaries of detected objects and classifying their respective classes. Multi-Object Tracking (MOT) foresees detecting and tracking multiple objects within a video sequence [212, 118]. The primary objective of MOT is to identify and locate objects of interest in each frame, subsequently associating them across frames to monitor their temporal movements. Critical challenges in MOT include occlusion, motion blur, and variations in object appearance. Typically, MOT is addressed through a tracking-by-detection pipeline, leveraging a detector for object retrieval and a data association module for linking instances between consecutive frames.

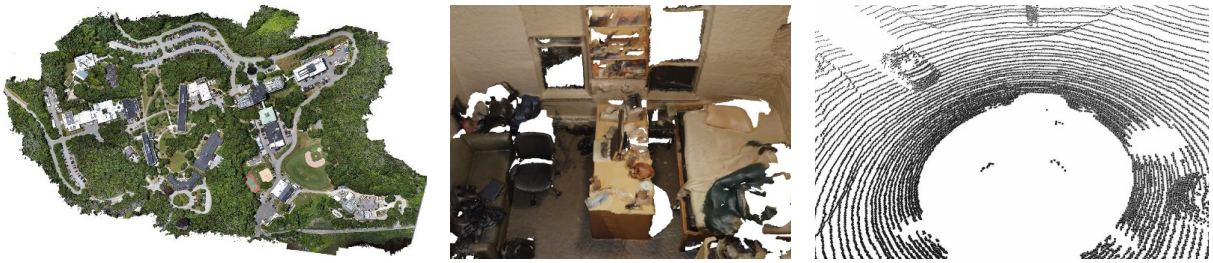
The tasks mentioned above have traditionally been explored within the domain of camera-based perception systems. Cameras are passive sensors, *i.e.*, they absorb and detect natural radiation or emissions from the target or the environment, such as sunlight or artificial light. They offer advantages in cost-effectiveness, portability, and easy integration into hardware architectures [148, 38]. However, cameras present two main limitations. Firstly, they lack precise distance measures [183], posing challenges for applications where accurately localizing objects in the 3D world is crucial, *e.g.*, autonomous driving or manufacturing robots. Although distance estimation methods exist, they encounter difficulties when applied to scenarios involving cross-traffic entities, especially in the context of monocular solutions. Secondly, cameras are susceptible to variations in lighting conditions, as they do not directly interact with the physical scene. Instead, they operate only on the resulting image, making them more vulnerable to adverse weather and lighting-related challenges. Issues such as low lighting, rain, and fog are known challenges in camera-based perception systems [239].

Light Detection and Ranging (LiDAR) sensors present an alternative to cameras. LiDARs are active sensors, *i.e.*, they interact with the sur-

rounding environment by emitting laser pulses and recording the reflected or scattered signals. After the acquisition, these scattered signals are processed to generate a 3D point cloud, namely, an unordered set of points that capture the geometries of the surrounding scene. In contrast to cameras, LiDAR sensors enable precise range measurements and are not dependent on lighting conditions [99]. Notably, during the 2007 DARPA Grand Challenge, a pivotal event in the field of autonomous driving, the top three winners [179, 128, 186] were mounting LiDAR sensors.

Early point cloud perception systems were model-based, relying on hand-crafted features [70]. However, their effectiveness and applicability were constrained by the limited computational capabilities of the hardware. Recent advances in hardware computational capabilities, the availability of large-scale datasets, and the improvement of optimization algorithms have driven a renewed interest in data-driven models, leading to the growth of deep learning. Deep learning is a sub-field of machine learning using Artificial Neural Networks (ANNs), algorithms inspired by the structure and functioning of the human brain. ANNs are composed of interconnected artificial neural layers that process information, learn, and allow predictions from data.

Due to their unprecedented performance, ANNs have found extensive applications in processing point clouds, including semantic segmentation [56, 27], object detection [166, 113], multi-object tracking [201, 79], instance and panoptic segmentation[6, 162], registration [197, 115], and simultaneous localization and mapping [39, 199]. However, the success of ANNs comes with inherent challenges. ANNs need sufficiently large datasets to train the network and achieve acceptable performance. Moreover, they require significant computational resources, often expensive, and have a *black box* data-driven nature, making it challenging to interpret predictions. One major challenge is that ANNs are prone to poor generalization under the



a) b) c)

Figure 1.1: Example of a point cloud acquired with a) photogrammetry, b) RGB-D cameras, and c) LiDAR sensor. Image credits to [54, 32, 9].

presence of *domain shift*. The problem of domain shift arises when training and test data is drawn from different domains, such as different cities, or by employing different sensors. As a result of domain shift, the network fits the training domain during training but generalizes poorly to the test domain. For instance, training a model on point clouds from Europe and testing it on data from the US may lead to low performance due to variations in vehicle types or urban contexts [196, 156]. Addressing domain shift is crucial in real-world applications where annotating data for each new domain is often unfeasible and expensive.

The following sections will delve into solutions for domain shift. Initially, we will provide an overview of point cloud acquisition in modern perception systems. Then, we will detail how 3D data can be processed and the main learning paradigms. Finally, we will present techniques for mitigating the domain shift in 3D point cloud perception.

1.2 3D point cloud acquisition

Point cloud acquisition techniques can be grouped into photogrammetry, RGB-D cameras, and LiDAR sensors, as illustrated in Fig. 1.1. Photogrammetry (Fig. 1.1a) involves obtaining reliable measurements from images via acquisition, matching, registration, refinement, and meshing. While suitable for topographic mapping, remote sensing, and architecture, its com-

putational demands and the need for specialized software may limit its use to some specific applications. RGB-D cameras (Fig. 1.1b) allow the acquisition of RGB images along with a depth map. Akin to standard cameras, they are lightweight, cost-effective, and easily integrated into hardware. However, they use infrared sensors for depth map retrieval, making them suitable mainly for indoor environments with stable lighting conditions. In contrast, Light Detection and Ranging (LiDAR) sensors are versatile and applicable in various scenarios, including large-scale acquisitions and indoor environments. Moreover, they can complement photogrammetry and RGB-D cameras to enhance the quality of final acquisitions.

A LiDAR sensor, also known as a laser scanner, operates as an active sensor, relying on the emission of pulsed light from a laser diode that travels until received by a sensor (Fig. 1.1c). The emitted signal, in the near-infrared range, employs the time-of-flight principle for distance measurement between emission and reception. LiDAR sensors are classified according to the type of point cloud they generate, resulting in 2D or 3D LiDAR. Alternatively, they can be classified according to their construction as rotary or solid-state LiDAR. In the case of 2D LiDAR, information is collected by projecting a single laser beam onto a rotating mirror perpendicular to the axis of rotation. On the other hand, 3D LiDAR provides a highly accurate 3D map of the environment by employing a set of diode lasers mounted on a rapidly rotating pod. The density of the point cloud obtained at each rotation is determined by the number of lasers, or laser beams, with current 3D LiDARs integrating 4 to 128 lasers or channels. For example, the famous VelodyneHDL64E is equipped with 64 laser beams.

These devices typically offer a horizontal FOV of 360° and a vertical FOV of 20–45 degrees, achieving centimeter-level accuracy. A recent addition is the solid-state LiDAR, which facilitates obtaining a 3D representation of the scene without any mobile parts. In solid-state LiDAR, a

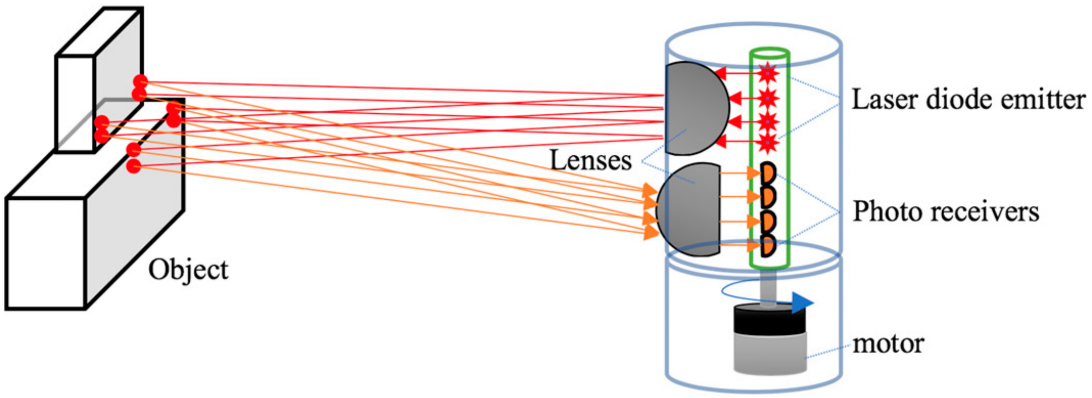


Figure 1.2: High-level overview of a LiDAR system. Image credits to [144].

micro-mirror circuit synchronizes with a laser beam to scan the horizontal field of view in multiple lines. The micro-mirror reflects the beam over a diffuser lens, creating a vertical line interacting with objects. The reflected light is captured by a lens and sent to a photodetector array, constructing the first line of a 3D matrix. This process is repeated until a point cloud of the scene is generated. Solid-state LiDARs have several advantages over rotary LiDAR: increased durability, reduced maintenance needs, and lower costs than rotary LiDAR. However, they tend to have a smaller FOV.

This thesis focuses on perception in 3D point clouds when acquired with 3D rotary LiDARs in driving environments. For this reason, we leave photogrammetry-based, multi-modal, and indoor perception for future works. Moreover, from now on, we will use the expression point cloud and 3D point cloud interchangeably.

1.3 Point cloud processing

The earlier section introduced point cloud acquisition. This section moves beyond the acquisition phase, presenting Deep Neural Network (DNN) architectures for point cloud processing. However, the main objective is not an exhaustive review of the DNN literature for point clouds but to offer

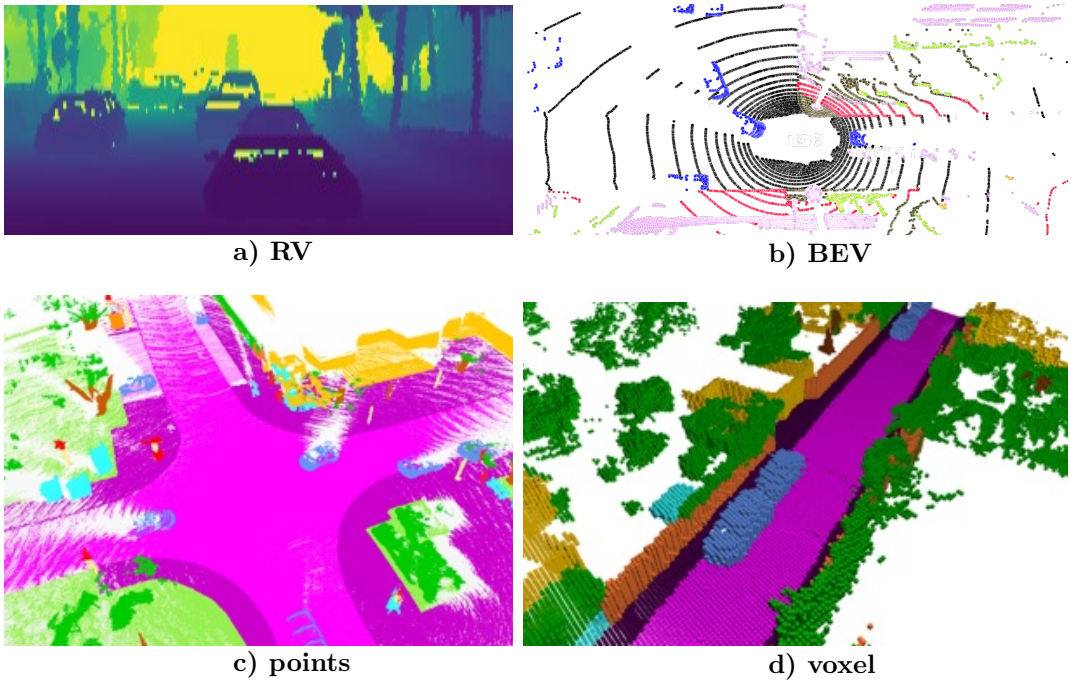


Figure 1.3: Examples of a) Range-View image (RV), b) Bird's Eye View image (BEV), c) a point cloud (points), d) a voxelized point cloud (voxel). Image credits to [8, 46].

insights into the main representations and network categories employed in point cloud processing. In contrast to images, where Convolutional Neural Networks (CNNs) dominate, point cloud processing is challenging due to the unordered and sparse nature of this data type, with density variations based on the distance from the sensor. The processing of point clouds relies on both the input representation and the final task. For example, in geometric deep learning, points are represented as a graph using Graph Neural Networks (GNNs) for data processing. Another example is robot navigation, where inference time is critical. Here, points are often projected into pseudo-images, enabling processing through lightweight CNN architectures. Let us move within the domain of 3D perception.

We can find three main types of architectures based on the input representation: point-based, projection-based, and voxel-based. Point-based architectures, one of the earliest network types developed for point cloud

processing, can learn feature representations directly from raw points without additional pre-processing steps. Early point-based architectures used point-wise Multi-Layer Perceptrons (MLP). For example, the pioneering PointNet [133] leveraged permutation-invariant operators such as point-wise MLP, spatial transformer networks, and pooling operators. Its successor, PointNet++[134], introduced a hierarchical spatial structure to enhance sensitivity to local geometric layout and incorporated shared 1D convolutional layers. More recent architectures build upon advances in the 2D field, introducing deformable convolutions [33, 178] and transformer networks [234, 188] for improved point cloud processing. The main limitation of point-based approaches lies in their computational demands when processing large-scale scans. Additionally, point-based architectures often show a slow convergence during training time.

Projection-based architectures involve a mapping step where 3D points are projected onto a dense 2D grid with specific features. Among the widely adopted projection-based representations for LiDAR data are Range-View (RV) and Bird's Eye View (BEV). RV projection (Fig.1.3a) maps input points into a spherical coordinate system. Following projection, each column shares an azimuth, and each row shares an inclination. These values denote the relative vertical and horizontal angles from the original input point. Pixel values contain the range (depth) of each corresponding point, the magnitude of the returned laser pulse (intensity), and other auxiliary information. In BEV projection (Fig.1.3b), points are top-down projected onto the horizontal x-z plane. During this projection, the x and z values of points define the occupancy grid. In BEV images, pixel values corresponding to occupied pixels contain the maximum height, laser pulse intensity, density of neighboring points, and auxiliary information. Following projection, these pseudo-images are processed using standard 2D CNN architectures. Early approaches [160] and [203] projected input points into

RV images and employed an AlexNet-inspired or a U-Net architecture for the tasks of 3D pedestrian detection and road-object segmentation, respectively. More recently, a trend has emerged towards enhancing the learning of better-projected images and leveraging contemporary 2D architectures. Notably, PointPillars [88] refines the projection strategy by learning a 2D pseudo-image where pixel values represent features derived from a point-based architecture. CenterPoint [223] learns a network to project input points into BEV feature maps, relying on SwinTransformer [107] for 3D object detection. Another example is PolarNet [232], which learns a BEV feature map over a polar grid to address the varying sparsity of LiDAR scans. Projection-based approaches offer the advantage of employing efficient 2D CNN architectures from image literature with fast training and efficient inference. However, these approaches suffer geometric information loss during the projection phase. In RV images, occluded objects may collapse into pixels with the same range, *e.g.*, a group of pedestrians. In BEV images, overlapping objects may collapse to the same pixel, *e.g.*, a tree and the underlying terrain. Additionally, selecting the correct projection strategy is crucial for best final performance.

Voxel-based architectures involve a voxelization phase, where 3D points are mapped into a quantized 3D grid of voxels (Fig.1.3d). Voxels are 3D base cubical units used to represent 3D structures. The size of a voxel is adjustable by varying the voxel size, corresponding to the quantization step size along each axis. During voxelization, the 3D space is partitioned into equidistant voxels based on the voxel size, and points are grouped according to the voxel they occupy. Furthermore, voxel values can include hand-crafted or learned features. Early approaches used 3D convolutional layers, analogous to 2D CNN, but employing 3D convolutional kernels for handling 3D data. VoxelNet [227] was the first approach within this category. After voxelization, it employed PointNet to learn input voxel features

and a 3D CNN for the primary task of 3D object detection. However, traditional 3D convolutions proved computationally and memory-intensive, as they did not account for the sparsity of a voxelized point cloud, computing features for occupied and empty regions. A pivotal change was the advent of 3D sparse convolutions, avoiding the computation of features for empty regions and reducing memory and computational costs. 3D sparse convolutions first appeared in SECOND [214] and gained popularity with SparseConv [57] and MinkowskiNet [28, 27]. Recent architectures further refine the voxelization phase and overall structure. For example, Cylinder3D [243] introduces a cylindrical voxelization strategy specific to LiDAR scans. Alternatively, VoTr [113] introduces the first sparse voxel-based transformer architecture designed for 3D object detection. Voxel-based architectures fully exploit the 3D geometric information of the input point cloud. However, the voxel size is a critical choice influencing the final efficiency and performance. A high resolution results in high computational and memory costs, while a low resolution may introduce strong geometric artifacts during voxelization. Another research direction explores the use of hybrid architectures. For example, PointGrid[92] integrates both voxel and point-based approaches, creating a 3D hybrid representation that combines discrete points and volumetric voxels. While hybrid architectures present an interesting direction, it is important to note that they use the strategies outlined above. Moreover, their application in 3D point cloud perception is relatively constrained. We recommend referring to [60] for a more in-depth discussion on hybrid architectures.

1.4 Learning paradigms

This section provides the background on the learning paradigms commonly used in machine learning (Fig. 1.4). The literature identifies three primary

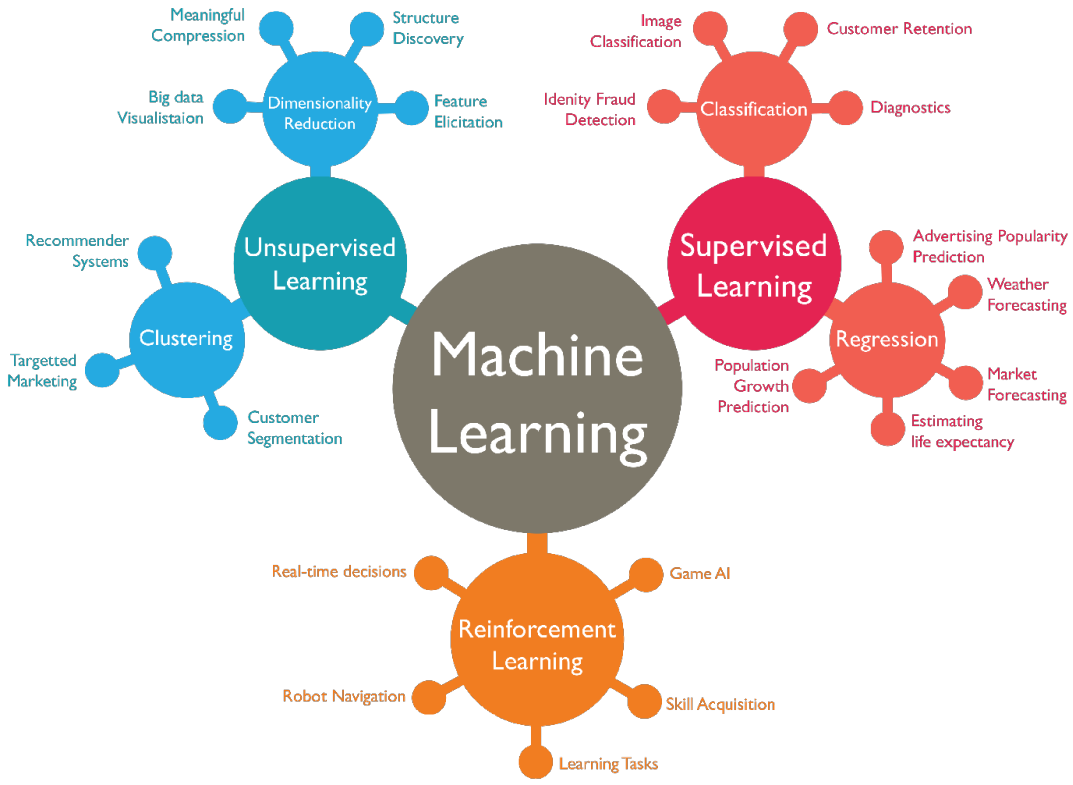


Figure 1.4: Overview of the learning paradigms in machine learning. Image credits to [3].

learning paradigms: supervised, unsupervised, and reinforcement learning.

The first and fundamental paradigm is supervised learning. It involves training a model using data paired with a source of supervision, such as annotations corresponding to each training sample. These annotations guide the learning process, helping to adjust the model parameters to ensure its output aligns with the provided annotations. The model is fed with data during training, and the output predictions are compared with the annotations to calculate the error. This error is then used to iteratively update the model parameters to minimize the discrepancy between predictions and annotations. This training process is repeated until the model achieves the desired level of accuracy over new, unseen data. Supervised learning finds extensive application across various tasks, from image classification and point cloud registration to audio-visual speech recognition. In

image classification, the training data consists of labeled images, and the supervision comes from the associated labels. In point cloud registration, the data include pairs of point clouds, with annotations representing the ground truth roto-translation matrices. In audio-visual speech recognition, the data contain pairs of audio and video, with supervision originating from text transcriptions.

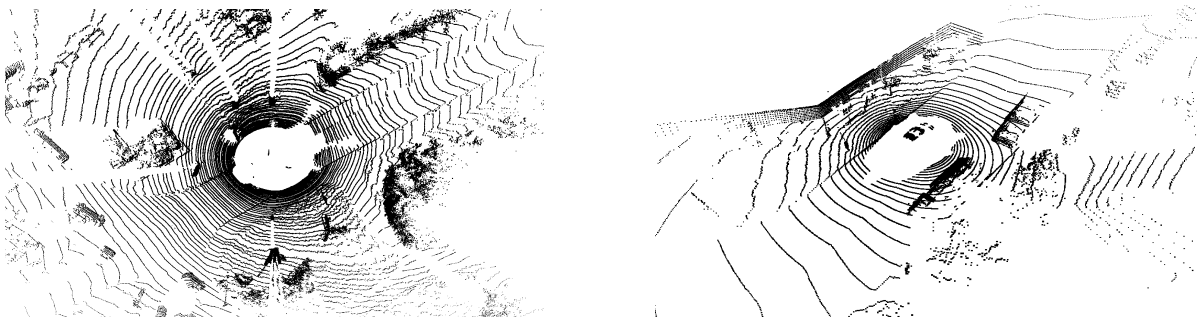
Unsupervised learning is the second fundamental paradigm in machine learning, focusing on training a model with unlabeled data to discover latent patterns within the training set. In this paradigm, the model processes the training data to reconstruct the input data or learn feature representations that best represent input data. Unlike supervised learning, where labeled data guide the learning process, unsupervised learning relies on alternative methods for evaluating the quality of learned representations. Standard evaluation methods include clustering algorithms and dimensionality reduction techniques. Following training, the acquired representations find utility in diverse downstream tasks, including visualization, compression, registration, matching, and localization. For example, in point cloud registration, a model trained with unsupervised loss can identify matching points between different views. Matches are then used to estimate transformation matrices for the final registration.

Between supervised and unsupervised lies an intermediate paradigm known as semi-supervised learning. In semi-supervised learning, only a portion or a few training samples are provided with supervision, while a large set of data is unsupervised. This paradigm can be further categorized into few-shot and one-shot learning based on the number of labeled samples. Semi-supervised learning leverages the strengths of both supervised and unsupervised paradigms, often offering enhanced performance over unsupervised approaches with limited annotated data. During training, labeled data guide the learning of the main task, while unlabeled data

contribute to robustness by learning an unsupervised objective, such as a reconstruction objective. Semi-supervised approaches are frequently employed to mitigate label dependency in various tasks, finding applications across domains ranging from image classification and clustering to more complex tasks like 3D point cloud perception.

Reinforcement learning is the third and final learning paradigm. Unlike the previous paradigms, reinforcement learning trains a model under an action-reward strategy focusing on learning from experience. In this paradigm, an agent makes decisions in an environment to maximize the final reward. The agent (model) parameters are updated with a trial and error approach, iteratively improving performance while gaining experience. Reinforcement learning finds extensive application in various robotic scenarios, including robot navigation, object grasping, and manipulation. However, it is rarely applied in perception tasks.

The paradigms above are categorized based on data supervision. Alternatively, they can be classified according to how data is available during training, leading to two distinct approaches: offline learning and online learning. In offline learning, the model is trained offline on data batches. This traditional paradigm often leads to superior performance and more stable training than online learning. However, it requires pre-acquired data before training, demands higher computational resources, and poses challenges for periodic fine-tuning in dynamic environments. In contrast, online learning updates model parameters on sequentially received data at each step. Each incoming sample is a new training sample, and previous data can be either discarded or stored partially in a queue to monitor training stability. Online learning is generally faster and requires fewer computational resources and storage than batch learning. Moreover, it facilitates dynamic adaptation to evolving environments, such as stock market prediction, weather forecasting, and navigation in dynamic settings. While



a) b)

Figure 1.5: Comparison between point clouds acquired a) with VelodyneHDL64E sensor in Karlsruhe, Germany and b) with a VelodyneHDL32E sensor in Boston, US.

online learning offers advantages, it has challenges. One notable issue is the potential for negative drift, making it challenging to prevent and detect when the model drifts toward trivial solutions.

In conclusion, selecting the most appropriate learning paradigm is connected to the available data, annotations, and the nature of the specific task. The decision among supervised, semi-supervised, and unsupervised learning is rooted in data and annotation availability. Reinforcement learning becomes particularly relevant when tasks necessitate learning through experience. Additionally, considerations like limited computational resources and a dynamic environment are crucial in choosing online learning over offline learning. This thesis primarily focuses on solving domain shift in 3D point cloud perception. This is done for 3D semantic segmentation and 3D object detection tasks by focusing on solutions following the unsupervised, semi-supervised, offline, and online paradigms.

1.5 Domain shift in point clouds

In 3D perception, the traditional pipeline involves training and evaluating a model using supervised data acquired *within* the same domain, *i.e.*, using the same sensor. However, a noticeable drop in test performance occurs when deploying a model trained on a different dataset. This drop is due

to domain shift between training (*source domain*) and test data (*target domain*). Domain shift arises when the source and target data are drawn from different data distributions. Its main components can be tentatively categorized into sensor-dependent and location-dependent.

Sensor-dependent components originate from the use of sensors with different specifications. For instance, consider the VelodyneHDL64E and VelodyneHLD32E sensors employed in the SemanticKITTI [9] and nuScenes [15] datasets, respectively. The VelodyneHDL64E (Fig. 1.5a) is equipped with 64 laser beams, measures a range of up to 120 meters, has a vertical resolution of 0.4°, and an azimuthal resolution ranging from 0.08° to 0.35°. Differently, the VelodyneHDL32E (Fig. 1.5b), equipped with 32 laser beams, measures a range of up to 100 meters, has a vertical resolution of 1.33°, and an azimuthal resolution ranging from 0.08° to 0.33°. A segmentation model trained on SemanticKITTI may show performance degradation on nuScenes due to the lower point density. Similarly, a detection model may predict inaccurate box sizes or positions due to sparser points.

Location-dependent components group the causes arising from the acquisition of point clouds in different geo-locations, environments, lighting and weather conditions, object sizes, and class distributions. For example, SemanticKITTI, acquired in Karlsruhe, Germany, differs from nuScenes, acquired in Boston, US. A segmentation model trained in the narrow streets of Karlsruhe may underperform when segmenting the broader streets in Boston. Similarly, variations in vehicle sizes between the two locations could affect the box predictions of a detection model.

In this section, we introduced the domain shift problem in 3D point clouds, categorizing its components into sensor and location-dependent. Our examples focused on domain shift between point clouds acquired in real locations, namely, real-to-real settings. Another important setting is

Table 1.1: Overview of the settings for handling domain shift in point clouds and publications in this thesis. The differences among these settings lie in the availability of source data during adaptation, the level of supervision on target domains, and the presence of target data during training. We report the type of supervision (Sup.) in each setting. Keys: available (✓), partial or sequential (∼), and unavailable (-).

Domain	Sup.	Setting 1: Semi-sup. target	Setting 2: Unsup. target	Setting 3: Unavail. source	Setting 4: Sequential target	Setting 5: Unknown target
Target	Labels	(∼)	-	-	-	-
	Data	✓	✓	✓	(∼)	-
Source	Labels	✓	✓	-	-	✓
	Data	✓	✓	-	-	✓
	Model	-	-	✓	✓	-
Publications		1	1	1	1	1

the synthetic-to-real (or synth-to-real). In this context, the training dataset is acquired with a sensor simulator, like CARLA [44], while the test dataset comprises real-world acquisitions. This thesis focuses on the domain shift problem arising in both the real-to-real and synthetic-to-real settings.

1.6 Handling domain shift in point clouds

In the previous section, we introduced the challenge of domain shift in real-world applications, particularly in the context of 3D point cloud perception. The increasing interest in autonomous driving has highlighted the critical need to address domain shift, especially concerning point cloud data. While the issue of domain shift has been extensively explored in camera-based perception through Domain Adaptation (DA) and Domain Generalization (DG) research, these approaches often fail or do not apply when dealing with the sparse and geometric nature of point clouds. As a result, this research direction still needs to be explored despite its practical implications.

This thesis focuses on solutions for the domain shift problem in the 3D point cloud perception field for autonomous driving scenarios. To address this, we introduce the concept of *Handling domain shift in point clouds*, studying five different settings commonly encountered when mitigating domain shift for 3D perception. These settings span from single or multiple supervised source domains to one or many target domains, each aiming to alleviate domain shift and enhance performance in the respective target domain. The differences among these settings lie in the availability of source data during adaptation, the level of supervision on target domains, and the presence of target data during training (Tab. 1.1).

1.6.1 Setting 1: Semi-supervised target

In this setting, we are given a supervised source dataset and a partially (semi-) supervised target dataset. The primary task, *e.g.*, 3D semantic segmentation, relies on guidance from the source supervision. The unlabeled target data plays a crucial role in addressing domain shift. Available target labels offer supplementary supervision, aiding the adaptation process and enhancing overall performance. This setting aligns with the Semi-Supervised Domain Adaptation (SSDA) task. It is the most straightforward among those studied in this thesis due to the optimistic assumption that a few target labels will be available. This introduces a unique and noise-free supervision from the target domain, simplifying adaptation. Despite that, this setting holds practical significance as it allows for exploring the improvements achievable with a negligible annotation effort.

1.6.2 Setting 2: Unsupervised target

This setting considers the most common situation where we are given a supervised source dataset and an unsupervised target dataset. Similar to

Setting 1, the primary perception task, *e.g.*, 3D semantic segmentation, is guided by source supervision, but the adaptation process has to rely only on unsupervised target data. This configuration poses additional challenges as domain shift is mitigated only under the guidance of the noisy and, often, unreliable target signal. This setting aligns with the Unsupervised Domain Adaptation (UDA) task, extensively explored in camera-based research. Feature alignment and adversarial training are commonly employed solutions in the camera-based literature. However, their direct application to point clouds is often impractical due to image-oriented assumptions and computational demands, which are often intractable in point clouds. Despite its complexity, this setting holds practical significance in real-world applications, where acquiring large-scale data is feasible, but annotation costs are often prohibitive.

1.6.3 Setting 3: Unavailable source

Despite being a common situation in many real-world applications, this setting is relatively under-studied in point cloud perception. Furthermore, it aligns with the Source-Free Unsupervised Domain Adaptation (SF-UDA) task. In this setting, we are given a pre-trained model over the source domain and an unlabeled target dataset. Moreover, source data are not available at adaptation time. The primary objective is to adapt the pre-trained model to the target domain while preventing degradation in the performance of the main perception task. This setting holds practical importance for two main reasons. Firstly, pre-trained models are easily accessible on the internet, yet their efficacy is affected by domain shift. Secondly, storing or sharing source data raises crucial challenges regarding memory constraints, security against adversarial attacks, and privacy issues.

1.6.4 Setting 4: Sequential target

This setting presents the novel task of Source-Free Online Unsupervised Domain Adaptation (SF-OUDA). Analogous to Setting 3, a pre-trained model on the source domain is available, and the source dataset is not accessible during adaptation. In contrast, the target dataset is a continuous stream of sequential point clouds. This scenario introduces two additional challenges compared to previous settings. Firstly, there is the risk of negative drift, where the network may drift towards representations harmful to the main task, resulting in the affected performance of the entire perception system. Secondly, the sequential nature of the target modality poses a particular challenge, requiring the network to rapidly adapt to new environments, scenarios, and dynamics. This configuration is one of the most challenging we study in this thesis as it emulates an autonomous vehicle to minimize domain shift while navigating a new, unseen target domain.

1.6.5 Setting 5: Unknown target

In this scenario, we have a single or multiple supervised source datasets but lack unlabeled data or information about future target domains. This aligns with the Domain Generalization (DG) task, and like the previous setting, we explore this for the first time in this thesis. The main challenges in DG involve the absence of target data and information about future domains. The model is trained only on the annotated source data, with generalization strategies to mitigate domain shift and enhance the model performance on unforeseen target domains. The target datasets are unavailable during training and are seen only at evaluation time. This is the most challenging yet practical setting investigated in this thesis. Enhancing the model generalization a priori, without knowledge of the target, holds central interest for any perception algorithm. Ideally, if a model

could generalize to any unseen domain, the need for adaptation would be precluded, resulting in significant time, cost, and computational resource savings.

1.7 Contributions

This thesis presents our contributions to 3D point cloud perception, specifically in semi-supervised domain adaptation, unsupervised domain adaptation, source-free unsupervised domain adaptation, source-free online adaptation, and domain generalization. Our **contributions** are listed as follows:

- **The first mixing-based method for unsupervised domain adaptation in 3D semantic segmentation** [Setting 2: Unsupervised target]: Our first contribution is the first mixing-based method for unsupervised domain adaptation in 3D LiDAR segmentation. Our method uses sample mixing, a two-branch symmetric pipeline, and mean-teacher update to process labeled source and unlabeled target data simultaneously. The proposed pipeline is simple, effective, and has the potential to improve and ease model adaptation in the context of autonomous driving.

- **A simple and effective extension to semi-supervised domain adaptation in 3D semantic segmentation** [Setting 1: Semi-supervised target]: Our second contribution is a new method for semi-supervised domain adaptation in 3D semantic segmentation. The new method enables our previous mixing-based pipeline to the semi-supervised settings. The additional target supervision is efficiently integrated into our double-branch pipeline: firstly, target labels are injected into the unsupervised target data, and secondly, the target

modality is mixed with a supervised source. This approach allows users to introduce a negligible amount of supervision in the adaptation process, further improving the adaptation performance. Therefore, our approach has the potential to be widely applied in real-world applications.

- **The first study and method for domain adaptation without source data in 3D object detection** [Setting 3: Unavailable source]: Our third contribution is the first method for unsupervised domain adaptation in 3D object detection without source data. Firstly, we introduce the problem of source-free unsupervised domain adaptation in the context of 3D object detection. Then, we propose the first approach in this setting using pseudo-annotations, reversible scale transformations, and motion coherency. This setting is considered a common situation for many applications. Therefore, it can be widely used in 3D perception pipelines starting from pre-trained models.

- **The first study and method for online adaptation in 3D semantic segmentation** [Setting 4: Sequential target]: Our fourth contribution is the study of source-free online unsupervised domain adaptation for 3D semantic segmentation. This setting is challenging as we aim to adapt a pre-trained source network to a target domain while navigating the new domain. We propose the first solution to this problem, using pseudo-labels, geometric propagation, and spatio-temporal feature consistency. The newly introduced setting is a common situation for many 3D perception systems. Therefore, our solution can be widely employed or adapted in real-world applications facing the same situation.

- **A new synthetic dataset for 3D semantic segmentation** [Set-

ting 4: Sequential target]: Our fifth contribution is a new synthetic dataset to facilitate the study of domain shift in 3D semantic segmentation between simulated and real LiDAR sensors. Our dataset includes two splits simulating two widely used real Velodyne sensors. Our new synthetic dataset can encourage the study of domain shift in 3D semantic segmentation. Moreover, it can be added to real-world data during training to improve final performance.

- **The first study and method for domain generalization in 3D semantic segmentation** [Setting 5: Unknown target]: Our sixth contribution is the first study on domain generalization in 3D semantic segmentation and introduces the first experimental setup for evaluating model performance across domains. Results reveal a significant performance gap between models trained on the source dataset and evaluated on a different target domain. To address this, we propose a novel method enhancing a sparse-convolutional 3D segmentation network by incorporating an additional dense 2D convolutional decoder. Our method enables robust feature learning transferable across domains and has the potential to be widely used in real-world applications. Moreover, this work aims to inspire cross-domain model development and evaluation within the research community.

In conclusion, our extensive exploration focused on handling the domain shift issue in 3D point cloud perception, covering the tasks of 3D semantic segmentation and 3D object detection. We have made several contributions in semi-supervised, unsupervised, source-free, online adaptation, and domain generalization. Our contributions will have a practical and lasting impact on the community, enabling the development of 3D perception systems with improved performance and robustness under the presence of domain shift.

1.8 Publications

This thesis is enclosed with a list of publications highlighting and supporting the contributions made in 3D point cloud perception, particularly focusing on domain adaptation and domain generalization. The following is a list of publications discussed in this thesis, ordered in chronological reverse order:

1. **C. Saltori**, F. Galasso, G. Fiameni, N. Sebe, F. Poiesi, E. Ricci, *Compositional Semantic Mix for Domain Adaptation in Point Cloud Segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI), 2023 [152]

2. **C. Saltori**, A. Osep, E. Ricci, L. Leal-Taixé, *Walking Your LiDOG: A Journey Through Multiple Domains for LiDAR Semantic Segmentation*, Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2023 [156]

3. **C. Saltori**, E. Krivosheev, S. Lathuiliére, N. Sebe, F. Galasso, G. Fiameni, E. Ricci, F. Poiesi, *GIPSO: Geometrically Informed Propagation for Online Adaptation in 3D LiDAR Segmentation*, European Conference on Computer Vision (ECCV), 2022 [154]

4. **C. Saltori**, F. Galasso, G. Fiameni, N. Sebe, E. Ricci, F. Poiesi, *CoSMix: Compositional Semantic Mix for Domain Adaptation in 3D LiDAR Segmentation*, European Conference on Computer Vision (ECCV), 2022 [153]

5. **C. Saltori**, S. Lathuiliére, N. Sebe, E. Ricci, F. Galasso, *SF-UDA^{3D}: Source-Free Unsupervised Domain Adaptation for LiDAR-Based 3D Object Detection*, IEEE International Conference on 3D Vision (3DV), 2020 [155]

In addition to the above publications, we report a list of publications that are not part of the material in support of this thesis. The following is a list of publications not discussed in this thesis, ordered in chronological reverse order:

1. L. Riz, **C. Saltori**, Y. Wang, E. Ricci, F. Poiesi, *Novel Class Discovery Meets Foundation Models for 3D Semantic Segmentation*, Under peer-review, 2023 [141]

2. G. Mei, **C. Saltori**, F. Poiesi, E. Ricci, Q. Wu, J. Zhang, *Unsupervised Point Cloud Representation Learning by Clustering and Neural Rendering*, Under peer-review, 2023 [116]

3. L. Riz, **C. Saltori**, E. Ricci, F. Poiesi, *Novel Class Discovery for 3D Point Cloud Semantic Segmentation*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023 [140]

4. G. Mei, F. Poiesi, **C. Saltori**, J. Zhang, E. Ricci, N. Sebe, *Overlap-Guided Gaussian Mixture Models for Point Cloud Registration*, Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2023 [115]

5. G. Mei, **C. Saltori**, F. Poiesi, J. Zhang, E. Ricci, N. Sebe, Q. Wu, *Data Augmentation-Free Unsupervised Learning for 3D Point Cloud Understanding*, British Machine Vision Conference (BMVC), 2022 [117]

6. **C. Saltori**, P. Rota, N. Sebe, J. Almeida, *Low-budget Label Query through Domain Alignment Enforcement*, Computer Vision and Image Understanding (CVIU), 2022 [157]

7. N. Bisagno, **C. Saltori**, B. Zhang, FGB De Natale, N. Conci, *Embedding Group and Obstacle Information in LSTM Networks for Human*

Trajectory Prediction in Crowded Scenes, Computer Vision and Image Understanding (CVIU), 2022 [11]

8. S. Roy, W. Menapace, S. Oei, B. Luijten, E. Fini, **C. Saltori**, and others, *Deep Learning for Classification and Localization of COVID-19 Markers in Point-of-care Lung Ultrasound*, IEEE Transactions on Medical Imaging (T-MI), 2021 [145]

9. F. Marra, **C. Saltori**, G. Boato, L. Verdoliva, *Incremental Learning for the Detection and Classification of GAN-Generated Images*, IEEE International Workshop on Information Forensics and Security (WIFS), 2019 [114]

10. **C. Saltori**, S. Roy, N. Sebe, G. Iacca, *Regularized Evolutionary Algorithm for Dynamic Neural Topology Search*, Image Analysis and Processing (ICIAP), 2019 [158]

Chapter 2

State of the art

2.1 Related work

2.1.1 Perception in point clouds

3D semantic segmentation

Semantic segmentation in point clouds can be performed at point-level [134], on range views [143], or on voxelized point clouds [227].

Point-level architectures process the input point cloud without intermediate representation processing. These architectures include PointNet [133], based on a multilayer perceptron series. PointNet++ [134] improves on PointNet by aggregating global and local point features at multiple scales. RandLA-Net [66] extends PoinNet++ [134] by embedding local spatial encoding, random sampling, and attentive pooling. KPConv [178] learns weights in the continuous space and introduces flexible and deformable convolutions for point cloud processing. More recent architectures follow advances from the image field and use transformer architectures. PointTransformer [234] is the first to introduce self-attention networks for 3D point cloud processing, solving a large set of tasks, including 3D point cloud semantic segmentation. PointTransformerV2 [205] improves on PointTransformer by grouping vector attention and introducing a partition-based

pooling method for more efficient sampling. In contrast to transformer-based architectures, PointNeXt [135] surpasses PointTransformer by improving PointNet++ with an improved training procedure, an inverted residual bottleneck, and separable MLPs. These methods are computationally inefficient when large-scale point clouds are processed. Computational efficiency can be improved by projecting 3D points on 2D representations [120] or using 3D quantization approaches [27]. The former includes 2D projection-based approaches that use 2D range maps and exploit standard 2D convolution filters [143] to segment these maps prior to a re-projection in the 3D space. RangeNet++ [120], SqueezeSeg networks [203, 204], 3D-MiniNet [2], PolarNet [232], and RangeFormer [83] are approaches that belong to this category. Although efficient, these approaches lose information when the input data are projected in 2D and re-projected in 3D. The latter includes 3D quantization-based approaches that transform the input point cloud into 3D discrete representations and that employ 3D convolutions [227] or 3D sparse convolutions [57, 27] to predict per-point classes. VoxelNet [227] maps input points into a voxel grid and processes the input voxel grid with 3D convolutions. SparseConv [57, 56] and MinkowskiNet [27] improves voxel processing and introduce sparse convolutions to improve efficiency. Cylinder3D [243] further improves voxel processing for LiDAR data by using cylindrical and asymmetrical 3D convolutions. SphereFormer [87] introduces sphere-transformer modules and aggregates information of dense and sparse LiDAR points with a radial window self-attention. While efficient, 3D quantization-based approaches are sensitive to the voxel size. A low voxel size can lead to increased computational demand, while a high voxel size may introduce geometric artifacts.

3D object detection

Object detection methods for 3D point clouds can be grouped into three main categories: point, projection, and point-voxel based methods.

Point-based methods use permutation invariant operators to implicitly capture local structures and fine-grained patterns without quantization, retaining the original geometry of raw points. F-PointNet [131] applies PointNet for 3D predictions on frustum points cropped based on 2D image predictions. PointRCNN [166] uses PointNet-like modules and ROI Pooling for distinguishing foreground points. A second canonical refinement step produces the final predictions. Similarly, VoteNet [132] uses PointNet++ on raw points and introduces Hough voting over proposals, which are further grouped and processed for the final predictions. 3DSSD [218] introduces a single-stage detector that uses a hybrid feature-distance based farthest point sampling strategy on raw points. Differently, Point-GNN [167] reasons on local neighborhood graphs constructed from the input points, on which each node iteratively summarizes semantic cues from neighbors. While point-based methods preserve the irregularity and locality of a point cloud, they have a higher latency than projection-based methods. This latency may be a challenge in real-time applications such as autonomous driving.

Projection methods involve projecting 3D points into a 2D representation for efficient convolution using CNNs. This is usually done with an intermediate 3D quantization step and by mapping the learned features into a BEV feature map. VoxelNet [227] is the pioneering in this category. Firstly, a voxelization step quantizes the input point cloud and learns voxel-wise features with a 3D CNN network. The learned representations are then projected into a 2D BEV-like feature map for the final 3D predictions. PointPillars [88] projects the input points into a BEV map

and employs a pillar network to generate fixed-size feature representations for each pillar. Differently, SECOND [214] uses sparse convolutions to process the point cloud data. Then, sparse features are projected into a BEV feature map, which is further processed for the final predictions. Voxel R-CNN [41] introduces a two-stage detector. Firstly, it applies voxelization for 3D feature extraction, then transforms the extracted features into BEV representation for generating 3D proposals. Finally, voxel features are extracted by a pooling strategy to refine proposals into the final predictions. SE-SSD [237] proposes a single-stage detector using the teacher-student learning paradigm. Teacher predictions supervise the student network, and shape-aware data augmentations are used to improve predictions further with orientation-aware loss. Projection-based methods have proven computational efficiency thanks to the BEV representation, reducing scale ambiguity and occlusion. However, they come with the main drawbacks of information loss of fine-grained patterns and increased computational overhead and memory usage in methods employing 3D CNNs.

Point-voxel methods are a recent research direction that integrates the advantage of point-based methods of preserving fine-grained patterns with the computational advantages of projection-based methods. For example, STD [219] proposes a two-stage 3D object detection framework. Firstly, raw points are fed to PointNet and PointNet++ backbones to generate accurate proposals. Feature pooling and voxelization are introduced for extracting proposal features. Then, proposals are improved in the second stage, yielding the final predictions. PV-RCNN [164] and the following PV-RCNN++ [165] integrate the effectiveness of 3D sparse convolutions and the flexible receptive fields of PointNet-like modules to learn more discriminative features. SA-SSD [62] introduces a single-stage detector interpolating 3D sparse convolution features for raw point clouds, on which an auxiliary network is applied to enable voxel features with structure-

aware capability. Differently, BADet [136] exploits long-range interactions iteratively among detection candidates, constructing local neighborhood graphs for a boundary-aware receptive field in a coarse-to-fine manner. Point-voxel methods aim to integrate the strengths of both projection and point-based approaches. However, they need to carefully address the trade-offs and potential drawbacks inherited from each approach.

2.1.2 Reducing domain shift in 3D semantic segmentation

Semi-supervised and unsupervised domain adaptation

Unlike domain adaptation for image-based tasks [180, 85], domain adaptation for point cloud segmentation still lacks a unified experimental setup to compare different approaches. Domain adaptation approaches for 3D semantic segmentation can be grouped into range-view, multi-modal, and 3D-focused methods (Tab. 2.1).

RV-based networks are affected by domain shift, which can be mitigated by using generative approaches [90, 235], feature alignment [90, 204, 235], and contrastive learning [142]. RayCast [90] tackles the real-to-real UDA problem by transferring the sensor pattern of the target domain to the source domain through ray casting. After training the deep network on the source data, a minimal-entropy correlation alignment loss is used to reduce domain shift [122]. SqueezeSegV2 [204] improves the SqueezeSeg [203] architecture and reduces domain shift in the synth-to-real setup by aligning source and target features with a geodesic correlation alignment [122]. ePointDA [235] addresses domain shift in the synth-to-real UDA setting at both input and feature levels. At the input level, a generative CycleGAN [242] is trained to simulate real sensor noise on synthetic source data. At the feature level, a higher-order momentum loss [21] is used to learn domain-agnostic features between source and target input data.

Gated [142] states that domain shift between source and target point clouds can be mitigated by solving the sparsity shift and introducing domain-specific parameters. Given an input pair of source and target RV images, they first solve the sparsity difference through self-supervised completion and applying a dropout mask. Then, residual gated adapters are added to the segmentation model to learn target-specific parameters. None of the RV-based methods tackle the semi-supervised scenario.

Multi-modal models are designed to process the information captured by multiple input sensors, *e.g.*, RGB cameras and LiDAR sensors are those typically used. Domain shift is tackled by enforcing prediction consistency among modalities and domains and using target (pseudo) labels. xMUDA [74] uses cross-modality and cross-domain consistency to learn a domain-agnostic model in the real-to-real UDA setting. Cross-modal consistency exploits source labels and target pseudo-labels to produce consistent multi-modal predictions in both domains. DeepCORAL feature alignment [171] enforces feature alignment between source and target domains. In [73], xMUDA is extended to SSDA settings, showing that cross-modal consistency is effective even in semi-supervised settings.

3D methods can process input point clouds with or without prior voxelization. UDA approaches for 3D segmentation include voxel-based architectures such as SparseConv [56] and MinkowskiNet [27]. Domain shift can be tackled by focusing on the sparsity problem [221, 208, 119] or by employing mixup strategies [153, 207]. Complete&Label [221] reduces the sparsity difference between real domains by formulating the domain adaptation problem as a point cloud completion problem. A self-supervised completion network is trained to make the sparse input point cloud denser. The pre-processed point clouds can be used as intermediate domains to lower the domain shift. PCT [208] disentangles domain shift between synthetic and real point clouds into appearance and sparsity. Then, PCT learns an

appearance translation module and a sparsity translation module. These modules are used for translating source data in the target modality. Translated data are then used with ST [246] and APE [81] in the UDA and SSDA settings, respectively. CoSMix [153] reduces domain shift in point cloud data by introducing a compositional semantic mixup strategy with a teacher-student learning scheme. The method obtains domain-invariant models/features by creating two new intermediate domains of composite point clouds: a mixed source and a mixed target. CoSMix-SSDA [152] further extends CoSMix to the SSDA settings by allowing target labels to be mixed in the source and target point clouds while improving adaptation. Similarly, PolarMix [207] crops, edits, and rotates point clouds of different domains at both scene and instance-level along the azimuthal angle in the 3D polar coordinate system. The mixed point clouds allow PolarMix to reduce domain shift in the UDA setting, but the method is not employed for SSDA.

Online domain adaptation

The previous domain adaptation approaches assume adaptation to be performed offline, with source and target data already acquired. Offline adaptation has been extensively studied either using source data [65, 108, 151, 245] or without using source data [225, 103, 155, 217]. Unlike these approaches, Online UDA (OUDA) can adapt a model to an unlabeled continuous target data stream through source domain supervision [190]. OUDA has been successfully applied in the context of image classification [121], image semantic segmentation [190], depth estimation [182, 233], robot manipulation [112], human mesh reconstruction [59], and occupancy mapping [181]. The assumption of unsupervised target input data can be relaxed and applied for online adaptation in image classification [95], video-object segmentation [189], and motion planning [176]. Recently, test-time adap-

tation methods have been applied to OUDA in classification using supervision from source data [173, 161, 192]. The only work exploring OUDA for 3D semantic segmentation is GIPSO [154]. GIPSO is the first pioneering work in this setting and introduces the problem of Source-Free OUDA (SF-OUDA). GIPSO reduces domain shift while navigating in a new, unseen scenario. It uses adaptive self-training and geometric-feature propagation to adapt a pre-trained source model online without requiring target labels, source data, and pre-recorded target data.

Domain generalization

Beyond domain adaptation, robustness to domain shift can be tackled via Domain Generalization (DG). In DG, the main goal is to learn domain-invariant features by training solely on the source domain data [238]. Due to the importance of learning robust representations, DG has been widely studied in the image domain [26, 238, 236, 125] in both, multi- [53, 96, 240] and single-source [17, 13, 236, 194, 91] setup. Image-based DG approaches reduce domain shift via domain alignment [101, 53], meta-learning [96, 7], data and style augmentations [163, 240, 236], ensembling techniques [45, 106], disentangled learning [78, 193], self-supervised learning [17, 13], regularization strategies [194, 68] and, reinforcement learning [68, 91]. Data augmentations have also been employed in DG for 3D object detection [93] to learn to detect vehicles damaged in car accidents. More recently, Robo3D [84] explores out-of-distribution generalization for 3D semantic segmentation and detection in severe weather conditions. In 3D semantic segmentation, DG has been recently studied by three concurrent works [80, 156, 159]. In [80], authors tackle single-domain generalization between real domains. Their core idea is to use random sub-sampling to enforce domain shift robustness. Firstly, feature consistency is enforced between the source domain and the augmented domain based on feature

affinity. Then, the correlation between class prototypes is constrained to be similar among point clouds. Differently, 3DLabelProp [159] accumulates the predictions of consecutive point clouds and propagates the predicted classes over time to obtain robustness against domain shift. LiDOG [156] explores DG for 3D semantic segmentation in single and multiple source domains. A 3D sparse convolutional network is enhanced with an auxiliary dense 2D convolutional decoder, trained on a dense BEV auxiliary to improve feature robustness against domain shift. Interestingly, BEV reasoning has also been used in cross-modal 3D semantic segmentation [97] and in sensor failure for 3D detection and map segmentation [50].

2.1.3 Reducing domain shift in 3D object detection

Solving domain shift in 3D object detection has received a growing interest in the last few years. The pioneering work of Wang et al. [196] studies domain adaptation in SSDA and weakly supervised settings. The former setting uses a few-shot set of annotated target samples introduced during source training. The latter requires weak supervision from object statistics, *i.e.*, average vehicle size, which is used to scale/re-scale annotated 3D bounding boxes or target (pseudo) annotations for reducing domain shift.

Domain shift can also be solved by bridging sensor specifications and point cloud densities between source and target domains [139, 200]. For example, LiDAR distillation [200] introduces an iterative UDA framework that progressively reduces the point cloud density from a denser source to a sparser target domain. Adversarial learning is another strategy for bridging sensor differences [37, 86, 93]. In [86], UDA for BEV vehicle detection is addressed using a generative network trained for translating source data into the target domain. Sensor oriented strategies are not used in the SSDA settings, except for [37] where both source and target supervision is required. However, generative methods proved additional computational

challenges that may not be trivially solved.

Self-training [155, 215, 109, 222] and mixing-based [25, 198] approaches provide a more lightweight alternative in reducing domain shift. Self-training has been widely studied in image adaptation [184, 206, 149, 210]. The main rationale is using knowledge distillation [64] from a source pre-trained network to pseudo-annotate target samples. In the context of 3D detection, SF-UDA^{3D} [155] addresses Source-Free UDA (SF-UDA) by following a self-training pipeline. To improve target pseudo-labels, authors introduce scale transformations and motion coherency. Scale transformations are also used in ST3D [215] and ST3D++ [216] during source pre-training. Then, ST3D [215] iteratively improves pseudo-labels with triplet box partition, memory ensemble and voting, and curriculum data augmentations. ST3D++ [216] improves over ST3D by denoising pseudo-labels with quality-aware partitioning. Other approaches introduce multi-level consistency in a teacher-student self-training pipeline [109] or a contrastive loss on BEV features during co-training [222].

Mixing-based approaches follow the advances in image segmentation [206, 124], and introduce mixup strategies [229, 230] for domain adaptation in 3D object detection [25, 198]. In [25], authors argue class balance to be one of the main domain shift causes. To solve this issue, they propose ReDB, a UDA method that involves a filtering mechanism for cleaning pseudo-labels and a mixing strategy for balancing classes in the target domain. Similarly, SSDA3D [198] reduces domain shift in the SSDA settings by introducing an inter- and intra-domain mixing strategy, targeting class balance and domain-agnostic features. Another research branch studies the domain shifts induced by adverse weather conditions. SPG [211] tackles UDA by recovering the lost foreground points with a semantic point generation network. Similarly, Hahnet et al. [61] introduce the first snowfall LiDAR simulator, enhancing robustness to snowfall weather conditions.

Table 2.1: Overview of existing methods for unsupervised (UDA), semi-supervised (SSDA), and source-free online unsupervised (SF-OUDA) adaptation in point cloud segmentation. For each approach, we report the sensor setup (Setup), the architecture (Input data type and Model), and the source and target datasets. Then, we classify the adaptation strategy into mixup based, adversarial learning based, alignment based, generative based, self-training based, and auxiliary task based. Furthermore, we report whether the implementation (Code) is publicly available.

Method	Setup	Architecture		Datasets		Settings			Adaptation						Code
		Input data	Model	Source	Target	UDA	SSDA	SF-OUDA	Mixup	Adv.	Align.	Gen.	Self-train.	Aux. task	
RayCast [90]	real-to-real	RV	RangeNet++ [120]	Sem.KITTI [9]	nuSc. [15]	✓					✓	✓			
ePointDA [235]	synth-to-real	RV	SqueezeSegV2 [204]	GTA-V [204]	KITTI [1] Sem.KITTI [9]	✓				✓	✓	✓			
SqueezeSegV2 [204]	synth-to-real	RV	SqueezeSegV2 [204]	GTA-V [204]	KITTI [1]	✓					✓				✓
Gated [142]	synth-to-real real-to-real	RV	SalsaNext [30]	GTA-V [204] nuSc. [15] KITTI [1]	KITTI [1] nuSc. [15]	✓						✓		✓	
xMUDA [74]	real-to-real	2D&3D	xMUDA [74]	nuSc. [15] KITTI [51] A2D2 [52]	nuSc.[15] KITTI [51] A2D2 [52]	✓					✓		✓		✓
Cross-modal [73]	real-to-real synth-to-real	2D&3D	xMUDA [74]	nuSc. [15] v.KITTI [14] A2D2 [52] Sem.KITTI [9] Waymo [172]	nuSc. [15] Sem.KITTI [9] Waymo [172]	✓	✓				✓		✓		✓
Complete&Label [221]	real-to-real	3D	SparseConv [57]	KITTI [1] Waymo [172] nuSc. [15]	KITTI [1] Waymo [172] nuSc. [15]	✓						✓			
PCT [208]	synth-to-real	3D	Minkowski [27]	SynLiDAR [208]	Sem.KITTI [9] Sem.POSS [126]	✓	✓					✓			✓
CoSMix-UDA [153]	synth-to-real	3D	Minkowski [27]	SynLiDAR [208]	Sem.KITTI [9] Sem.POSS [126]	✓			✓				✓		✓
CoSMix [152]	real-to-real synth-to-real	3D	Minkowski [27]	SynLiDAR [208] Sem.KITTI [9]	Sem.KITTI [9] Sem.POSS [126] nuSc. [15]	✓	✓		✓				✓		✓
PolarMix [207]	synth-to-real	3D	SPVCNN [175] Minkowski [27]	SynLiDAR [208]	Sem.KITTI [9] Sem.POSS [126]	✓			✓				✓		✓
SALUDA [119]	synth-to-real real-to-real	3D	Minkowski [27]	SynLiDAR [208] nuSc. [15]	SemKITTI [9] Sem.POSS [126] nuSc. [15]	✓							✓	✓	✓
GIPSO [154]	synth-to-real	3D	Minkowski [27]	SynLiDAR [208] Synth4D [154]	SemKITTI [9] nuSc. [15]			✓					✓		✓

Chapter 3

Setting 1&2: Semi-supervised and unsupervised target

In this chapter, we explore the first and second settings for handling domain shift in point clouds, specifically, "Setting 1: Semi-supervised target" and "Setting 2: Unsupervised target." Both settings aim to mitigate domain shift in 3D point cloud segmentation, with learning guided by the source domain supervision. However, they differ in terms of the level of supervision available for the target domain. This chapter introduces the first framework to address domain shift in both these settings. This framework consists of two versions of the same novel method. The first version is designed to handle domain shift with an unsupervised target, while the second version incorporates a single, one-shot, user-annotated target sample.

3.1 Compositional Semantic Mix for Domain Adaptation in Point Cloud Segmentation

LiDAR is currently the most suitable sensor for capturing accurate 3D measurements of an environment for autonomous driving [94] and robotic navigation [241]. Semantic scene understanding is a crucial component for

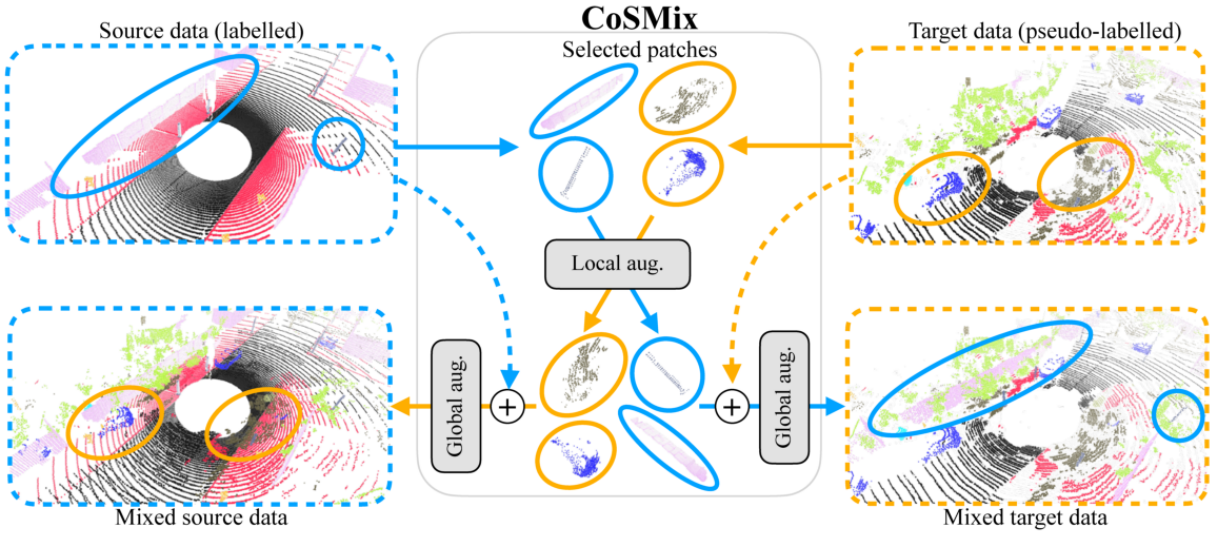


Figure 3.1: CoSMix applied to source and target data. Given (labeled) source and (pseudo-labeled) target data, we select domain-specific patches with semantic information to be mixed across domains. The resulting mixed data are a compositional semantic mix between the two domains, mixing source supervision in the target domain and target self-supervision (object and scene structure) in the source domain. Augmentations are applied at both local and global levels.

AI-based perception systems [100]. LiDAR measurements can be analyzed as 3D point clouds, with point cloud semantic segmentation used to assign a finite set of semantic labels to the 3D points [27]. To train accurate deep learning models, large-scale datasets with point-level annotations are necessary [9, 126, 15]. This involves a costly and labor-intensive data collection process, as point clouds need to be captured in the real world and manually annotated. An alternative is to use synthetic data, which can be conveniently generated with simulators [31]. However, deep neural networks suffer from domain shift when trained and tested on data from different domains [31]. Although simulators can reproduce the acquisition sensor with high fidelity, further research is still required to address such domain shift [208].

Data augmentation techniques based on combining samples and their

labels, such as Mixup [229] or CutMix [226], have been proposed to enhance deep network generalization. The underlying concept involves mixing samples to expand the training set and reduce overfitting. These methods were initially applied to image classification tasks and later adapted for domain adaptation and domain generalization in image recognition [210, 111]. Similar ideas have also been successfully extended to 2D semantic segmentation [184, 49]. While Unsupervised Domain Adaptation (UDA) for semantic segmentation in the image domain has been extensively studied [195, 49, 184, 245, 246], less attention has been devoted to developing adaptation techniques for point cloud segmentation. Point cloud UDA can be addressed in the input space [208, 235] with dropout rendering [235] or adversarial networks [208], or in the feature space through feature alignment [204]. A few studies have proposed exploiting sample mixing for point cloud data [123, 244], but they target different applications than UDA for semantic segmentation.

In this chapter, we introduce a novel UDA and SSDA framework for 3D LiDAR segmentation, named CoSMix, which can mitigate the domain shift by creating two new intermediate domains of composite point clouds obtained by applying a novel mixing strategy at the input level (Fig. 3.1). CoSMix is designed to mitigate the domain shift by mixing semantically-informed groups of points (patches) across domains. Specifically, we design a two-branch symmetric deep neural network pipeline that concurrently processes point clouds from a source domain (*e.g.*, synthetic or real) and point clouds from a target domain (*e.g.*, real or real but captured with a different sensor). Target point clouds can be either unlabeled or partially labeled if one wants to use CoSMix for UDA or SSDA, respectively. Each branch is domain specific, *i.e.*, the source branch is in charge of mixing a source point cloud with selected patches of a target point cloud, and vice versa for the target branch. We formulate mixing as a composition

operation, similar to the concatenation operation proposed in [123, 244], but unlike them, we leverage the semantic information to mix domains. Patches from the source point cloud are selected based on the semantic labels of their points. Patches from the target point cloud can be selected based on the predicted semantic pseudo-labels in the case of UDA and based on human annotations in the case of SSDA. We will show that only a handful of manually annotated points can significantly improve the domain adaptation performance. When patches are mixed across domains, we apply data augmentation both at local and global semantic levels to boost the efficacy of the mixing. An additional key difference between our method and [123, 244] is the teacher-student learning scheme that we implement to improve the accuracy of the pseudo-labels. We evaluate CoSMix on large-scale point cloud segmentation benchmarks, featuring both synthetic and real-world data in several directions, such as synthetic to real and real to real. Specifically, we use the following datasets: SynLiDAR [208], SemanticPOSS [126], SemanticKITTI [9], and nuScenes [15]. Our results show that CoSMix can reduce the domain shift, outperforming state-of-the-art methods in both UDA and SSDA settings. We perform detailed analyses of CoSMix and an ablation study of each component, highlighting its strengths and discussing its limitations.
Our main **contributions** can be summarised as follows:

- We introduce a novel scheme for mixing point clouds by leveraging semantic information and data augmentation.
- We show that the proposed mixing strategy can be used for reducing the domain shift and design CoSMix, the first adaptation framework for 3D LiDAR semantic segmentation based on point cloud mixing.
- We extend the original CoSMix to tackle the SSDA setup, allowing a user to input a few annotated points to significantly improve the

semantic segmentation performance on the target domain.

- We conduct extensive experiments on large-scale point cloud segmentation benchmarks, featuring synthetic and real-world data in several directions, such as synthetic to real and real to real. Our results demonstrate the effectiveness of CoSMix, which outperforms state-of-the-art methods.

3.1.1 Method

Preliminaries and definitions

CoSMix implements a teacher-student learning scheme that exploits the supervision from the source domain, the self-supervision from the target domain and, if available, the supervision from a few labeled target samples to improve the semantic segmentation on the target domain. Our method is trained on two different mixed point cloud sets. The first is the composition of the source point cloud with pseudo-labeled portions of points, or *patches*, of the unlabeled target point cloud. Target patches bring the target modality into the source domain, making the altered source domain more similar to the target domain. The second is the composition of the unlabeled target point cloud with randomly selected patches of the source point cloud. Source patches make the altered target domain more similar to the source domain, preventing overfitting from noisy pseudo-labels. If available, labeled points of the target point clouds can also be used in both the mixed point cloud sets. This target supervision can further reduce domain shift. The teacher-student learning scheme iteratively improves pseudo labels, progressively reducing the domain gap. Fig. 3.2 illustrates the block diagram of CoSMix.

Let $\mathcal{S} = \{(\mathcal{X}^s, \mathcal{Y}^s)\}$ be the source dataset that is composed of $N^s = |\mathcal{S}|$ labeled point clouds, where $\mathcal{X}^s$ is a point cloud and $\mathcal{Y}^s$ is its point-level

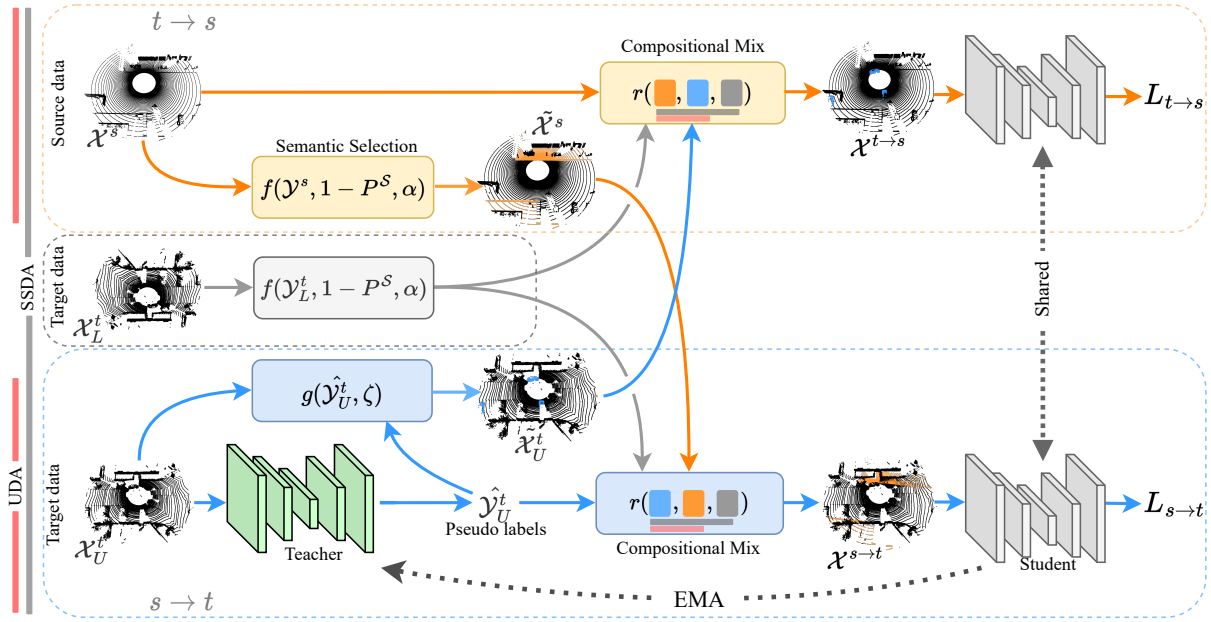


Figure 3.2: Block diagram of CoSMix detailing the UDA and SSDA settings. The UDA setting uses the top and bottom branches (red line). The SSDA setting also uses the middle branch in addition to those used in UDA (gray line). In the top branch, the input source point cloud $\mathcal{X}^s$ is mixed with the unsupervised target point cloud $\mathcal{X}_{\mathsf{U}}^t$ obtaining $\mathcal{X}^{t\to s}$. In the bottom branch, the input target point cloud $\mathcal{X}_{\mathsf{U}}^t$ is mixed with the source point cloud $\mathcal{X}^s$ obtaining $\mathcal{X}^{s\to t}$. In the SSDA setting, the labeled target data $\mathcal{X}_{\mathsf{L}}^t$ is mixed with the source point cloud $\mathcal{X}^s$ and with the unsupervised target point cloud $\mathcal{X}_{\mathsf{U}}^t$. A teacher-student learning architecture is used in both the UDA and SSDA settings to improve pseudo-label accuracy while adapting over the target domain.

labels, and $|.|$ is the cardinality of a set. Labels take values from a set of semantic classes $\mathcal{C} = \{c\}$, where c is a semantic class. Let $\mathcal{T}_{\mathsf{U}} = \{\mathcal{X}_{\mathsf{U}}^t\}$ be the unlabeled target dataset composed of $N_{\mathsf{U}}^t = |\mathcal{T}_{\mathsf{U}}|$ unlabeled point clouds. Let $\mathcal{T}_{\mathsf{L}} = \{(\mathcal{X}_{\mathsf{L}}^t, \mathcal{Y}_{\mathsf{L}}^t)\}$ be the semi-supervised set of $N_{\mathsf{L}}^t = |\mathcal{T}_{\mathsf{L}}|$ labeled target point clouds with $N_{\mathsf{L}}^t \ll N_{\mathsf{U}}^t$.

On the upper branch, the source point cloud $\mathcal{X}^s$ is mixed with selected patches of the target point cloud $\mathcal{X}_{\mathsf{U}}^t$ and selected patches of the supervised point cloud $\mathcal{X}_{\mathsf{L}}^t$ when available. The unlabeled target patches from $\mathcal{X}_{\mathsf{U}}^t$ are subsets of points that correspond to the most confident pseudo-labels $\hat{\mathcal{Y}}_{\mathsf{U}}^t$

that the teacher network produces during training. The supervised target patches are subsets of points that are randomly selected based on the class frequency distribution in the source training set. On the lower branch, the target point cloud $\mathcal{X}^t_{\mathsf{U}}$ is mixed with the selected patches of the source point cloud $\mathcal{X}^s$ and with the selected patches of $\mathcal{X}^t_{\mathsf{L}}$, if available. The source patches are subsets of points that are randomly selected based on their class frequency distribution in the training set.

We define the branch that mixes target point cloud patches to the source point cloud as $t \rightarrow s$ and the branch that does the vice versa as $s \rightarrow t$. Let $\mathcal{X}^{t \rightarrow s}$ be the mixed point cloud obtained from the upper branch, and $\mathcal{X}^{s \rightarrow t}$ be the mixed point cloud obtained from the lower branch. Lastly, let Φ_θ and $\Phi_{\theta'}$ be the student and teacher deep networks with learnable parameters θ and θ', respectively.

Semantic selection

To train the student networks with balanced data, we perform a selection of reliable and informative point cloud patches prior to mixing points and labels across domains. To select patches from the source point cloud, we use the class frequency distribution by counting the number of points of each semantic class within $\mathcal{S}$. Unlike DSP [49], which selects long-tail classes in advance, we exploit the source distribution and the semantic classes available to dynamically sample classes at each iteration.

Let $P^s_{\mathcal{Y}}$ be the class frequency distribution of $\mathcal{S}$. We create a function f that randomly selects a subset of classes at each iteration based on the labels $\tilde{\mathcal{Y}}^s \subset \mathcal{Y}^s$. f performs a weighted random sampling of α classes from the input point cloud by using $1 - P^s_{\mathcal{Y}}$ as the class weight for each class. α is a hyperparameter that regulates the ratio of selected classes for each point cloud. The output of f is a set of point-level labels belonging to the sampled classes, *i.e.*, $\tilde{\mathcal{Y}}^s$. The likelihood that f selects a class c is inversely

proportional to its class frequency in $\mathcal{S}$. Formally we have

$$\tilde{\mathcal{Y}}^s = f(\mathcal{Y}^s, 1 - P^s_{\mathcal{Y}}, \alpha). \tag{3.1}$$

Example: with $\alpha = 0.5$, the algorithm selects a number of patches corresponding to 50% of the available classes, *i.e.*, long-tailed classes are selected with a higher likelihood.

Let $\tilde{\mathcal{X}}^s$ be the set of points that correspond to $\tilde{\mathcal{Y}}^s$, and let $\tilde{\mathcal{X}}^s_c \subset \tilde{\mathcal{X}}^s$ be a patch (set of points) that belongs to class $c \in \mathcal{C}$. To select patches from the target point clouds, we apply the same set of operations but use the pseudo-labels produced by the teacher network based on their prediction confidence. Specifically, we define a function g that selects reliable pseudo-labels based on their confidence value. The selected pseudo-labels are defined as

$$\tilde{\mathcal{Y}}^t_{\mathsf{U}} = g(\Phi_{\theta'}(\mathcal{X}^t_{\mathsf{U}}), \zeta), \tag{3.2}$$

where $\Phi_{\theta'}$ is the teacher network, ζ is the confidence threshold used by the function g and $\tilde{\mathcal{Y}}^t_{\mathsf{U}} \subset \hat{\mathcal{Y}}^t_{\mathsf{U}}$.

Let $\tilde{\mathcal{X}}^t_{\mathsf{U}}$ be the set of points that correspond to $\tilde{\mathcal{Y}}^t_{\mathsf{U}}$.

In the case of target supervision, we apply f to the target labels $\mathcal{Y}^t_{\mathsf{L}}$ and randomly select target patches as

$$\tilde{\mathcal{Y}}^t_{\mathsf{L}} = f(\mathcal{Y}^t_{\mathsf{L}}, 1 - P^s_{\mathcal{Y}}, \mu), \tag{3.3}$$

where μ is a hyperparameter that regulates the ratio of selected classes for each point cloud similarly to α.

Compositional mix

Our compositional mixing module aims to create mixed point clouds based on the selected semantic patches. The compositional mix involves three consecutive operations: *local random augmentation*, where patches are augmented randomly and independently from each other; *concatenation*, where

the augmented patches are concatenated to the point cloud of the other domain to create the mixed point cloud; *global random augmentation*, where the mixed point cloud is randomly augmented. This module is applied twice, once for the $t \rightarrow s$ branch (top of Fig. 3.2), where target patches are mixed within the source point cloud, and once for the $s \rightarrow t$ branch (bottom of Fig. 3.2), where source patches are mixed within the target point cloud. Unlike Mix3D [123], our mixing strategy embeds data augmentation at the local level and global level.

Let δ be the indicator function that we define as

$$\delta(\mathcal{T}_\mathsf{L}) = \begin{cases} 1 & if \quad \mathcal{T}_\mathsf{L} \neq \emptyset \\ 0 & \text{otherwise,} \end{cases} \tag{3.4}$$

which indicates whether the supervised target set $\mathcal{T}_\mathsf{L}$ is empty or not. This can be interpreted as the user desire or need to use additional target supervision.

In the $s \rightarrow t$ branch, we apply the local random augmentation h to all the points $\tilde{\mathcal{X}}_c^s \subset \tilde{\mathcal{X}}^s$. We repeat this operation for all $c \in \tilde{\mathcal{Y}}^s$. Note that h is a local and random augmentation that produces a different result each time it is applied to a set of points. We define the result of this operation as

$$h(\tilde{\mathcal{X}}^s) = \left\{ h(\tilde{\mathcal{X}}_c^s), \forall c \in \tilde{\mathcal{Y}}^s \right\}. \tag{3.5}$$

If $\delta(\mathcal{T}_\mathsf{L}) = 1$ we can apply h also to $\tilde{\mathcal{X}}_\mathsf{L}^t$ and obtain

$$h(\tilde{\mathcal{X}}_\mathsf{L}^t) = \left\{ h(\tilde{\mathcal{X}}_\mathsf{L}^t), \forall c \in \tilde{\mathcal{Y}}_\mathsf{L}^t \right\}. \tag{3.6}$$

Then, we concatenate the locally augmented patches with the target point cloud X_L^t and we apply the global random augmentation, such as

$$\mathcal{X}^{s \rightarrow t} = \begin{cases} r(h(\tilde{\mathcal{X}}^s) \cup h(\tilde{\mathcal{X}}_\mathsf{L}^t) \cup \mathcal{X}_\mathsf{U}^t) & if \quad \delta(\mathcal{T}_\mathsf{L}) = 1, \\ \\ r(h(\tilde{\mathcal{X}}^s) \cup \mathcal{X}_\mathsf{U}^t) & \text{otherwise} \end{cases} \tag{3.7}$$

Their respective labels are concatenated accordingly as

$$\mathcal{Y}^{s\to t} = \begin{cases} r(h(\tilde{\mathcal{Y}}^s) \cup h(\tilde{\mathcal{Y}}^t_{\mathsf{L}}) \cup \mathcal{Y}^t_{\mathsf{U}}) & if \quad \delta(\mathcal{T}_{\mathsf{L}}) = 1, \\ \\ r(h(\tilde{\mathcal{Y}}^s) \cup \mathcal{Y}^t_{\mathsf{U}}) & \text{otherwise,} \end{cases} \tag{3.8}$$

where r is the global augmentation function.

The same operations of Eq. 3.7-3.8 are also performed in the $t \to s$ branch by mixing target patches within the source point cloud. Instead of using source labels, we use the teacher network to generate pseudo-labels from the target data. Additionally, we use target supervision if $\delta(\mathcal{T}_{\mathsf{L}}) = 1$. Then, we concatenate them with the labels of the source data. This results in $\mathcal{X}^{t\to s}$ and $\mathcal{Y}^{t\to s}$. Note that $\mathcal{T}_{\mathsf{L}}$ may be used without compositional mix and without double branched mixing. We implement h and r by using typical augmentation strategies for point clouds [27], *i.e.*, random rotation, scaling, and translation. We report additional information in the Implementation Details.

Network update

We leverage the teacher-student learning scheme to facilitate the transfer of knowledge acquired during the course of the training with mixed domains. We use the teacher network $\Phi_{\theta'}$ to produce target pseudo-labels $\hat{\mathcal{Y}}^t_{\mathsf{U}}$ for the student network Φ_θ, and train Φ_θ to segment target point clouds by using the mixed point clouds $\mathcal{X}^{s\to t}$ and $\mathcal{X}^{t\to s}$ based on their mixed labels and pseudo-labels.

At each batch iteration, we update the student parameters Φ_θ to minimize a total objective loss $\mathcal{L}_{tot}$ defined as

$$\mathcal{L}_{tot} = \mathcal{L}_{s\to t} + \mathcal{L}_{t\to s}, \tag{3.9}$$

where $\mathcal{L}_{s\to t}$ and $\mathcal{L}_{t\to s}$ are the $s \to t$ and $t \to s$ branch losses, respectively. Given $\mathcal{X}^{s\to t}$ and $\mathcal{Y}^{s\to t}$, we define the segmentation loss for the $s \to t$ branch as

$$\mathcal{L}_{s\to t} = \mathcal{L}_{seg}(\Phi_\theta(\mathcal{X}^{s\to t}), \mathcal{Y}^{s\to t}), \tag{3.10}$$

the objective of which is to minimize the segmentation error over $\mathcal{X}^{s\to t}$, thus learning to segment source patches in the target domain. Similarly, given $\mathcal{X}^{t\to s}$ and $\mathcal{Y}^{t\to s}$, we define the segmentation loss for the $t \to s$ branch as

$$\mathcal{L}_{t\to s} = \mathcal{L}_{seg}(\Phi_\theta(\mathcal{X}^{t\to s}), \mathcal{Y}^{t\to s}), \tag{3.11}$$

whose objective is to minimize the segmentation error over $\mathcal{X}^{t\to s}$ where target patches are composed with source data. We implement $\mathcal{L}_{seg}$ as the Dice segmentation loss [71], which we found effective for the segmentation of large-scale point clouds as it can cope with long-tail classes well.

Lastly, we update the teacher parameters θ' every γ iterations following the exponential moving average (EMA)[177] approach

$$\theta'_i = \beta\theta'_{i-1} + (1-\beta)\theta, \tag{3.12}$$

where i indicates the training iteration and β is a smoothing coefficient hyperparamenter.

3.1.2 Experiments

We evaluate our method in both synthetic-to-real and real-to-real UDA and SSDA settings. We use SynLiDAR [208] as synthetic dataset, and SemanticKITTI [9, 51, 1], SemanticPOSS [126], and nuScenes [15] as real-world datasets. We compare CoSMix with five state-of-the-art UDA methods: two general purpose adaptation methods (ADDA [185], Ent-Min [191]), one image segmentation method (ST [246]), and two point cloud segmentation methods (PCT [208], ST-PCT [208]). Then, we compare CoSMix

with five state-of-the-art SSDA methods: three general purpose adaptation methods (MMD [185], MME [150], APE [81]), and two point cloud segmentation methods (PCT [208], APE-PCT [208]). We refer to CoSMix-UDA and CoSMix-SSDA to indicate the version of CoSMix for UDA and SSDA, respectively. We use CoSMix to refer to our method in general otherwise. PCT, ST-PCT and APE-PCT are the only three state-of-the-art methods developed for 360° LiDAR point clouds and have only been applied for synthetic-to-real UDA and SSDA settings. We re-implemented the comparison methods and adapted them to the same backbone network as that of CoSMix. We refer to these methods as EntMin* [191], ST* [246], MME* [150], MMD* [185], *Source**, *Target** and *Fine-tuned**. Moreover, we extended EntMin [191] and ST [246] to the SSDA setting, and refer to them as EntMin-SSDA* and ST-SSDA*. For completeness, we also include the results of these methods as they are reported in [208].

Datasets and metrics

SynLiDAR [208] is a large-scale synthetic dataset that is created with the Unreal Engine [44]. It is composed of 198,396 annotated point clouds with 32 semantic classes. We use 19,840 point clouds for training and 1,976 point clouds for validation [208].

SemanticPOSS [126] is composed of 2,988 annotated real-world point clouds with 14 semantic classes. We use the sequence 03 for validation and the remaining sequences for training [126]. For the SSDA settings, we follow [208] and use the point cloud 172 of sequence 02 as the semi-supervised target set.

SemanticKITTI [9] is a large-scale segmentation dataset consisting of LiDAR acquisitions of the popular KITTI dataset [51, 1]. It is composed of 43,552 annotated real-world point clouds with more than 19 semantic classes. We use sequence 08 for validation and the remaining sequences

for training [9]. For the SSDA settings, we follow [208] and use the point cloud 848 from sequence 06 and the point cloud 940 from sequence 02 as semi-supervised target set.

nuScenes [15] is a large-scale segmentation dataset. It is composed of real-world 850 sequences (700 for training and 150 for validation), for a total of 34,000 annotated point clouds with 32 semantic classes. We use the official training and validation splits in all our experiments. For the SSDA settings, we follow the same selection protocol used in [208] and use the point cloud with token n015-2018-07-24-11-13-19+0800__LIDAR_TOP__1532402013 197655 as semi-supervised target set. We make source and target labels compatible across our datasets, *i.e.*, SynLiDAR → SemanticPOSS, SynLiDAR → SemanticKITTI and, Semantic KITTI → nuScenes. In SynLiDAR → SemanticPOSS and SynLiDAR → SemanticKITTI, we follow [208] and map labels into 14 segmentation classes and 19 segmentation classes, respectively. In SemanticKITTI → nuScenes, we map source and target labels into 7 common segmentation classes as in [154]. We evaluate the semantic segmentation performance before and after domain adaptation [208] by using the Intersection over the Union (IoU) [138] for each segmentation class and report the per-class IoU. We average the IoU over all the segmented classes and report the mean Intersection over the Union (mIoU).

Implementation details

We implemented CoSMix in PyTorch and ran our experiments on 4 NVIDIA A100 (40GB SXM4). We use MinkowskiNet as our point cloud segmentation network [27]. In particular, we use MinkUNet32 as in [208]. We pre-train our network on the source domain with Dice loss [71] starting from randomly initialized weights. In SSDA, we start from the pre-trained source model and finetune on both source and labeled target for two ad-

ditional epochs. The finetuned model is used as pre-trained model in semi-supervised settings. In UDA, we initialize student and teacher networks with the parameters obtained after pre-training. The pre-training and adaptation stages share the same hyperparameters. In both the pre-training and adaptation steps, we use Stochastic Gradient Descent with a learning rate of 0.001.

We set the value of α by examining the long-tailed classes present in the source domain during the adaptation process. Similarly, we set the parameter μ to the same value. We assign the values of α and μ based on our prior experience rather than optimizing these parameters through a systematic process. In the target semantic selection function g, we establish the value of ζ based on a qualitative assessment of a few target frames, with the aim of producing spatially compact predictions. This approach yields approximately 80% of pseudo-labeled points per scene.

On SynLiDAR $\rightarrow$ SemanticPOSS, we use a batch size of 12 and perform adaptation for 10 epochs. We set source and supervised target semantic selection (f) with $\alpha = 0.5$ and $\mu = 0.5$ while we set target semantic selection (g) with a confidence threshold $\zeta = 0.85$. On SynLiDAR $\rightarrow$ SemanticKITTI, we use a batch size of 16, adapting for 3 epochs. During source and supervised target semantic selection (f), we set $\alpha = 0.5$ and $\mu = 0.5$, while in target semantic selection (g), we use a confidence threshold of $\zeta = 0.90$. We use these last same hyperparameters also on SemanticKITTI $\rightarrow$ nuScenes and SynLiDAR $\rightarrow$ nuScenes.

Our local augmentations h and global augmentations r are based on data augmentation strategies that are typical in the LiDAR segmentation literature [27]. h involves rigid rotation around the z-axis, scaling along all the axes and random point downsampling. We remove xy rotation to produce co-planar and concentric mixed point clouds, and to preserve point ranges. For the same reason, we remove rigid translations. We

Table 3.1: Unsupervised adaptation results on SynLiDAR → SemanticPOSS. We denote our reproduced baselines and results with ⋆, *e.g.*, *Source*⋆. *Source*⋆ and *Target*⋆ correspond to the model trained on the source synthetic dataset (lower bound) and on the target real dataset (upper bound), respectively. Results are reported in terms of mean Intersection over the Union (mIoU).

Model	pers.	rider	car	trunk	plants	traf.	pole	garb.	buil.	cone.	fence	bike	grou.	mIoU
Source	3.7	25.1	12.0	10.8	53.4	0.0	19.4	12.9	49.1	3.1	20.3	0.0	59.6	20.7
Source⋆	21.7	20.1	9.7	3.4	56.8	4.8	24.1	6.1	39.9	0.3	15.3	5.3	73.4	21.6
Target⋆	61.8	54.7	33.0	19.3	73.9	26.7	30.9	11.0	71.3	32.5	44.6	43.2	78.5	44.7
ADDA [185]	27.5	35.1	18.8	12.4	53.4	2.8	27.0	12.2	64.7	1.3	6.3	6.8	55.3	24.9
Ent-Min [191]	24.2	32.2	21.4	18.9	61.0	2.5	36.3	8.3	56.7	3.1	5.3	4.8	57.1	25.5
Ent-Min⋆ [191]	24.8	28.0	13.4	4.1	59.6	2.0	23.3	5.8	47.0	0.0	16.1	5.8	71.6	23.2
ST [246]	23.5	31.8	22.0	18.9	63.2	1.9	**41.6**	13.5	58.2	1.0	9.1	6.8	60.3	27.1
ST⋆ [246]	47.7	42.6	24.4	13.8	62.5	3.3	36.1	23.5	50.9	18.8	14.6	4.0	68.9	31.6
PCT [208]	13.0	35.4	13.7	10.2	53.1	1.4	23.8	12.7	52.9	0.8	13.7	1.1	66.2	22.9
ST-PCT [208]	28.9	34.8	27.8	18.6	63.7	4.9	41.0	16.6	64.1	1.6	12.1	6.6	63.9	29.6
CoSMix-UDA	**55.8**	**51.4**	**36.2**	**23.5**	**71.3**	**22.5**	34.2	**28.9**	**66.2**	**20.4**	**24.9**	**10.6**	**78.7**	**40.4**

bound rotations between $[-\pi/2, \pi/2]$ and scaling between $[0.95, 1.05]$, and perform random downsampling for 50% of the patch points. r involves rigid rotation, translation, and scaling along all three axes. We set the parameters of r the same as those used in [27]. During the network update step, we update the teacher parameters θ'_i with $\beta = 0.99$. On SynLiDAR → SemanticPOSS, we set $\gamma = 1$ and do not perform parameter tuning. On SynLiDAR → SemanticKITTI, we increase γ to $\gamma = 500$ to obtain a stable teacher behavior, *i.e.*, stable source performance, high average confidence of pseudo-labels, and $\sim 80\%$ of pseudo-labeled points. We use these same hyperparameters also on SemanticKITTI → nuScenes, and SynLiDAR → nuScenes.

Quantitative comparisons for UDA

Synthetic-to-real. Tabs. 3.1&3.2 report the results in the UDA settings on SynLiDAR → SemanticPOSS, and on SynLiDAR → SemanticKITTI, respectively. The *Source*⋆ model is the lower bound of each scenario with 21.6 mIoU on SynLiDAR → SemanticPOSS and 23.8 mIoU on SynLiDAR

Table 3.2: Unsupervised adaptation results on SynLiDAR → SemanticKITTI. We denote our reproduced baselines and results with ⋆, *e.g.*, *Source*⋆. *Source*⋆ and *Target*⋆ correspond to the model trained on the source synthetic dataset (lower bound) and on the target real dataset (upper bound), respectively. Results are reported in terms of mean Intersection over the Union (mIoU).

Model	car	bi.cle	mt.cle	truck	oth-v.	pers.	b.clst	m.clst	road	park.	sidew.	oth-g.	build.	fence	veget.	trunk	terra.	pole	traff.	mIoU
Source	42.0	5.0	4.8	0.4	2.5	12.4	43.3	1.8	48.7	4.5	31.0	0.0	18.6	11.5	60.2	30.0	48.3	19.3	3.0	20.4
Source⋆	60.7	1.9	22.0	10.3	8.0	16.7	11.3	20.3	70.4	6.4	40.4	0.0	25.6	8.6	59.5	18.4	29.1	29.0	13.9	23.8
Target⋆	90.0	6.3	20.3	63.0	18.1	31.1	39.6	5.8	90.9	29.0	74.7	4.0	85.4	23.3	83.9	46.2	62.2	40.7	20.6	44.0
ADDA [185]	52.5	4.5	11.9	0.3	3.9	9.4	27.9	0.5	52.8	4.9	27.4	0.0	61.0	17.0	57.4	34.5	42.9	23.2	4.5	23.0
Ent-Min [191]	58.3	5.1	14.3	0.3	1.8	14.3	**44.5**	0.5	50.4	4.3	34.8	0.0	48.3	19.7	67.5	34.8	**52.0**	33.0	6.1	25.8
Ent-Min⋆ [191]	63.8	**8.5**	23.0	15.9	5.0	17.2	33.3	22.8	61.6	3.1	34.4	**0.2**	52.2	6.2	63.3	16.9	19.9	27.5	9.4	25.5
ST [246]	62.0	5.0	12.4	1.3	9.2	16.7	44.2	0.4	53.0	2.5	28.4	0.0	57.1	18.7	69.8	**35.0**	48.7	32.5	6.9	26.5
ST⋆ [246]	69.7	6.4	18.3	4.4	5.8	14.8	23.3	20.2	54.2	5.3	34.1	0.1	44.3	5.1	63.5	16.8	26.9	30.6	12.2	24.0
PCT [208]	53.4	5.4	7.4	0.8	10.9	12.0	43.2	0.3	50.8	3.7	29.4	0.0	48.0	10.4	68.2	33.1	40.0	29.5	6.9	23.9
ST-PCT [208]	70.8	7.3	13.1	1.9	8.4	12.6	44.0	0.6	56.4	4.5	31.8	0.0	**66.7**	**23.7**	**73.3**	34.6	48.4	**39.4**	11.7	28.9
CoSMix-UDA	**75.1**	6.8	**29.4**	**27.1**	**11.1**	**22.1**	25.0	**24.7**	**79.3**	**14.9**	**46.7**	0.1	53.4	13.0	67.7	31.4	32.1	37.9	**13.4**	**32.2**

→ SemanticKITTI. The *Target*⋆ model is the upper bound of each scenario with 44.7 mIoU on SynLiDAR → SemanticPOSS and 44.0 mIoU on SynLiDAR → SemanticKITTI. Note that *Source*⋆ models always outperform *Source*. This may be due to a better parameter choice that leads an improved generalization ability. In SynLiDAR → SemanticPOSS (Tab. 3.1), CoSMix-UDA outperforms the other methods on all the classes, except on *pole* where ST achieves a better result. On average, we achieve 40.4 mIoU, surpassing ST-PCT by +10.8 mIoU and improving over the *Source*⋆ of +18.8 mIoU. CoSMix-UDA improves also on difficult classes as *person*, *traffic-sign*, *cone*, and *bike*, whose performance are rather low before domain adaptation. ST⋆ and EntMin⋆ improve over *Source*⋆. ST⋆ improves over ST while EntMin⋆ achieves lower performance. Tab. 3.2 reports the results of SynLiDAR → SemanticKITTI. SemanticKITTI is challenging as the validation sequence includes a wide range of different scenarios with a large number of semantic classes. CoSMix-UDA improves all the classes when compared to Source⋆, except for *traffic-cone*. We believe this is due to the noise introduced by the pseudo labels on these classes and in related

Table 3.3: Unsupervised adaptation results on SemanticKITTI → nuScenes. We denote our reproduced baselines and results with ⋆, *e.g.*, *Source*⋆. *Source*⋆ and *Target*⋆ correspond to the model trained on the source real dataset (lower bound) and on the target real dataset (upper bound), respectively. Results are reported in terms of mean Intersection over the Union (mIoU).

Model	**car**	**pers.**	**road**	**side.**	**terr.**	**manm.**	**vege.**	**mIoU**
Source⋆	29.4	15.6	73.2	29.1	14.7	58.5	59.9	40.1
Target⋆	35.8	43.2	93.6	62.1	49.0	76.4	73.9	62.0
EntMin⋆[191]	**33.3**	12.6	**78.3**	**35.7**	18.4	63.1	62.4	43.4
ST⋆[246]	30.6	20.6	79.1	34.4	18.9	62.4	59.3	43.6
CoSMix-UDA	32.1	**26.3**	78.1	35.1	**20.2**	**66.4**	**65.2**	**46.2**

classes such as *road*. CoSMix-UDA improves on 10 out of 19 classes, with a large margin in the classes *car*, *motorcycle*, *truck*, *person*, *road*, *parking* and *sidewalk*. On average, we achieve state-of-the-art performance with a 32.2 mIoU, outperforming ST-PCT by +3.3 mIoU and improving over Source⋆ of about +8.4 mIoU.

Real-to-real. Tab. 3.3 reports the results on SemanticKITTI → nuScenes in the UDA setting. SemanticKITTI → nuScenes is a more challenging direction as source and target sensors are different, and nuScenes has rather sparse point clouds. *Source*⋆ and *Target*⋆ models achieve 40.1 mIoU and 62.0 mIoU, respectively. CoSMix-UDA outperforms the compared methods on 4 out of 7 classes, with the largest margin on the class *person*. On average, EntMin⋆ and ST⋆ achieve 43.4 mIoU and 43.6 mIoU, showing a limited improvement over *Source*⋆. CoSMix-UDA achieves the best results of 46.2 mIoU, outperforming all the compared methods.

Quantitative comparison for SSDA

Synthetic-to-real. Tabs. 3.4&3.5 report the results in the SSDA settings on SynLiDAR → SemanticPOSS, and on SynLiDAR → SemanticKITTI,

respectively. *Source** and *Target** models are the lower and upper bound of the UDA settings. The *Fine-tuned** model is obtained by fine-tuning *Source** with the semi-supervised target samples. It shows the highest possible bound without any adaptation approach. *Fine-tuned** always outperforms *Fine-tuned* from [208]. Similarly, the discrepancy between MMD* and MME*, and the results reported in [208] may be due to a different parameter choice. In SynLiDAR $\rightarrow$ SemanticPOSS (Tab. 3.4), CoSMix-SSDA outperforms all the comparison methods on all the classes, except on *plants*, *fence* and *bike* where MME and MME* achieve better results. On average, we reach 41.0 mIoU, outperforming APE-PCT by +9.8 mIoU and improving over *Source** by +19.4 and over *Fine-tuned** by +15.5. Compared to CoSMix-UDA, CoSMix-SSDA achieves a +0.6 mIoU, getting closer to the *Target* upper bound. In SynLiDAR $\rightarrow$ SemanticKITTI (Tab. 3.5), CoSMix-SSDA brings a significant improvement on 12 out 19, especially on *car*, *truck*, *motorcyclist*, *road*, *parking*, and *pole*. On average, CoSMix-SSDA achieves 34.3 mIoU, outperforming the best baseline APE-PCT by +7.3 mIoU and improving over *Source** of +10.5 mIoU and over *Fine-tuned** of +9.8 mIoU. Compared to our UDA pipeline, CoSMix-SSDA improves by +2.1 mIoU, showing that the additional target supervision is beneficial for further reducing the domain gap.

Real-to-real. Tab. 3.6 reports the results on SemanticKITTI $\rightarrow$ nuScenes in the SSDA settings. *Source** and *Target** models achieve 40.1 mIoU and 62.0 mIoU, respectively. The *Fine-tuned** model improves over *Source** and achieves 43.5 mIoU. CoSMix-SSDA achieves the best results on 3 out of 7 classes, with the largest margin on the class *pedestrian*. On average, MMD* is the best performing method among the comparison methods with 47.0 mIoU. CoSMix-SSDA achieves 48.9 mIoU, outperforms all the comparison methods, and further improves over CoSMix-UDA.

Table 3.4: Semi-supervised adaptation results on SynLiDAR → SemanticPOSS. We denote our reproduced baselines and results with ⋆, *e.g.*, *Source*⋆. *Source*⋆ and *Target*⋆ correspond to the model trained on the source synthetic dataset (lower bound) and on the target real dataset (upper bound), respectively. Results are reported in terms of mean Intersection over the Union (mIoU).

Model	**pers.**	**rider**	**car**	**trunk**	**plants**	**traf.**	**pole**	**garb.**	**buil.**	**cone.**	**fence**	**bike**	**grou.**	**mIoU**
Source	3.7	25.1	12.0	10.8	53.4	0.0	19.4	12.9	49.1	3.1	20.3	0.0	59.6	20.7
Source⋆	21.7	20.1	9.7	3.4	56.8	4.8	24.1	6.1	39.9	0.3	15.3	5.3	73.4	21.6
Fine-tuned	25.2	36.1	18.2	12.8	58.6	1.7	30.5	5.6	25.7	3.0	12.0	10.6	75.6	24.3
Fine-tuned⋆	25.2	27.2	21.6	9.6	60.4	0.7	16.2	10.8	44.5	12.0	24.1	2.5	76.7	25.5
Target⋆	61.8	54.7	33.0	19.3	73.9	26.7	30.9	11.0	71.3	32.5	44.6	43.2	78.5	44.7
MMD [185]	25.5	35.7	28.9	6.7	64.3	1.7	23.2	5.6	53.3	3.3	30.2	13.9	70.4	27.9
MMD⋆[185]	28.1	12.2	18.8	11.4	71.5	10.0	14.7	0.0	64.6	0.0	28.1	25.1	78.6	27.9
MME [150]	33.2	40.2	25.0	11.0	61.9	0.4	31.2	7.3	56.1	5.7	**37.1**	6.7	71.2	29.8
MME⋆[150]	35.8	16.1	21.4	7.9	**73.7**	7.9	24.2	1.5	67.6	0.0	32.8	**32.0**	77.0	30.6
APE [81]	34.3	40.1	21.5	16.3	62.6	0.9	31.1	2.3	55.9	13.3	34.3	9.6	71.6	30.3
EntMin-SSDA⋆	24.7	9.4	16.5	10.7	69.9	6.7	11.7	0.0	62.6	0.0	25.2	22.5	78.9	26.1
ST-SSDA⋆	40.1	24.3	22.5	7.9	70.2	13.4	21.7	1.4	66.9	0.1	34.7	**32.0**	78.1	31.8
PCT [208]	25.8	36.8	27.8	11.3	62.2	1.9	31.2	5.2	58.7	2.6	34.3	8.5	68.7	28.8
APE-PCT [208]	34.7	36.3	27.2	15.8	62.9	0.8	31.6	8.7	62.3	9.8	35.1	9.3	70.9	31.2
CoSMix-SSDA	**54.9**	**50.6**	**33.4**	**22.5**	73.0	**13.6**	**38.4**	**26.4**	**68.5**	**16.2**	29.6	27.4	**79.0**	**41.0**

Qualitative results

Fig. 3.3 shows some domain adaptation results on SynLiDAR → SemanticPOSS. Predictions of *Source*⋆ are often incorrect. CoSMix-UDA improves the segmentation results with more homogeneous regions and correctly assigned classes, and CoSMix-SSDA further improves the segmentation quality. Fig. 3.4 shows the results on SynLiDAR → SemanticKITTI that follows the same result pattern as in Fig. 3.3. Some classes (*e.g.*, *car*, *vegetation*, *pole*) greatly improve when CoSMix-SSDA is used. An evident increment of performance can be observed from *Source*⋆ to CoSMix-UDA to CoSMix-SSDA in both the studied domains. Although the limited amount of target supervision used in CoSMix-SSDA, these experiments show evidence of the benefits of our SSDA method.

Table 3.5: Semi-supervised adaptation results on SynLiDAR → SemanticKITTI. We denote our reproduced baselines and results with ⋆, *e.g.*, *Source*⋆. *Source*⋆ and *Target*⋆ correspond to the model trained on the source synthetic dataset (lower bound) and on the target real dataset (upper bound), respectively. Results are reported in terms of mean Intersection over the Union (mIoU).

Model	car	bi.cle	mt.cle	truck	oth-v.	pers.	b.clst	m.clst	road	park.	sidew.	oth-g.	build.	fence	veget.	trunk	terra.	pole	traff.	mIoU
Source	42.0	5.0	4.8	0.4	2.5	12.4	43.3	1.8	48.7	4.5	31.0	0.0	18.6	11.5	60.2	30.0	**48.3**	19.3	3.0	20.4
Source⋆	60.7	1.9	22.0	10.3	8.0	16.7	11.3	20.3	70.4	6.4	40.4	0.0	25.6	8.6	59.5	18.4	29.1	29.0	13.9	23.8
Fine-tuned	56.2	3.0	15.1	1.0	5.0	20.2	42.1	2.8	52.1	0.7	19.8	0.0	41.3	5.8	62.1	34.0	42.0	24.6	1.4	22.6
Fine-tuned⋆	61.8	2.8	21.7	10.8	4.2	14.5	18.0	15.9	65.6	6.4	40.2	0.0	34.2	7.0	60.6	22.1	39.8	32.2	8.4	24.5
Target⋆	90.0	6.3	20.3	63.0	18.1	31.1	39.6	5.8	90.9	29.0	74.7	4.0	85.4	23.3	83.9	46.2	62.2	40.7	20.6	44.0
MMD [185]	56.4	3.3	13.3	1.5	6.1	21.4	34.6	1.6	54.3	0.4	21.4	0.0	50.2	5.8	61.2	**37.0**	44.9	31.6	2.2	23.5
MMD⋆[185]	46.5	3.2	6.3	12.1	3.3	8.8	21.7	13.4	47.2	4.6	29.9	0.0	50.6	5.9	62.2	16.2	23.3	19.4	4.7	20.0
MME [150]	51.0	5.6	13.1	1.3	7.3	15.1	54.4	4.4	43.1	0.2	28.3	0.0	**60.7**	13.3	66.1	30.1	39.9	24.8	6.6	24.5
MME⋆[150]	28.7	0.2	1.0	1.8	0.9	2.0	1.6	3.5	53.6	1.8	31.1	0.0	40.6	7.2	57.6	12.1	26.7	14.4	0.1	15.0
APE [81]	58.6	6.2	16.6	3.1	11.3	14.2	35.8	3.7	61.5	1.7	30.3	0.0	54.7	**15.4**	64.6	20.0	**45.5**	23.9	9.1	25.1
EntMin-SSDA⋆	52.0	2.6	7.8	10.3	3.5	8.4	20.9	13.2	42.1	3.5	31.2	0.0	44.1	5.8	62.5	15.2	22.7	18.2	5.8	19.5
ST-SSDA⋆	60.0	2.8	10.6	14.6	5.0	10.4	19.2	20.8	63.9	4.5	35.7	0.1	43.4	7.1	62.2	13.3	26.9	24.8	10.4	22.9
PCT [208]	56.0	7.0	17.1	2.8	9.9	23.7	43.7	5.6	55.3	0.8	22.9	0.0	50.1	8.4	65.3	23.1	43.5	28.8	7.5	24.8
APE-PCT [208]	58.1	7.3	17.8	2.6	**13.9**	**24.7**	**46.5**	5.1	60.5	1.9	31.3	0.0	56.8	14.6	67.9	23.7	44.3	26.1	9.3	27.0
CoSMix-SSDA	**76.9**	**10.4**	**27.1**	**23.1**	13.4	24.0	21.7	**27.9**	**75.8**	**17.9**	**49.7**	**0.1**	60.3	14.7	**69.8**	36.8	40.9	**45.6**	**16.2**	**34.3**

3.1.3 Ablation study

We investigate the performance of CoSMix in its UDA and SSDA variants using the SynLiDAR → SemanticPOSS setup. The first three experiments are designed to study CoSMix in the UDA setting. Firstly, we analyze CoSMix-UDA components. Secondly, we compare our mixing approach with three recent point cloud mixing strategies, namely, Mix3D [123], PointCutMix [230] and PolarMix [207]. Then, we investigate the robustness of CoSMix to noisy pseudo-labels by changing the confidence threshold ζ and with different pre-trained models. In the last experiment, we analyze CoSMix in the SSDA setting, comparing our semi-supervised mixing approach with three variations of our approach.

Table 3.6: Semi-supervised adaptation results on SemanticKITTI → nuScenes. We denote our reproduced baselines and results with ⋆, *e.g.*, *Source*⋆. *Source*⋆ and *Target*⋆ correspond to the model trained on the source real dataset (lower bound) and on the target real dataset (upper bound), respectively. Results are reported in terms of mean Intersection over the Union (mIoU).

Model	**car**	**pers.**	**road**	**side.**	**terr.**	**manm.**	**vege.**	**mIoU**
Source⋆	29.4	15.6	73.2	29.1	14.7	58.5	59.9	40.1
Fine-tuned⋆	47.0	22.9	75.1	28.6	15.3	61.8	53.7	43.5
Target⋆	35.8	43.2	93.6	62.1	49.0	76.4	73.9	62.0
MMD⋆ [185]	38.3	14.1	83.2	32.4	33.9	63.7	63.2	47.0
MME⋆ [150]	41.0	9.5	**83.9**	31.7	32.9	63.3	57.8	45.7
EntMin-SSDA⋆	31.8	13.6	81.8	**35.5**	30.7	66.2	65.7	46.5
ST-SSDA⋆	44.9	13.9	71.9	22.6	**34.1**	**68.0**	**67.7**	46.2
CoSMix-SSDA	**45.3**	**26.9**	80.1	34.5	20.8	**68.0**	67.0	**48.9**

Method components

We analyze CoSMix by organizing its components into three groups: mixing strategies (*mix*), augmentations (*augs*), and other components (*others*). In the *mix* group, we assess the importance of the mixing strategies used in our compositional mix after semantic selection. In the *augs* group, we assess the importance of the local h and global r augmentations used in the compositional mix. In the *others* group, we assess the importance of the mean teacher update (β) and of the long-tail weighted sampling f. When the $t \rightarrow s$ branch is active, also the pseudo-label filtering g is utilized, while when f is not active, $\alpha = 0.5$ source classes are selected randomly. With different combinations of components, we obtain different versions of CoSMix, which we name CoSMix (a-h). The complete version of our method is named *Full*, where all the components are activated. The *Source*⋆ performance is also added as a reference for the lower bound. See Tab. 3.7 for the definition of these different versions.

When the $t \rightarrow s$ branch is used, CoSMix (a) achieves an initial 31.6

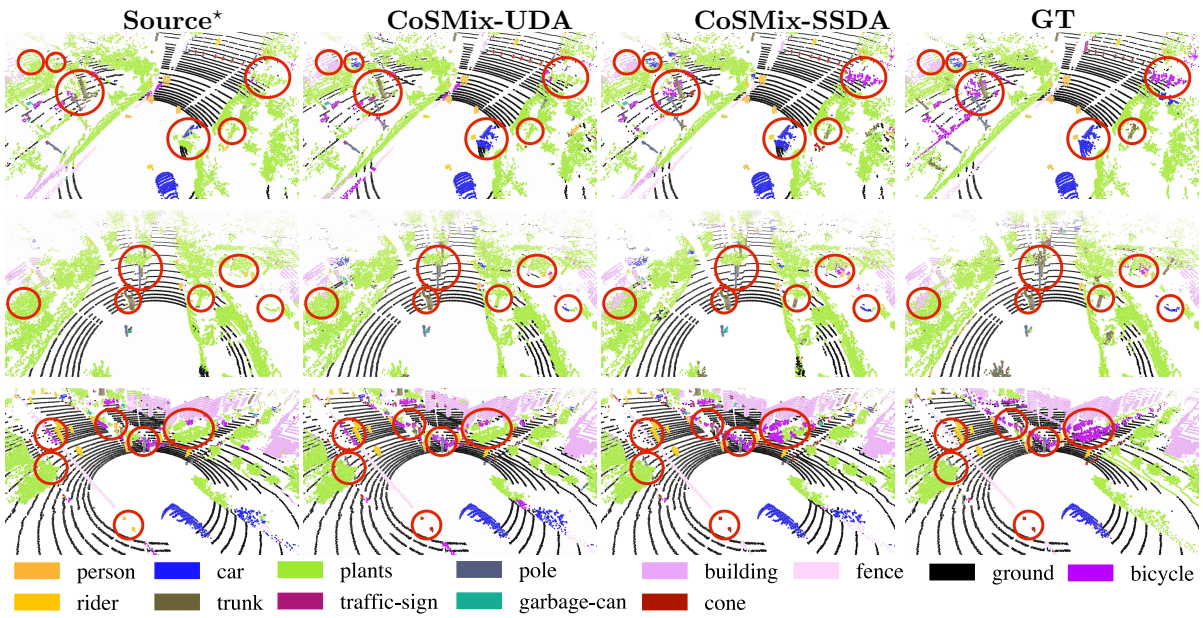


Figure 3.3: Results on SynLiDAR → SemanticPOSS. *Source*⋆ predictions are often wrong and mingled in the same region. After adaptation, CoSMix-UDA and CoSMix-SSDA improve segmentation with homogeneous predictions and correctly assigned classes. The red circles highlight regions with interesting results.

mIoU, showing that the $t \rightarrow s$ branch provides a significant adaptation contribution over the *Source*⋆. When we also use the $s \rightarrow t$ branch and the mean teacher β, CoSMix (b-d) further improve performance, achieving a 35.4 mIoU. By introducing local and global augmentations in CoSMix (e-h), we can improve performance up to 39.1 mIoU. The best performance of 40.4 mIoU is achieved with CoSMix Full where all the components are activated.

Point cloud mix

We compare CoSMix with Mix3D [123], PointCutMix [230], and PolarMix [207] to show the effectiveness of the different mixing designs. As per our knowledge, Mix3D and PolarMix are the only mixup strategies designed for 3D semantic segmentation, while PointCutMix and PolarMix are the only strategies for mixing portions of different point clouds. We implement

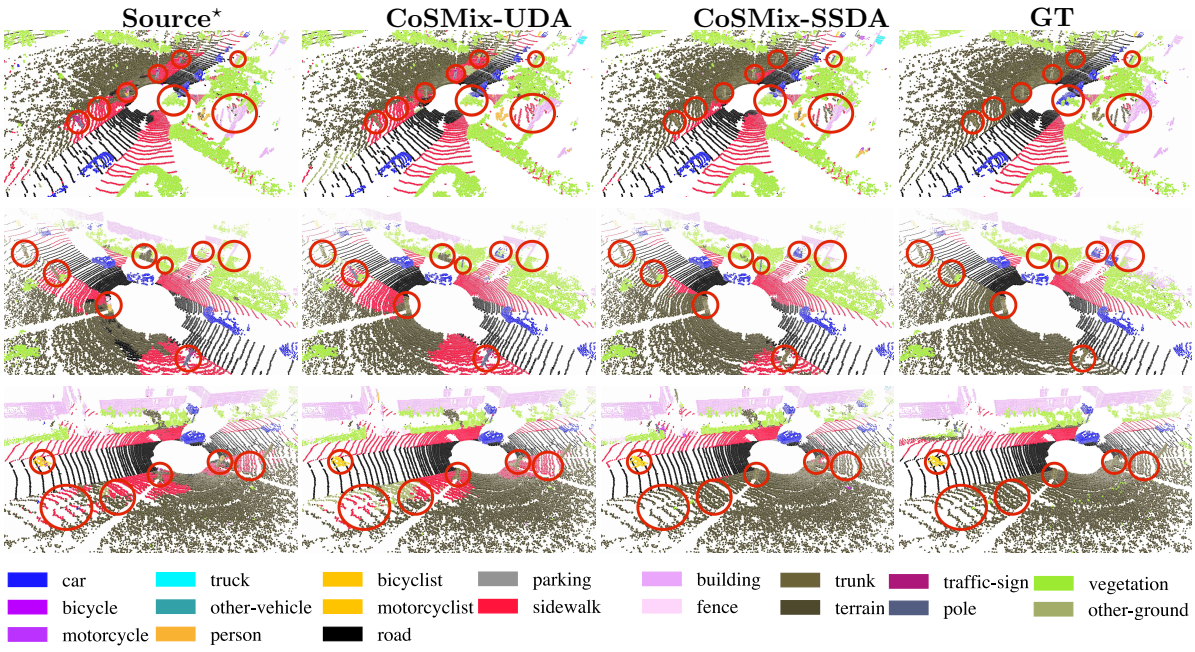


Figure 3.4: Results on SynLIDAR → SemanticKITTI. *Source*⋆ predictions are often wrong and mingled in the same region. After adaptation, CoSMix-UDA and CoSMix-SSDA improve segmentation with homogeneous predictions and correctly assigned classes. The red circles highlight regions with interesting results.

Mix3D and PointCutMix based on authors descriptions: we concatenate point clouds (random crops for PointCutMix) of the two domains, *i.e.*, $\mathcal{X}^s$ and $\mathcal{X}^t$, as well as their labels and pseudo-labels, *i.e.*, $\mathcal{Y}^s$ and $\hat{\mathcal{Y}}^t$, respectively. PolarMix [207] uses our same experimental settings and backbone therefore, we consider the results reported in their manuscript. We refer to these mixing strategies as Mix3D⋆, PointCutMix⋆, and PolarMix†. CoSMix *double* is our two-branch network with sample mixing. We deactivate the weighted sampling and the mean teacher update for a fair comparison. We keep local and global augmentations activated.

Fig. 3.6a shows that Mix3D⋆ outperforms the *Source*⋆ model, achieving 28.5 mIoU, followed by PolarMix† which achieves 30.4 mIoU. PointCutMix⋆ reaches 31.6 mIoU, outperforming the previous strategies. When we use the $t \rightarrow s$ branch alone, we can achieve 32.9 mIoU and when we use the $s \rightarrow t$ branch alone, CoSMix can further improve the results, achieving

Table 3.7: Ablation study of the CoSMix components: mixing strategy ($t \to s$ and $s \to t$), compositional mix augmentations (local h and global r), mean teacher update (β) and, weighted class selection in semantic selection (f). Each combination is named with a different version (a-h). *Source** performance are added as lower bound and highlighted in gray to facilitate the reading.

CoSMix version	mix		augs		others		
	$t \to s$	$s \to t$	h	r	β	f	**mIoU**
Source*	-	-	-	-	-	-	21.6
(a)	✓						31.6
(b)	✓				✓		31.9
(c)	✓	✓					35.0
(d)	✓	✓			✓		35.4
(e)	✓	✓	✓		✓		36.8
(f)	✓	✓		✓	✓		37.3
(g)	✓	✓	✓	✓	✓		39.0
(h)	✓	✓	✓	✓		✓	39.1
Full	✓	✓	✓	✓	✓	✓	**40.4**

34.8 mIoU. This shows that the supervision from the source to the target is effective for adaptation on the target domain. When we use the contribution from both branches simultaneously, CoSMix achieves the best result with 38.9 mIoU.

Robustness to noisy pseudo-labels

We investigate the robustness of CoSMix to increasingly noisier pseudo-labels. Firstly, we study the effect of different confidence thresholds ζ. Secondly, we evaluate different versions of pre-trained models that we use for generating pseudo-labels.

Confidence threshold. We study the importance of setting the correct confidence threshold ζ for pseudo-label distillation in g. We repeat the experiments with a confidence threshold from 0.65 to 0.95 and report the obtained adaptation performance in Fig. 3.6b. CoSMix is robust to noisy pseudo-

labels reaching a 40.2 mIoU with the low threshold of 0.65. The best adaptation performance of 40.4 mIoU is achieved with a confidence threshold of 0.85. By using a high confidence threshold of 0.95, performance is affected reaching 39.2 mIoU. With this configuration, too few pseudo-labels are selected to provide an effective contribution for the adaptation.

Model pre-training. We quantify the robustness of CoSMix and ST* [246] in response to pseudo-labels generated with different pre-trained models on SynLiDAR and tested on SemanticPOSS. In this experiment, we only utilize ST* as it is the sole method from those we benchmarked that is based on pseudo-labels. We denote the pre-trained model as P*. Fig. 3.5 displays its performance at different epochs: (a) 1, (b) 2, (c) 4, and (d) 9. Unlike CoSMix, ST* proves sensitive to pseudo-labels as it underperforms P* in three out of the four cases. A plausible explanation for this is that ST* refines the pre-trained model using filtered pseudo-labels during adaptation, depending on the quality of pseudo-labels. This dependency may cause ST* to drift during the adaptation process, thus impacting performance. Differently, CoSMix blends source and target (pseudo) labels, producing two intermediate domains with mixed labels. In the mixed point clouds, pseudo-labels are integrated with (noise-free) source labels ($t \rightarrow s$) or noise-free selections ($s \rightarrow t$), thus mitigating the negative effects of noisy and imprecise regions. Furthermore, applying a teacher-based approach allows us to rely on progressively more precise pseudo-labels, thereby minimizing undesirable drift effects.

Mixing target supervision

We compare CoSMix-SSDA to three alternative mixing strategies: naive, $sup \rightarrow s$ and $sup \rightarrow t$. In Fig. 3.6c, we name each strategy as CoSMix-SSDA (a-c). In version CoSMix-SSDA (a), we apply CoSMix-UDA without mixing $\mathcal{T}_\mathsf{L}$ in $\mathcal{X}^{s \rightarrow t}$ and $\mathcal{X}^{t \rightarrow s}$. Dice segmentation loss is applied sep-

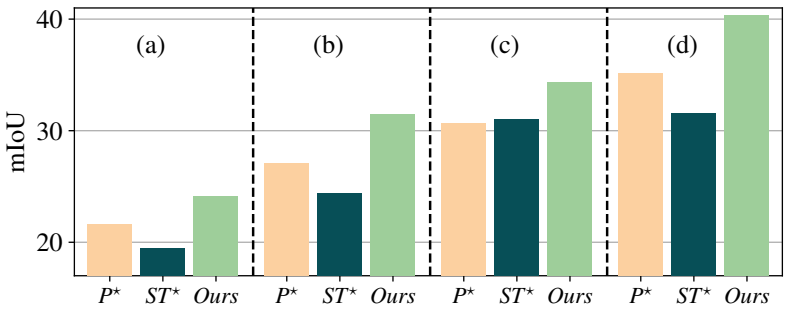


Figure 3.5: Adaptation results on SynLiDAR→SemanticPOSS with different pre-trained models. We compare the adaptation results of CoSMix (Ours) with ST⋆ starting from different initialization points (P⋆) indicated with (a-d).

arately on $\mathcal{T}_L$ and averaged with our total objective loss in Eq. 3.9. In the single branch mixing with source point clouds ($sup \rightarrow s$) and with target point clouds ($sup \rightarrow t$), versions (b-c), we apply only the upper or lower branch of CoSMix-SSDA, respectively. Full is our proposed double branched CoSMix-SSDA.

CoSMix-SSDA (a) approach reaches 33.7 mIoU, which shows that traditional training by using labeled target points as is leads to inferior performance than using our SSDA approach. Both the single branch mixing strategies achieve better performance with 38.9 mIoU and 40.5 mIoU for $sup \rightarrow s$ and $sup \rightarrow t$, respectively. Version (b) shows that the mixed target modality with noise-free annotations helps in reducing the domain shift. Version (c) suggests that the addition of target noise-free labels helps us achieve higher performance. However, both the single branch approaches are not sufficient to outperform the Full mixing strategy.

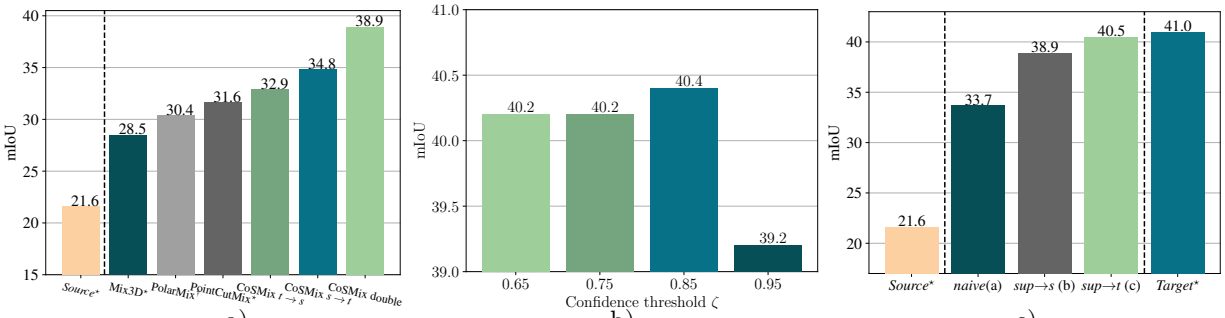


Figure 3.6: **a)** Comparison of the adaptation performance with different point cloud mix up strategies. Compared to the recent mixing strategies Mix3D [123], PointCutMix [230] and PolarMix [207], our mixing strategy and its variations achieve superior performance. **b)** Comparison of the adaptation performance on confidence threshold values. Adaptation results show that ζ should be set to achieve a trade-off between pseudo-label correctness and object completeness. **c)** Comparison of the SSDA performance with different mixing strategies: optimization without mix (naive), single branch mixing with source point clouds ($sup \rightarrow s$), single branch mixing with unsupervised target point clouds ($sup \rightarrow t$). Each variation is named with a different version (a-c). In all the experiments, *Source** and *Target** performance is the lower and upper bound.

Chapter 4

Setting 3: Unavailable source

This chapter explores the third setting, "Setting 3: Unavailable source". Similar to the previous, the main objective remains mitigating domain shift on an unlabeled target domain. However, the additional challenge is the absence of the source domain after model pre-training. This translates into addressing domain shift solely under the guidance of the target domain and under the learned task of a pre-trained model. Within this chapter, we introduce the first source-free adaptation method tailored for 3D object detection in LiDAR point clouds. This setting is of practical interest since pre-trained perception models are readily available online. However, acquiring source data often entails difficulties related to retrieval, storage, and concerns about security and privacy.

4.1 SF-UDA^{3D}: Source-free unsupervised domain adaptation for LiDAR-based 3D object detection

LiDAR is one of the key sensors for the longer-term autonomy of cars [72]. It is a native 3D sensor, which reads up to hundreds of meters, providing point clouds. Further to the proliferation of LiDAR companies, the efficacy of LiDAR is proven by the fact that LiDAR-only-based detectors

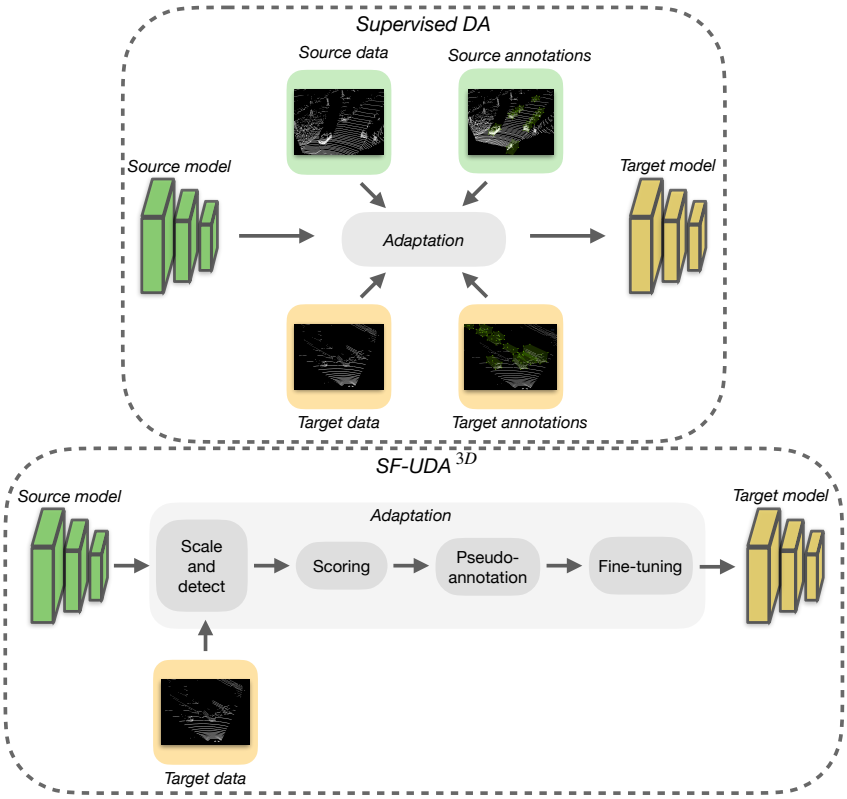


Figure 4.1: Existing supervised DA methods for LiDAR-based 3D detection [196] require both source and target data and annotations to adapt a pre-trained deep model to a target domain. Differently, leveraging on pseudo-annotations, reversible scale-transformations and motion coherency, SF-UDA^{3D} adapts a pre-trained source network by using only unlabeled target data.

are robust and accurate [214, 166, 34, 88] and provide state-of-the-art performance, currently held by PointRCNN [166]. LiDAR-based detectors, however, are prone to domain shift issues that may be more serious than for their RGB counterparts [196]. As in the RGB case, domain shift may be due to environmental changes (*e.g.*, data collected in different cities and weather conditions) or to appearance variations of specific objects (*e.g.*, car shapes and sizes may vary among different countries). Additionally, the performance of LiDAR-based models significantly depends on the density of the LiDAR point cloud, spatial resolution, and ranges.

To address this problem, in this chapter, we propose SF-UDA^{3D}, the

first Source-Free Unsupervised Domain Adaptation framework for LiDAR-based 3D detection. The proposed technique features the case where the 3D detector is applied, *e.g.*, in a different country, with differences in the local cars and roads, as well as to imagery acquired with different LiDAR sensors. Our approach is an *unsupervised* DA method because it does not require any annotation in the target domain, and it is *source-free* because we assume that only the 3D detector trained on source data is available, while we do not have access to the source annotations and data (see Fig. 4.1). Both aspects are novel. To the best of our knowledge, despite its practical relevance, the problem of building LiDAR-based 3D detectors which are robust to domain shift has only recently been addressed in [196]. However, the method proposed in [196] assumes the availability of both source and target domain data and annotations. In this section, we argue that this assumption is rarely satisfied in many real-world applications, where we may have only access to a pre-trained detector and it may be hard or even impossible to acquire target data annotations.

SF-UDA^{3D} considers the PointRCNN [166] architecture and is based on the pseudo-annotation of the target unlabelled dataset by means of reversible scale-transformations and motion coherency. In detail, SF-UDA^{3D} annotates the unlabelled target data at multiple scales by a PointRCNN model pre-trained on the unavailable source dataset. Then, it assesses the quality of the annotations by scoring the resulting detection tracks by means of an unsupervised coherency metric (Mean Volume Variation). Subsequently, it reverses the scale-transformations, aggregates the best detection labels by confidence, and finally fine-tunes Point RCNN. In spite of its simplicity, SF-UDA^{3D} surpasses in performance state-of-the-art domain adaptation methods for 3D object detection, which require target data annotations or annotation statistics [196]. Our algorithm also outperforms previous source-free general purpose unsupervised domain adap-

tation methods [102] by 30%. Overall, we fill in 66% of the gap between the source and target 3D detector when adapting from nuScenes to KITTI and 30% in the much harder case of KITTI to nuScenes.
To summarize, our main **contributions** are as follows:

- We propose to study a novel problem, *i.e.*, how to build LiDAR-based 3D detectors robust to domain shift when (i) we do not have access to source data and annotations and only a source pre-trained model is available, and (ii) no annotations are provided in the target domain, *i.e.*, we are in an unsupervised domain adaptation setting.
- We propose SF-UDA^{3D}, a novel approach for source-free unsupervised domain adaptation that empowers the state-of-the-art PointRCNN [166] architecture by means of pseudo-annotations, reversible scale-transformations, and motion coherency.
- We evaluate the proposed SF-UDA^{3D} against relevant approaches and we show that our method outperforms both previous source-free feature-based domain adaptation methods [102] and, notably, state-of-the-art adaptation approaches for LiDAR-based 3D detection, although they additionally use few-shot target annotations or target annotation statistics [196].

4.1.1 Method

In this section, we introduce a four-stage pipeline for adaptation, detailed in the next sections and illustrated in Fig. 4.2. Firstly, we introduce the preliminary definitions. Then, the following sections introduce our proposed approach, named SF-UDA^{3D}, which foresees four consecutive steps: scale and detect, scale scoring with temporal consistency, pseudo-annotation, and fine-tuning.

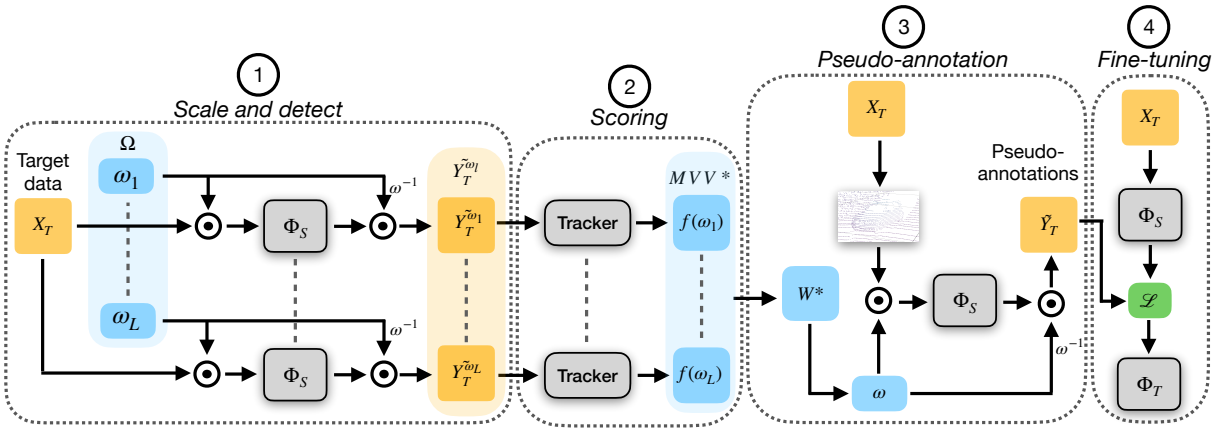


Figure 4.2: Overview of the SF-UDA^{3D} pipeline. Given a scaling solution space Ω, in the first step, detections over target sequences are obtained by scaling input data by ω and by re-scaling predictions by $1/\omega$. Next, time consistency of each sequence is used through a tracker to score each solution. During the third stage, scores are used to identify the best scaling interval W^*, and pseudo-annotations are obtained over multiple iterations with the same procedure of step one and are merged through NMS. Finally, we obtain the target adapted model $\Phi_\mathcal{T}$ by fine-tuning the source model over target data and pseudo-annotations.

Preliminaries and definitions

Given a 3D detection model Φ_S trained on a source dataset $\mathcal{X}_\mathcal{S} = \{x_S^n\}_{1<n<N}$ with source annotations $\mathcal{Y}_\mathcal{S} = \{y_S^n\}_{1<n<N}$, the goal of UDA in 3D detection is to obtain a target adapted 3D detector Φ_T by exploiting a target dataset $\mathcal{X}_\mathcal{T} = \{x_T^m\}_{1<m<M}$ without ground-truth annotations $\mathcal{Y}_\mathcal{T} = \{y_T^m\}_{1<m<M}$. In this work, we consider the challenging scenario of Source-Free Unsupervised Domain Adaptation (SF-UDA), where the source data $\mathcal{X}_\mathcal{S}$, the source annotations $\mathcal{Y}_\mathcal{S}$ and the target annotations $\mathcal{Y}_\mathcal{T}$ are not available at adaption time, namely, when training the target model Φ_T. When the LiDAR sensors differ across source and target domains, the geometries of the point clouds are different. Assume that the source and target point clouds are sampled by respective generating probability distributions $P(\mathcal{X})$, *i.e.*, $P(\mathcal{X}_\mathcal{S}) \neq P(\mathcal{X}_\mathcal{T})$. We illustrate in Sec. 4.1.2 that this is especially the

case for point could densities in the nuScenes [15] and KITTI [51] datasets considered in this work. When the source and target datasets are acquired across different domains, *e.g.*, countries, the ground-truth annotations also differ, *e.g.*, since the shapes of cars are also different. Assuming that annotations are sampled from generating probability distributions, these would differ, *i.e.*, $P(\mathcal{Y}_\mathcal{S}) \neq P(\mathcal{Y}_\mathcal{T})$. This is the case of the nuScenes [15] and KITTI [51] datasets, acquired respectively in USA/Singapore and Germany. The same discrepancy should be reflected in the 3D detector output spaces, trained to mimic the ground-truth annotations. In this work, we propose to align the annotation-scale distributions $P(\mathcal{Y}_\mathcal{S})$ and $P(\mathcal{Y}_\mathcal{T})$ by scale-transformation parameters, which we estimate by temporal coherency. Furthermore, we account for the misalignment of the point cloud distributions $P(\mathcal{X}_\mathcal{S})$ and $P(\mathcal{X}_\mathcal{T})$ by fine-tuning the source model Φ_S on the pseudo-annotated target point cloud.

Scale and detect

In the *Scale-and-detect* stage, we consider a set of L scaling parameters $\Omega = [\omega_1, ..., \omega_L]$ where each $\omega_l = (\omega_x, \omega_y, \omega_z) \in \mathbf{R}^{+3}$ parametrizes the scaling transformation along the 3D axes. To specify Ω, we use a regular grid over the intervals centered around 1: $[1-\epsilon, 1+\epsilon]^3$ with $\epsilon > 0$. For each ω_l, we generate a transformed version of the dataset $\mathcal{X}_\mathcal{T}^{\omega_l}$ by re-scaling each sample in $\mathcal{X}_\mathcal{T}$. Then, we employ the source object detector Φ_S on every sample of $\mathcal{X}_\mathcal{T}^{\omega_l}$ obtaining detections $\tilde{\mathcal{Y}}_\mathcal{T}^{\omega_l}$. Finally, to have detections $\tilde{\mathcal{Y}}_\mathcal{T}^{\omega_l}$ in the original target 3D space, we re-scale the detections by multiplying the position and dimension values by $(1/\omega_x, 1/\omega_y, 1/\omega_z)$. Besides, we note that this re-scaling step is required to obtain detections in the same 3D space and to allow a fair temporal consistency comparison. In all our experiments, we employ the PointRCNN detector [166] since it recently obtained state-of-the-art performance in object detection benchmarks.

Scale scoring with temporal consistency

To identify the quality of the estimated detections $\tilde{\mathcal{Y}}_{\mathcal{T}}^{\omega_l}$, we leverage the temporal consistency of detections between sequential frames (see Fig. 4.3). We propose to use a tracker and to evaluate the stability of its prediction to *score the detection quality*. More specifically, we run a state-of-the-art tracking-by-detection pipeline [202]. For a given sequence V, we assume to obtain J tracks. Considering the tracked object with the index $j < J$, its track can be defined as lists of T_j consecutive bounding-boxes $B_j = [b_{t_j}, ..., b_{t_j+T_j}]$, where t_j denotes the frame index where the object appears, and each b_t is the 3D bounding-box dimensions and locations at time t. Inspired by [209], we employ the Mean Volume Variation (MVV) between consecutive detections as scoring function:

$$MVV(V) = \frac{1}{J}\sum_{j=1}^{J}\sqrt{\frac{\sum_{t=t_j}^{t_j+T_j}(v_j^t - \bar{v_j})^2}{T_j - 1}} \tag{4.1}$$

where v_j^t is the bounding box volume of the j-th track at time t and $\bar{v_j}$ is the mean volume of the bounding boxes in B_j. Assuming an optimal detector and rigid objects, the intuition behind this scoring function is that the bounding-box volume would be constant. Therefore, a good detector would predict detections with a stable volume leading to a small MVV value. Importantly, we observed that tracks that last less than 5 frames may be false positives, and therefore, we propose to treat these tracks differently. More precisely, we introduce a penalty term H^* in the MVV score for every sequence without tracks longer than 5 frames. Our robust version of the MVV score can be written as:

$$MVV^*(V) = \begin{cases} MVV(V), & J \neq 0 \\ H^*, & \text{if no tracks} \end{cases} \tag{4.2}$$

Finally, as previously mentioned, we are interested in scoring each scaling solution over all the target training sequences, therefore we consider as the scoring function $f(\omega_i)$ for ω_i the mean of MVV^* over all the target sequences.

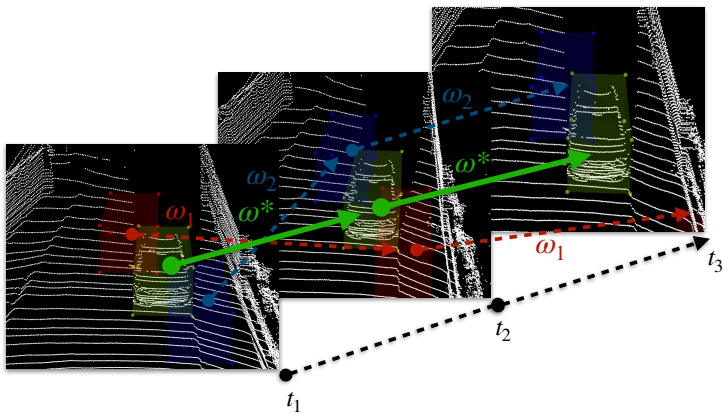


Figure 4.3: Given multiple possible scales ω, SF-UDA^{3D} selects the best ω^* as the one generating the most time consistent detections.

Pseudo-annotation and fine-tuning

Once each scaling parameter in Ω is scored, we proceed with *Pseudo-annotation*. In a first *single-scale* (*SS*) approach, we only consider the scaling parameter corresponding to the lowest MVV*, *i.e.*, the best scale, referred to as ω^* (see Fig. 4.3). In this approach, the detections $\tilde{\mathcal{Y}}_{\mathcal{T}}^{\omega*}$ at scale ω^* from the first stage of our pipeline are considered as pseudo-annotations. We confirm in Sec. 4.1.2 that the identified best scale provides the best single-scale results. We also propose a *multi-scale* (*MS*) approach, which combines the top-K best scaling parameters to improve the pseudo-annotations. We name Ω^* this set of best scales, and we use it to determine the best scaling interval $W^* = W_x^* \times W_y^* \times W_z^*$, as follows:

$$\forall a \in [x, y, z], W_a^* = [\min_{\omega \in \Omega^*}(\omega_a), \max_{\omega \in \Omega^*}(\omega_a)] \tag{4.3}$$

Given W^*, for every frame of the target dataset, we sample a random ω in W^*. The point cloud is scaled using ω. Then, we use the source model Φ_S to obtain 3D detections. Finally, we re-scale the predictions with $1/\omega$ similarly to the *scale-and-detect* stage, to obtain bounding boxes in the original point-cloud space.

Sampling scales is repeated several times, and the resulting 3D detections are collected and aggregated by non-maximal-suppression (NMS), yielding the final pseudo-labels $\tilde{\mathcal{Y}}_\mathcal{T}$. Sampling the scales multiple times is beneficial, possibly for two reasons: (i) cars within a point cloud occur at multiple scales, and multiple sampling may find more of them; (ii) PointRCNN randomly sub-samples the input point cloud to obtain a constant number of input points and having multiple scale samples ensures more robustness against this randomness. Note that PointRCNN provides detections with confidence scores, which we find beneficial to threshold increasingly from low to high values. In other words, at early steps, we use a low threshold to increase recall and include also low confidence detections. Later, we raise the threshold and only consider detections with a higher confidence. Finally, we merge pseudo-annotations from different steps through NMS and fine-tune the source model Φ_S by using $\tilde{\mathcal{Y}}_\mathcal{T}$ as annotations.

4.1.2 Experiments

In this section, we present the experimental evaluation of SF-UDA^{3D} on two modern large-scale benchmarks of KITTI [51, 1] and nuScenes [15] against state-of-the-art methods, albeit none of them is source-free and unsupervised. Then, we conduct a thorough ablation study of the proposed framework. In the following, we introduce benchmarks and metrics.

Table 4.1: Datasets overview. Each dataset is acquired using sensors with different resolutions and numbers of channels. While nuScenes uses the original maximum depth, KITTI provides pre-filtered LiDAR data with a maximum depth of 70 m.

Dataset	Samples	Max depth	Sensor	Channels	Resolution	Mean points	Classes
KITTI [51]	15k	70 m	HDL64E	64	$0.08° \times 0.4°$	16384	8
nuScenes [15]	34k	100 m	HDL32E	32	$0.08° \times 1.33°$	3808	23

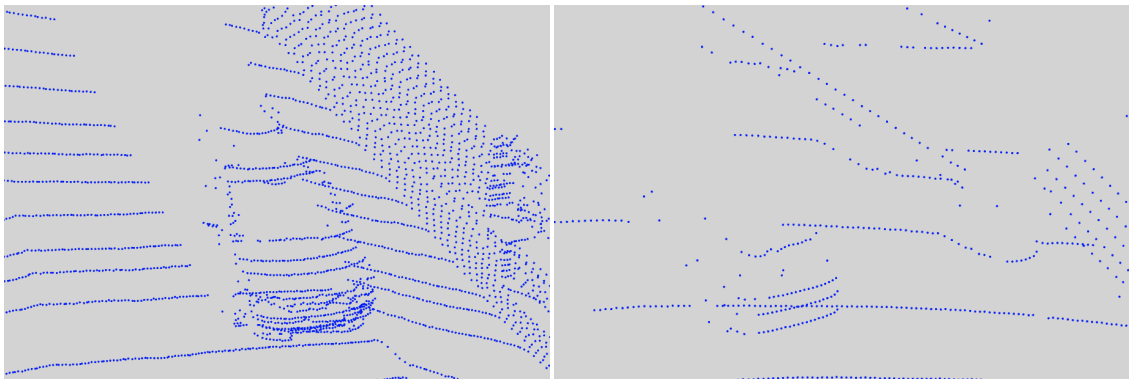
Datasets

The KITTI object detection benchmark dataset [51, 1] has been acquired in Karlsruhe, Germany, and is composed of 7481 training images, divided into 3712 training samples, 3769 validation samples, and 7518 test images. The nuScenes dataset [15], acquired in Boston (USA) and Singapore, is ~ 2.3 times larger, composed of 1000 driving sequences, for a total of 34149 images, divided into 28130 training samples and 6019 validation samples (which we treat as test samples), and it has both LiDAR scans and RGB images. The datasets differ under three main aspects:

- *Sensors.* Different sensors were used for data acquisition, as summarized in Tab. 4.1. These sensors sample points differently in terms of density (*i.e.*, number of points), temporal frequency, spatial resolutions and ranges. Fig. 4.4 illustrates example differences, which affect the performances of the 3D object detectors.

- *Environmental conditions.* Being acquired in diverse countries, the datasets depict objects of different shapes and sizes. Cars, which we target in this work, change much in these two aspects due to the differences between Germany and USA.

- *Dataset pre-processing.* The authors of the datasets made different choices for data collection, annotation, and filtering. For instance, in KITTI, only objects within $70m$ are annotated, while in nuScenes objects up to $100m$ have ground truth annotations. Another difference



a) KITTI [1] b) nuScenes [15]

Figure 4.4: Example of cars from the KITTI and nuScenes datasets.

is the range where objects are annotated. Indeed, in KITTI, only objects visible in the front camera view are annotated, while in nuScenes also objects are annotated in the entire 360° surrounding space. In all our experiments, we consider the points which are visible from the frontal RGB-camera viewpoint (*e.g.*, the CAM-FRONT in nuScenes and the rectified camera space for KITTI).

Metrics

For KITTI, we adopt the official metrics of [51, 1] and the classification into easy/moderate/hard detections, according to the visibility of the objects. In more detail, we report the Average Precision (AP) over the 3D Intersection over the union (IoU) with an IoU threshold of 0.7. The final average (Avg) AP is obtained by averaging over the three difficulty categories.

For nuScenes, we consider the official metrics [15]. In our experiments, we consider the center-based definition of AP as used in the official nuScenes benchmark [15] and report both the AP at each of the four different distance thresholds of $[0.5, 1.0, 2.0, 4.0]$ meters and the final average (Avg) over the four thresholds.

Implementation details

Our method is implemented in PyTorch, building upon the publicly-available PointRCNN 3D detector. For training and evaluating our method, we ran all our experiments on a DGX-1 server (8 Nvidia Tesla V-100 GPUs) and on a Lambda Blade server (8 NVidia Quadro RTX 6000 GPUs). The source training is performed with the ADAM optimization algorithm and one-cycle policy for 200 and 70 epochs for the RPN and RCNN, respectively, with a batch size of 64 and 32, respectively, and with a maximum learning rate of 0.02 as in [166]. Similarly, during the target fine-tuning we keep the same training setup with the only difference that we train the RPN for 100 epochs with a maximum learning rate of 0.002. During the scale search, we use a grid size parameter $\epsilon = 0.3$, and we employ a stride $s = 0.075$ along each axis. We thus obtain a solution space of dimension of $L = 125$. As for the tracking-by-detection module, we use the publicly-available code of [202] and change the tracker hyper-parameters as follows: both the initialization and death thresholds are set to 2 matched detections and missed detections, respectively. Finally, regarding the pseudo-annotation procedure, we annotate four times the target dataset for each confidence threshold in the list $[0.05, 0.1, 0.2, 0.3]$ and use an NMS IoU threshold of 0.1 to merge pseudo-annotations.

Comparison with the state-of-the-art

Our work is the first to tackle SF-UDA for 3D detection. For the sake of evaluation against state-of-the-art methods, we still compare with the following methods:

- *AdaBN* [102]: This is a state-of-the-art UDA method originally proposed for the classification tasks. We chose it because it is one of the few to operate in the source-free setting, differently from most previ-

ous approaches in UDA [31, 146, 18]. Here, we adapted AdaBN to the detection task by updating the batch normalization (BN) source-model mean and variance statistics of the PointRCNN features to the target validation set.

- *Few-Shot (FS) fine-tuning* [196]: This method proposes to adapt the detector to the target domain by fine-tuning on a set of 10 randomly sampled annotated target data. Following [196], we ran it 5 times and reported the average performance.

- *Output Transformation (OT)* [196]: This is a weakly-supervised DA technique. It uses average sizes of objects in both the source and target domain to transform the predictions of the source model at test time over to the target data.

We compare SF-UDA^{3D} in both the single-scale (SS) and top-K multi-scale (MS-K) approaches. We consider two settings: (i) nuScenes as source and KITTI as target (nuScenes→KITTI) and (ii) KITTI as source and nuScenes as target (KITTI→nuScenes). In order to be comparable with the literature on 3D object detection, we performed the experiments in the nuScenes→KITTI setting by using the same split as in [166] and in the KITTI→nuScenes setting by using the official nuScenes splits [15]. Note that in [196], results are evaluated on different data splits which are not the official ones.

The results on the nuScenes→KITTI task are reported in Tab. 4.2; those on the KITTI→nuScenes are shown in Tab. 4.3. In both cases, SF-UDA^{3D} outperforms both OS [196] and FS [196], although both of them use information from the target domain annotations. Analyzing the performance of different variations of our method, we observe that in the nuScenes→KITTI task (Tab. 4.2), the MS-3 version of SF-UDA^{3D} is the best-performing method, gaining a +0.08 Avg-AP over SF-UDA^{3D} (SS). Similarly, in the

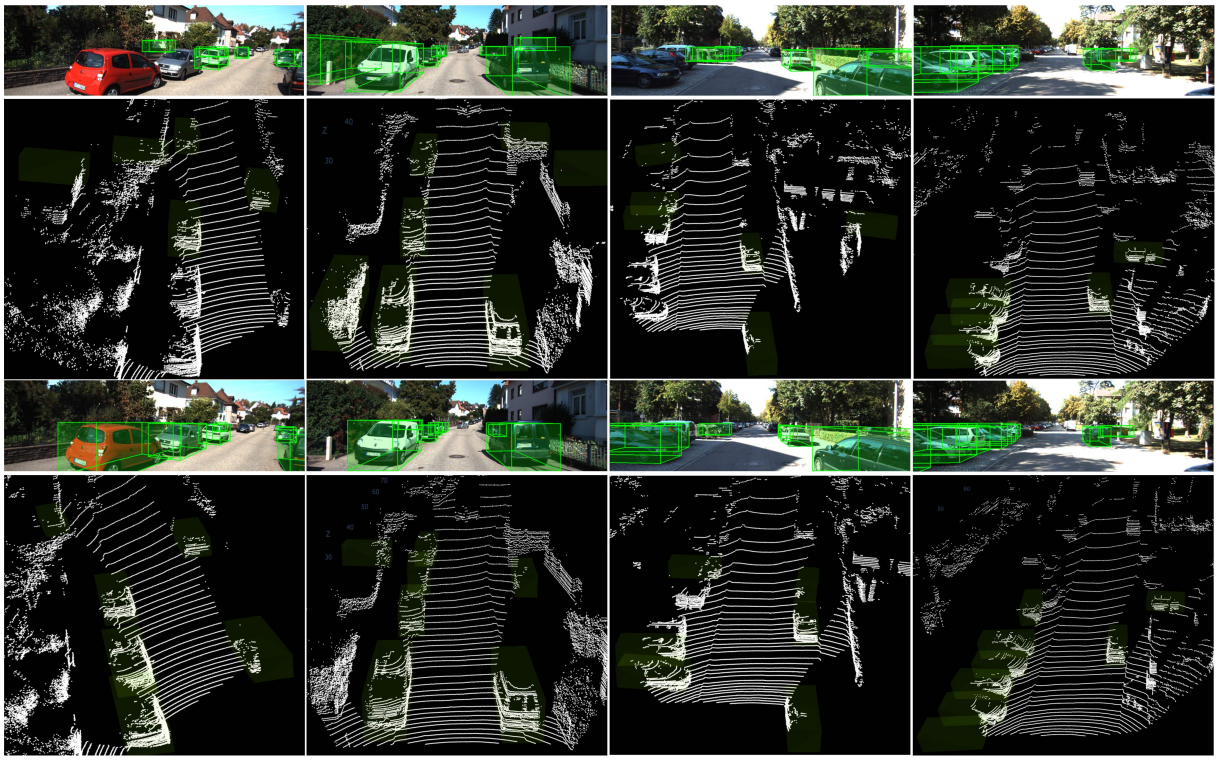


Figure 4.5: Before (top) and after (bottom) adaptation on nuScenes → KITTI. After adaptation with MS-3, performance improves and more objects are detected.

KITTI→nuScenes task (Tab. 4.3) SF-UDA^{3D} (MS-5) leads to an improvement of +0.019 Avg-AP compared to SF-UDA^{3D} (SS). These results confirm the effectiveness of our scale-based pseudo-annotation approach and the importance of the combination of the top-K solutions in the annotation procedure. Notably, AdaBN [102] is not effective on either the adaptation tasks. This may indicate that geometry-based methods are better suited than feature-based methods for 3D LiDAR-based detection adaptation and that features may need more sophisticated DA approaches. Additionally, to provide further insights on the results of Tab. 4.2 and Tab. 4.3, we also report and discuss the scaling parameters automatically selected by our method. Considering experiments on the nuScenes→KITTI task, SF-UDA^{3D} (SS) estimates the parameters $[1.30, 1.30, 1.15]$ along the axis X,Y and Z, respectively, showing that KITTI data have to be upscaled along

Table 4.2: Adaptation results: nuScenes→KITTI

Method	Easy	Moderate	Hard	Avg-AP
Source	0.273	0.196	0.188	0.219
AdaBN [102]	0.277	0.200	0.188	0.222
OT [196]	0.199	0.166	0.153	0.173
FS [196]	0.506	0.436	0.396	0.446
SF-UDA^{3D} (SS)	0.589	0.414	0.388	0.464
SF-UDA^{3D} (MS-3)	**0.688**	**0.498**	**0.450**	**0.545**
SF-UDA^{3D} (MS-5)	0.657	0.479	0.427	0.521
Target	0.873	0.769	0.760	0.801

each axis to better match with nuScenes data. Similarly, SF-UDA^{3D} (MS-3) provides an indication in the upscaling direction and selects the scale parameters within the intervals $[1.15, 1.30]$ along X, $[1.15, 1.30]$ along Y and $[1.15, 1.30]$ along Z. Conversely, for the more challenging KITTI→nuScenes setting, SF-UDA^{3D} (SS) computes the scaling parameters $[0.85, 0.70, 1.00]$, indicating that nuScenes objects point clouds should be downscaled along X and Y while keeping the original dimension along Z. Also the scale parameters adopted by the best performer SF-UDA^{3D} (MS-5) indicate downsampling for adaptation, by the ranges $[0.85, 1.15]$ along X, $[0.70, 0.85]$ along Y and $[0.85, 1.00]$ along Z. Here the model finds the wider range in X beneficial. Note that, in all of the above cases, the selected scales along the three axes are different. So further to scaling, SF-UDA^{3D} implicitly learns to change the aspect ratios for domain adaptation.

4.1.3 Ablation studies

Here we present the results of the ablation on each component of our method.

Scaling parameters. We investigate the importance of the scaling parameters in the single-scale (SS) setup of SF-UDA^{3D} during the pseudo - annotation phase. We consider the nuScenes→KITTI task and compare

Table 4.3: Adaptation results: KITTI→nuScenes

Method	AP-0.5	AP-1.0	AP-2.0	AP-4.0	Avg-AP
Source	0.143	0.208	0.224	0.234	0.202
AdaBN [102]	0.144	0.208	0.224	0.234	0.203
OT [196]	0.124	0.202	0.224	0.233	0.196
FS [196]	0.170	0.211	0.235	0.250	0.216
SF-UDA^{3D} (SS)	0.136	0.260	**0.290**	**0.308**	0.249
SF-UDA^{3D} (MS-3)	0.203	**0.266**	**0.290**	**0.308**	0.267
SF-UDA^{3D} (MS-5)	**0.211**	0.264	0.288	0.307	**0.268**
Target	0.370	0.422	0.440	0.455	0.422

our results with three baselines:

- *No score*: we remove from SF-UDA^{3D} the temporal-coherence tracking-based scores. This results in the random sampling of ω in the range $[0.7, 1.3]$ along each axis during the pseudo-annotation phase and quantifies the mere scale-augmentation.
- *No scale*: we remove from SF-UDA^{3D} the scale-transformations altogether. So the target dataset is simply pseudo-annotated by the source model *as is*.
- *Supervised scale (Sup-scale)*: we make the single-scale SF-UDA^{3D} weakly-supervised by providing it with the ground-truth scale differences between the target and source average object sizes.

Tab. 4.4 reports the results of our evaluation, comparing different scaling methods in terms of Avg-AP and reporting the selected scales. We observe from Tab 4.4 that *No scale* is a poor adaptation strategy, probably because there is a significant average scale difference between KITTI and nuScenes. Among the three ablation variants, *No score* is the worst, a bit surprisingly. While data augmentation is mostly a positive tool, here it shows that it is important to augment at the relevant scales, which we determine by tracking. *Sup-scale* respects the intuition and is slightly above our

Table 4.4: Ablation results: different scaling parameters.

Method	Selected Scale			Avg-AP
	X	Y	Z	
No scale	1.00	1.00	1.00	0.306
No score	[0.70, 1.30]	[0.70, 1.30]	[0.70, 1.30]	0.223
SS	1.30	1.30	1.15	0.464
Sup-scale	1.18	1.09	1.16	**0.499**

single-scale (SS) SF-UDA^{3D}, since knowing the ground truth is important. However, the improvement is marginal, which confirms that the estimation of scale-transformation parameters is effective.

Scale selection metric

We compare possible variants of unsupervised metrics to assess the quality of tracks (and thus to identify the most suitable scale transformations) on both the considered adaptation settings. In more detail, we take inspiration from the work of [209] on benchmarking super voxels and compare the applicable unsupervised metrics, namely the Time-Extension (TEX) and the Mean Volume Variation (MVV) – note that [209] introduces MSV, applicable to images, which we generalize to the point cloud volumes with MVV. TEX measures the temporal extent of tracking across frames, since intuitively a more stable tracking algorithm generates longer tracks. For MVV we compare the bare metric, as well as our proposed extension MVV* with the penalty below a minimum-length track.

Table 4.5: Ablation results: different scoring metrics.

Metric	Avg-AP	
	nuScenes→KITTI	KITTI→nuScenes
TEX [209]	0.030	0.229
MVV [209]	**0.488**	0.125
MVV*	0.464	**0.249**

Table 4.6: nuScenes→KITTI setting: quality of pseudo-annotations by using the best 3-scored scaling parameters. The results differ from Tab. 4.4 since here we measure the quality of pseudo-annotations before the fine-tuning step.

Scale	**Selected Scale**			**Avg-AP**
	X	**Y**	**Z**	
No scale	1.00	1.00	1.00	0.202
1^{st} best	1.30	1.30	1.15	0.370
2^{nd} best	1.15	1.30	1.30	0.394
3^{rd} best	1.30	1.15	1.15	0.369
Sup-scale	1.18	1.09	1.16	**0.435**

From Tab. 4.5, we see that TEX is not a suitable metric, though. In the realm of point clouds, longer tracks may correspond to wrong matches over time. This table shows that MVV is slightly better for the adaptation nuScenes→KITTI, but MVV* greatly outperforms it for the adaptation KITTI→nuScenes. So the penalty terms play an important role in the scoring over the noisier and more difficult nuScenes dataset.

Assessing pseudo-annotations by scaling just

We target to measure the quality of the pseudo-labels due to scaling transformations and to investigate why the combination of multiple scales is superior to the single-scale (*e.g.*, MS-3 Vs. SS) pseudo-labelling. To this goal, we focus on estimating the best scales by temporal coherency. Then we consider each of the three best (1st, 2nd and 3rd best) and re-scale the target point cloud according to it. Finally, we detect with the original source model Φ_S, without fine-tuning. Compared to the full pipeline of Fig. 4.2, this study excludes step 4 of the pipeline. The results are reported in Tab. 4.6. Re-scaling the point cloud according to each of the three best scales improves the performance of the source model considerably. In fact, the source model passes from a performance of 0.202 Avg-AP on the non-rescaled point-cloud (*No scale*) to 0.370 Avg-AP in the case of the 1st

best, which is 83% better. Also, re-scaling according to the ground-truth annotation statistics (*Sup-scale* entry in the Table) is understandably better. Finally, note that the three best scales resize the point cloud in the same direction, by approximately similar upscaling factors, but with different aspect ratios. This may account for the better performance of MS-3 Vs. SS.

Pseudo-annotations

We report qualitative examples of the pseudo-annotations obtained with SF-UDA^{3D} and used in the last fine-tuning step. In Fig. 4.6, we report the pseudo-annotations of SF-UDA^{3D} for the KITTI dataset while in Fig. 4.7 we report the pseudo-annotations for the nuScenes dataset. As visible, KITTI is an easier adaptation direction allowing to more *stable* pseudo-annotations with a small presence of false positives. Differently, the nuScenes dataset is an harder adaptation direction, with an high density difference and more difficult detections. Nevertheless, SF-UDA^{3D} is capable to obtain stable pseudo-annotations, with an higher presence of false positives compared to KITTI, which allow to an improvement after the fine-tuning step. Interestingly, we found that false positives do not affect models performance.

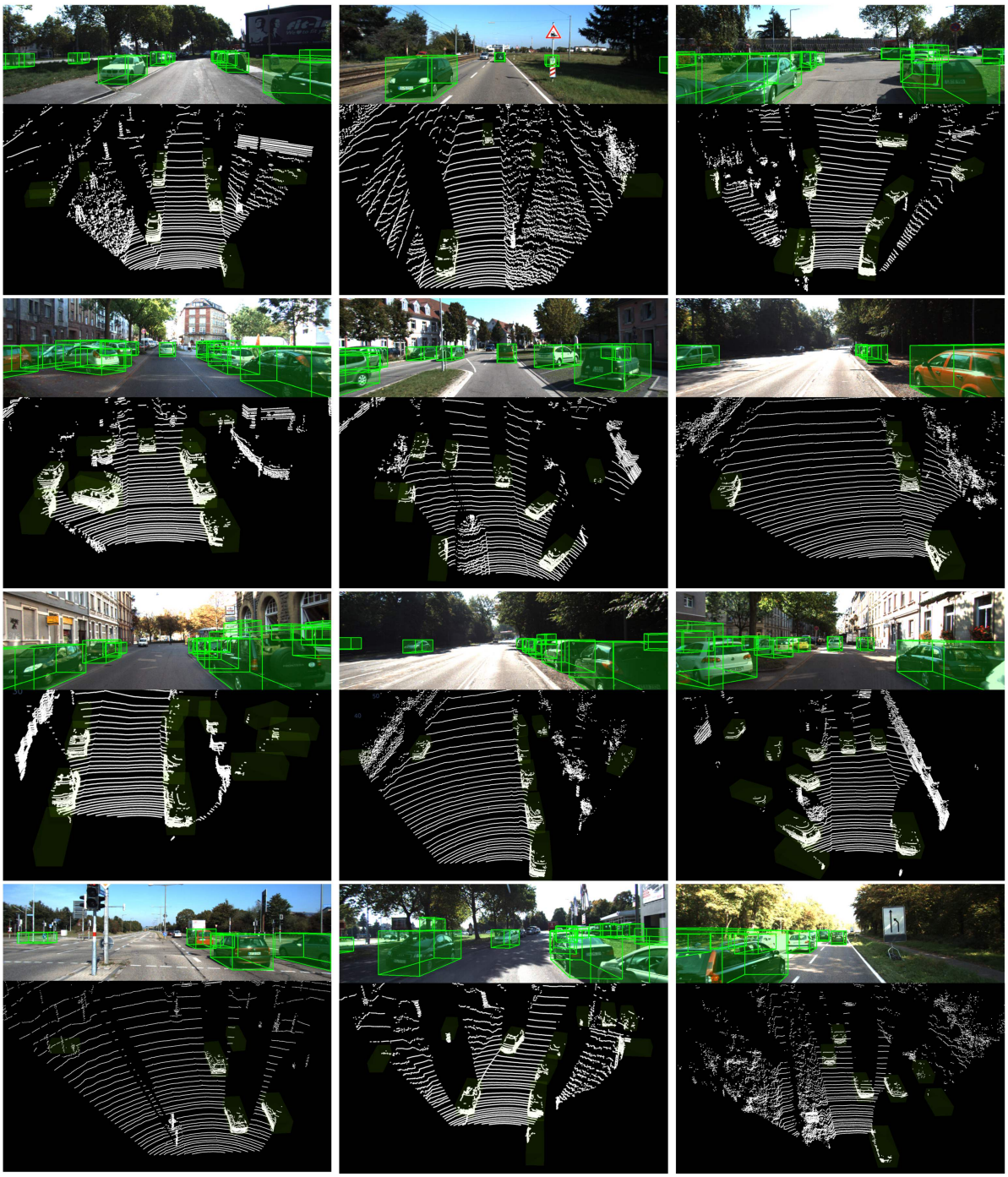


Figure 4.6: Pseudo-annotations of the KITTI target dataset obtained by using the multi-scale top-3 pseudo-annotation pipeline of SF-UDA^{3D}, based on scale transformation, temporal coherency, and weighted multi-scale aggregation.

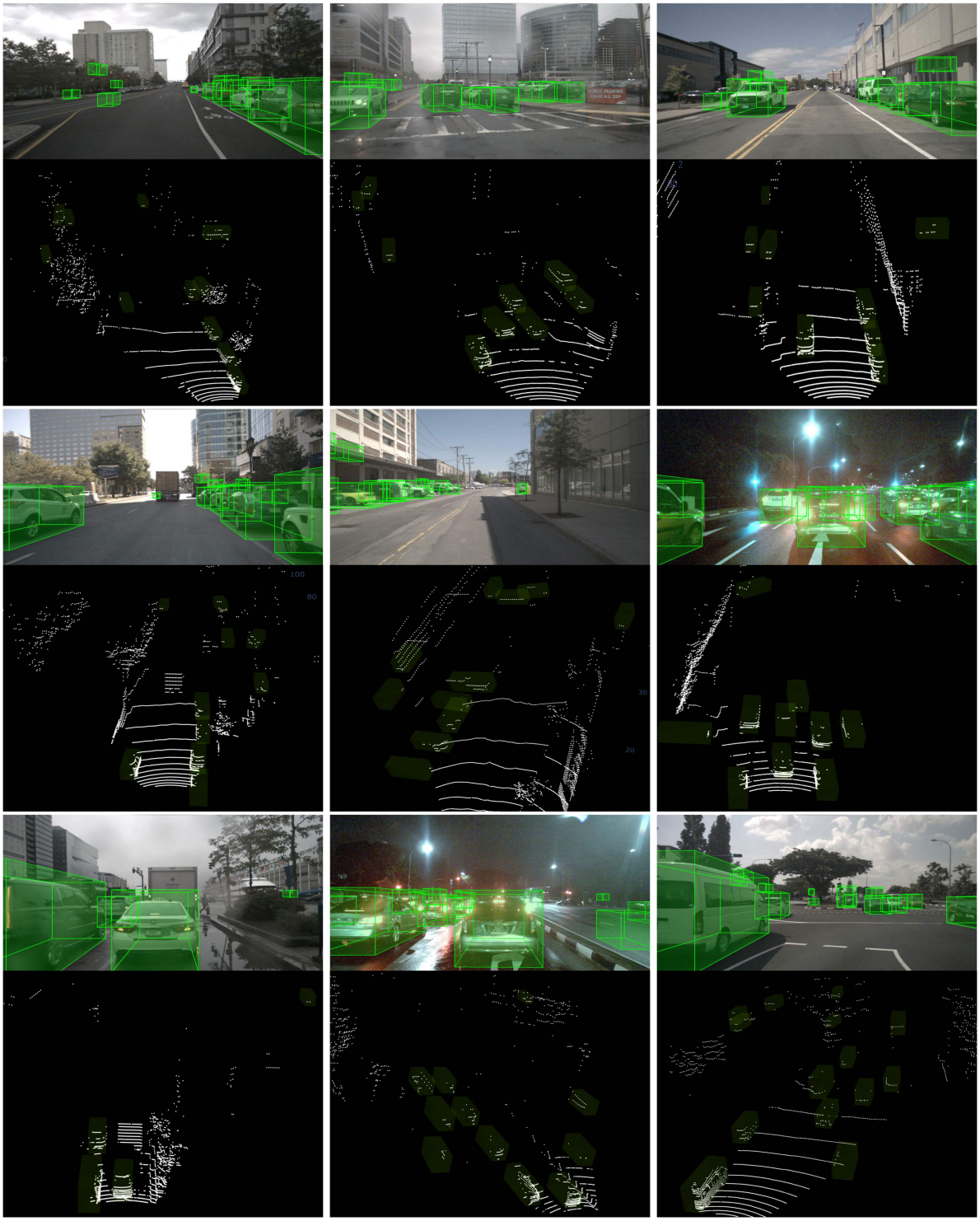


Figure 4.7: Pseudo-annotations of the nuScenes target dataset obtained by using the multi-scale top-3 pseudo-annotation pipeline of SF-UDA^{3D}, based on scale transformation, temporal coherency, and weighted multi-scale aggregation.

Chapter 5

Setting 4: Sequential target

In this chapter, we delve into the fourth setting, "Setting 4: Sequential target". Similar to Setting 3, we begin with a pre-trained source model, with source data not available during adaptation. However, in this case, the unlabeled target dataset is presented as a continuous stream of sequential point clouds, introducing additional challenges, *i.e.*, the risk of negative drift. We focus on the task of 3D semantic segmentation and introduce a novel online adaptation method designed for source-free adaptation in scenarios where target point clouds are given sequentially. This setting is both novel and practically significant, as it simulates the scenario of a vehicle adapting while navigating through a new and unseen environment

5.1 GIPSO: Geometrically informed propagation for online adaptation in 3D LiDAR segmentation

Autonomous driving requires accurate and efficient 3D visual scene perception algorithms. Low-level visual tasks such as detection and segmentation are crucial to enable higher-level tasks such as path planning [29, 36] and obstacle avoidance [187]. Deep learning-based methods have proven to be

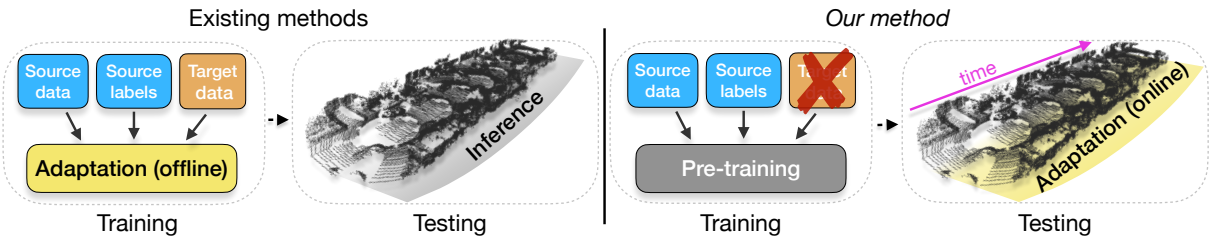


Figure 5.1: Existing methods adapt 3D semantic segmentation networks *offline*, requiring both source and target data. Differently, real-world applications urge solutions capable of adapting to unseen scenes online having access only to a pre-trained model.

the most suitable option to meet these requirements so far, but at the cost of requiring large-scale annotated datasets for training [224]. Relying only on annotated data is not always a viable solution. This problem can be mitigated by considering synthetic data, as it can be generated at a low cost with potentially unlimited annotations and under different environmental conditions [44, 69]. However, when a model trained on synthetic data is deployed in the real world, typically, it will underperform due to domain shift, *e.g.*, caused by varying lighting conditions, clutter, occlusions and materials with different reflective properties [5]. We argue that a 3D semantic segmentation algorithm running on an autonomous vehicle should be capable of adapting online – handling scenarios that are visited for the first time while driving – and it should do so by only using the newly captured data. A variety of research works have addressed the adaptation problem in the context of 3D semantic segmentation. However, most approaches operate offline and assume to have access to training (source) data [90, 235, 221, 245, 246, 204]. In this section, we argue that these two assumptions are too restrictive in an autonomous driving scenario (Fig. 5.1). On the one hand, offline adaptation would be equivalent to performing model adaptation on the data a vehicle has captured when the navigation has terminated, which is clearly a sub-optimal solution for autonomous driving [75]. On the other hand, having to rely on source data may not be

a viable option, as it requires the method to store and query a potentially large amount of data, thus hindering scalability [103, 225].

To overcome these limitations, in this section, we explore the new problem of Source-Free Online Unsupervised Domain Adaptation (SF-OUDA) for semantic segmentation, *i.e.*, that of adapting a deep semantic segmentation model while a vehicle navigates in an unseen environment without relying on human supervision. Specifically, in this chapter, we first implement, adapt and thoroughly analyze existing adaptation methods for the 3D semantic segmentation problem in a SF-OUDA setup. We experimentally observe that none of these methods provides consistent and satisfactory performance when employed in a SF-OUDA setting. However, there are elements of interest that, when carefully combined and extended, can be generally applicable. This leads us to move toward and design GIPSO (Geometrically Informed Propagation for Source-free Online adaptation), the first SF-OUDA method for 3D point cloud segmentation that builds upon recent advances in the literature, and exploits geometry information and temporal consistency to support the domain adaptation process. We also introduce two new synthetic datasets to benchmark SF-OUDA in two different real-world datasets, *i.e.*, SemanticKITTI [9, 51, 1] and nuScenes [15]. We validate our approach on these new synthetic-to-real benchmarks. Our motivation for creating these datasets is to make evaluation more comprehensive and to assess the generalization ability of different techniques to different experimental setups. In summary, our **contributions** are:

- A thorough experimental analysis of existing domain adaptation methods for 3D semantic segmentation in a SF-OUDA setting.
- A novel method for SF-OUDA that exploits low-level geometric properties and temporal information to continuously adapt a 3D segmentation model.

Table 5.1: Comparison between public synthetic datasets and Synth4D in terms of sensor specifications, acquisition areas, number of scans, number of points, presence of odometry data, and whether the semantic classes are all or partially shared.

Name	Specifications		Areas	Scans	Points	Odometry	Shared semantic classes	
	Sensor	FOV					S-KITTI [3]	nuScenes [4]
SinthCity [58]	MLS	360°	city	1	367M		no	no
GTA-LiDAR [204]	HDL64E	90°	town	121087	-		partial	no
PreSIL [69]	HDL64E	90°	town	51074	3135M		partial	no
SynLiDAR [208]	HDL64E	360°	city, town harbor, rural	198396	19482M		all	no
Synth4D (ours)	HDL64E HDL32E	360°	city, town rural, highway	20000 20000	2000M 2000M	✓	all	all

- The introduction of two new LiDAR synthetic datasets that are compatible with the SemanticKITTI and nuScenes datasets.

5.1.1 Datasets for synthetic-to-real adaptation

Autonomous driving simulators enable users to create ad-hoc synthetic datasets that can resemble real-world scenarios. Examples of popular simulators are GTA-V [213] and CARLA [44]. In principle, synthetic datasets should be compatible with their real-world counterpart [9, 15, 51], *i.e.*, they should share the same semantic classes and the same sensor specifications, such as the resolution (32 vs. 64 channels) and the horizontal field of view (*e.g.*, 90° vs. 360°). However, this is not the case for most of the synthetic datasets in the literature. The SynthCity [58] dataset contains large-scale point clouds that are generated from collections of several LiDAR scans, making it unsuitable for online domain adaptation as no odometry data is provided. PreSIL [69] and GTA-LiDAR's [204] point clouds are captured from a moving vehicle using a simulated Velodyne HDL64E [104], as that of SemanticKITTI, however they are rendered with a different field of view, *i.e.*, 90° as opposed to 360° of SemantiKITTI. SynLIDAR's [208] point

Table 5.2: Number of annotated points for each adaptation category for the simulated Velodyne HDL32E and Velodyne HDL64E. Each sensor setup was acquired in a different run.

Velodyne	**# labels** (10^8)						
	vehicle	pedestrian	road	sidewalk	terrain	manmade	vegetation
HDL32E	2.52	0.04	4.35	1.07	0.95	1.48	1.24
HDL64E	1.15	0.03	6.09	1.25	1.51	1.11	0.75

clouds are obtained using a simulated Velodyne HDL64E with 360° field of view, as in SemantiKITTI. However, the odometry data is not provided, *i.e.*, point clouds are all configured in their local reference frame. Therefore, domain adaptation algorithms that are based on ray-casting like [90] cannot be used.

To enable full compatibility with SemanticKITTI [9] and nuScenes [15], we present a new synthetic dataset, namely Synth4D, which we created using the CARLA simulator [44]. Tab. 5.1 compares Synth4D to the other synthetic datasets. Synth4D is composed of two sets of point cloud sequences, one compatible with SemanticKITTI and one compatible with nuScenes. Each set is composed of 20K labeled point clouds. Synth4D is captured using a vehicle navigating in four scenarios, *i.e.*, town, highway, rural area, and city.

Because UDA requires consistent labels between source and target, we mapped the labels of Synth4D with those of SemanticKITTI/nuScenes using the original instructions given to annotators [9, 15], thus producing eight macro classes: *vehicle*, *pedestrian*, *road*, *sidewalk*, *terrain*, *manmade*, *vegetation* and *unlabelled*. Fig. 5.2 shows examples of annotated point clouds from Synth4D. Moreover, we report in Tab. 5.2 the class distributions for Synth4D after class mapping. Additional information about the mapping can be found in our paper [154].

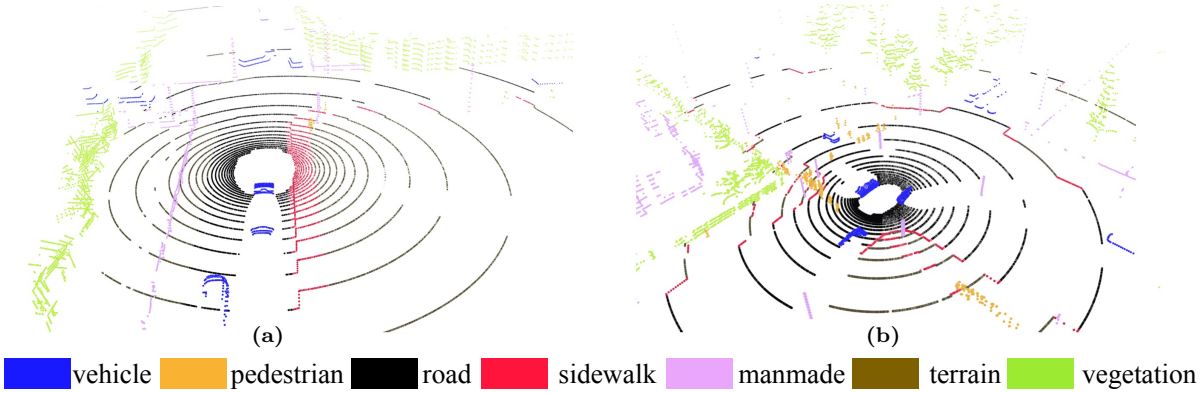


Figure 5.2: Example of point clouds from Synth4D using the simulated Velodyne (a) HDL32E and (b) HDL64E.

5.1.2 Method

Preliminaries and definitions

We formulate the problem of SF-OUDA for 3D point cloud segmentation as follows. Given a deep network model $F_{\mathcal{S}}$ that is pre-trained with supervision on the source domain $\mathcal{S}$, we aim to adapt $F_{\mathcal{S}}$ on the target domain $\mathcal{T}$ given an unlabelled point cloud stream as input. $F_{\mathcal{S}}$ is pre-trained using the source data $\Gamma_{\mathcal{S}} = \{(X^i_{\mathcal{S}}, Y^i_{\mathcal{S}})\}_{i=1}^{M_{\mathcal{S}}}$, where $X^i_{\mathcal{S}}$ is a synthetic point cloud, $Y^i_{\mathcal{S}}$ is the segmentation mask of $X^i_{\mathcal{S}}$ and $M_{\mathcal{S}}$ is the number of available synthetic point clouds. Let $X^t_{\mathcal{T}}$ be a point cloud of our stream at time t and $F^t_{\mathcal{T}}$ be the target model adapted using $X^t_{\mathcal{T}}$ and $X^{t-w}_{\mathcal{T}}$, with $w > 0$. $Y_{\mathcal{T}}$ is the set of unknown target labels and C is the number of classes contained in $Y_{\mathcal{T}}$. The source classes and the target classes are coincident.

Our approach

The input to GIPSO is the point cloud $X^t_{\mathcal{T}}$ and an already processed point cloud $X^{t-w}_{\mathcal{T}}$. These point clouds are used to adapt $F_{\mathcal{S}}$ to $\mathcal{T}$ through self-supervision (Fig. 5.3). Two modules process the input. The first module aims to create labels for self-supervision by segmenting $X^t_{\mathcal{T}}$ with the source

model $F_{\mathcal{S}}$. Because an unsupervised deep network produces these labels, we refer to them as *pseudo-labels*. We select a subset of segmented points that have reliable pseudo-labels through adaptive selection criteria and propagate them to less reliable points. The propagation uses geometric similarity in the feature space to increase the number of pseudo-labels available for self-supervision. To this end, we use an auxiliary deep network (F_{aux}) that is specialized in extracting geometrically-informed representations from 3D points. The second module aims to encourage temporal regularization of semantic information between $X_{\mathcal{T}}^{t}$ and $X_{\mathcal{T}}^{t-w}$. Unlike recent works [67], where a global point cloud descriptor of the scene is learned, we exploit a self-supervised framework based on stop gradient [23] to ensure smoothness over time. Self-supervision through pseudo-label geometric propagation and temporal regularization are concurrently optimized to achieve the desired domain adaptation objective.

Adaptive pseudo-label selection

An accurate selection of pseudo-labels is key to reliably adapt a model. In dynamic real-world scenarios, where new structures appear/disappear in/from the LiDAR field of view, traditional pseudo-labeling techniques [220, 168] can suffer from unexpected variations of class distributions, producing overconfident incorrect pseudo-labels and making more populated classes prevail on others [245, 246]. We overcome this problem by designing a class-balanced adaptive-thresholding strategy to choose reliable pseudo-labels. First, we compute an uncertainty index to filter out likely unreliable pseudo-labels. Second, we apply a different threshold for each class based on the uncertainty index distribution. This uncertainty index is directly related to the robustness of the output class distribution for each point. Robust pseudo-labels can be extracted from those points that consistently provide similar output distributions under different dropout per-

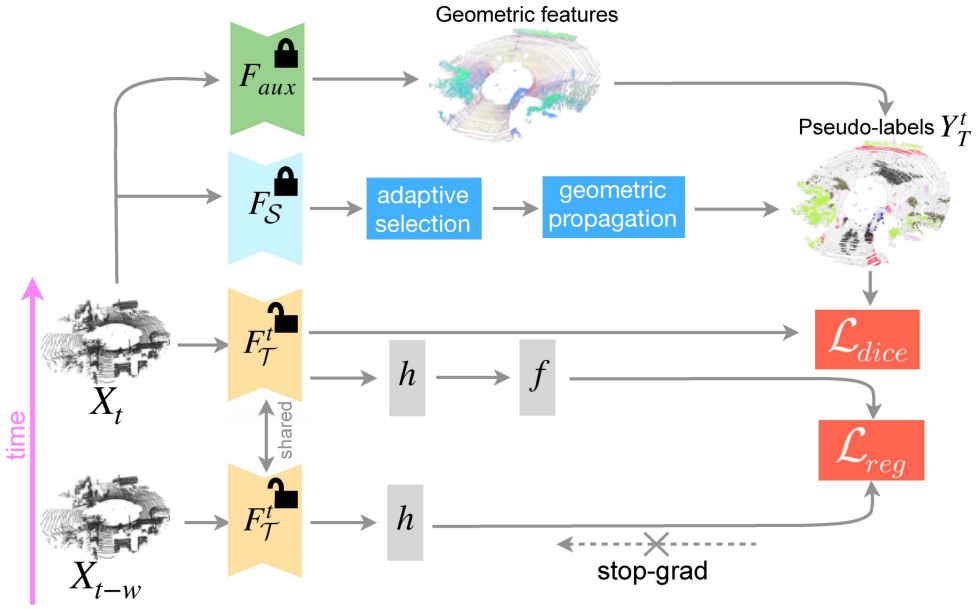


Figure 5.3: Overview of GIPSO. A source pre-trained model F_S selects *seed pseudo-labels* through our adaptive-selection approach. An auxiliary model F_{aux} extracts geometric features to guide pseudo-label propagation. $\mathcal{L}_{dice}$ is minimised over the pseudo-labels Y_T^t. In parallel, semantic smoothness is enforced with $\mathcal{L}_{reg}$ over time. (🔒) frozen parameters. (🔓) learnable parameters.

turbations [77]. We found that this approach works better than alternative confidence based approaches [245, 246].

Given the point cloud X_T^t, we perform J iterations of inference with F_S by using dropout and obtain

$$p_T^t = \frac{1}{J}\sum_{j=1}^{J} p\left(F_S | X_T^t, d_j\right), \tag{5.1}$$

where p_T^t is the averaged output distribution of F_S given X_T^t and d_j, *i.e.*, the dropout at j-th iteration. We compute the uncertainty index ν_T^t as the variance over the C classes of p_T^t as

$$\nu_T^t = E\left[\left(p_T^t - \mu_T^t\right)^2\right], \tag{5.2}$$

where $\mu_T^t = E[p_T^t]$ is the expected value of p_T^t. Then, we select the least uncertain points by using a different uncertainty threshold for each class.

Let λ_c^t be the uncertainty threshold of class c at time t. Since $\nu_{\mathcal{T}}^t$ defines the uncertainty for each point, we group $\nu_{\mathcal{T}}^t$ values per class and compute λ_c^t as the a-th percentile of $\nu_{\mathcal{T}}^t$ for class c. Therefore, at time t and for class c, we select only those pseudo-labels having the corresponding uncertainty index lower than λ_c^t and use the corresponding pseudo-labels as *seed pseudo-labels*.

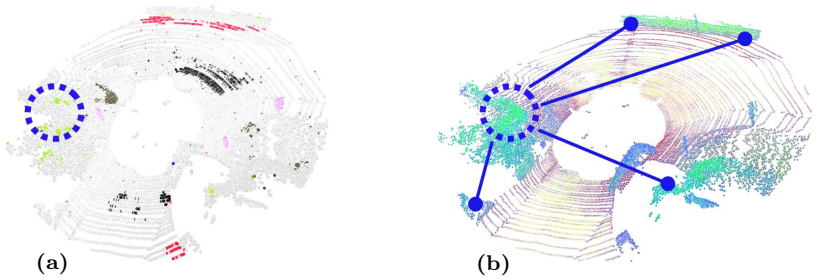
Geometric pseudo-label propagation

Typically, seed pseudo-labels are few and uninformative for the adaptation of the target model – the deep network is already confident about them. Therefore, we aim to propagate these pseudo-labels to potentially informative points. This is challenging because the model may drift during adaptation. We propose to use the features produced by an auxiliary geometrically-informed encoder F_{aux} to propagate seed pseudo-labels to geometrically-similar points. Geometric features can be extracted using deep networks that compute 3D local descriptors [55, 4, 129]. 3D local descriptors are compact representations of local geometries with great generalization abilities across domains. Our intuition is that, while the propagation in the metric space may propagate only in the spatial neighborhood of seed pseudo-labels, the use of geometric features would allow us to propagate to geometrically similar points, which can be distant from their seeds in the metric space (Fig. 5.4).

Given a seed pseudo-labeled point $\tilde{\mathbf{x}}^t \in X_{\mathcal{T}}^t$, we compute a set of geometric similarities as

$$\mathcal{G}_{\tilde{\mathbf{x}}}^t = \|F_{aux}(\tilde{\mathbf{x}}^t) - F_{aux}(X_{\mathcal{T}}^t)\|_2, \tag{5.3}$$

where $||\cdot||_2$ is the l_2-norm and $\mathcal{G}_{\tilde{\mathbf{x}}}^t$ is the set that contains the similarity values between $\tilde{\mathbf{x}}^t$ and all the other points of $X_{\mathcal{T}}^t$ (except $\tilde{\mathbf{x}}^t$). Then, we select the points that correspond to top K values in $\mathcal{G}_{\tilde{\mathbf{x}}}^t$ and assign the pseudo-label of $\tilde{\mathbf{x}}^t$ to them. Let $Y_{\mathcal{T}}^t$ be the final set of pseudo-labels that



(a) (b)

Figure 5.4: Example of geometric propagation: a) starting from *seed pseudo-labels*, b) geometric features are used to expand labels toward geometrically consistent regions.

we use for fine-tuning our model.

Self-supervised temporal consistency loss

While the vehicle moves, the LiDAR sensor samples the environment from different viewpoints generating point clouds with different point distributions due to clutter and/or occlusions. As points of consecutive point clouds can be simply matched over time by using the vehicle's odometry [51, 15], we can reasonably consider local variations of point distributions as local augmentations with the same semantic information. As a result, we can exploit recent self-supervised techniques to enforce temporal smoothness of our semantic features.

We begin by computing the set of corresponding points between $X_{\mathcal{T}}^{t-w}$ and $X_{\mathcal{T}}^{t}$ by using the vehicle's odometry. Let $T_{t-w\to t} \in \mathbb{R}^{4\times4}$ be the rigid transformation (from odometry) that maps $X_{\mathcal{T}}^{t-w}$ in the reference frame of $X_{\mathcal{T}}^{t}$. We define the set of corresponding point $\Omega^{t,t-w}$ as

$$\begin{aligned}\Omega^{t,t-w} = \big\{\{\mathbf{x}^t \in X_{\mathcal{T}}^{t}, \mathbf{x}^{t-w} \in X_{\mathcal{T}}^{t-w}\} : \\ \mathbf{x}^t = \texttt{NN}\left(T_{t-w\to t} \circ \mathbf{x}^{t-w}, X_{\mathcal{T}}^{t}\right), \\ \|\mathbf{x}^t - \mathbf{x}^{t-w}\|_2 < \tau\big\}, \end{aligned} \tag{5.4}$$

where $\texttt{NN}(n, m)$ is the nearest-neighbour search given the set m and the query n, $\circ$ is the operator that applies $T_{t-w\to t}$ to a 3D point and τ is a distance threshold.

We adapt the self-supervised learning framework proposed in SimSiam [23] to semantically smooth point clouds over time. We add an encoder network $h(\cdot)$ and a predictor head $f(\cdot)$ to the target model $F_{\mathcal{T}}$ and minimize the negative cosine similarity between consecutive semantic representations of corresponding points. Let $z^t \triangleq h(x^t)$ be the encoder features over the target backbone for x^t and let $q^t \triangleq f(h(x^t))$ be the respective predictor features. We minimize the negative cosine similarity as

$$\mathcal{D}_{t \to t-w}(q^t, z^{t-w}) = -\frac{q^t}{\|q^t\|_2} \cdot \frac{z^{t-w}}{\|z^{t-w}\|_2} \tag{5.5}$$

Time consistency is symmetric in the backward direction, hence we use the corresponding point of x^t from $\Omega^{t,t-w}$ and define our self-supervised temporal consistency loss as

$$\mathcal{L}_{reg} = \frac{1}{2}\mathcal{D}_{t \to t-w}(q^t, z^{t-w}) + \frac{1}{2}\mathcal{D}_{t-w \to t}(q^{t-w}, z^t) \tag{5.6}$$

where stop-grad is applied on z^t and z^{t-w}.

Online model update

Classes are typically highly unbalanced in each point cloud, *e.g.*, a pedestrian class may be 1% the number of points of the *vegetation* class. To this end, we use the soft Dice loss [71] as we found it works well when classes are unbalanced. Let $\mathcal{L}_{dice}$ be our soft Dice loss that uses the pseudo-labels selected though Eq. 5.3 as supervision. We define the overall adaptation objective as $\mathcal{L}_{tot} = \mathcal{L}_{dice} + \mathcal{L}_{reg}$, where $\mathcal{L}_{reg}$ is our regularization loss defined in Eq. 5.6.

5.1.3 Experiments

Experimental setup

Datasets. We pre-train our source models on Synth4D and SynLiDAR [208], and validate our approach on the official validation sets of SemanticKITTI

[9] and nuScenes [15] (target domains). In SemanticKITTI, we use the sequence 08 that is composed of 4071 point clouds at 10Hz. In nuScenes, we use 150 sequences, each composed of 40 point clouds at 2Hz.

Implementation details. We use MinkowskiNet as deep network for point cloud segmentation [27]. We use ADAM: initial learning rate of 0.01 with exponential decay, batch-size 16, and weight decay 10^{-5}. As auxiliary network F_{aux}, we use the PointNet-based architecture proposed in [129] trained on Synth4D that outputs a geometric features (descriptors) for a given 3D point. For online adaptation, we fix the learning rate to 10^{-3} and do not use schedulers as they would require prior knowledge about the stream length. Because we adapt our model on each new incoming point cloud, we use batch-size equal to 1. We set J=5, a=1, τ=0.3cm and use 0.5 dropout probability. We set K=10, w=5 on SemanticKITTI, and K=5, w=1 on nuScenes. Parameters are the same in all the experiments.

Evaluation protocol. We follow the traditional evaluation procedure for online learning methods [20, 228], *i.e.*, we evaluate the model performance on a new incoming frame using the model adapted up to the previous frame. We compute the Intersection over Union (IoU) [138] and report the average IoU (mIoU) improvement over the source (averaged over all the target sequences). We also evaluate the online version of our source model by finetuning it with ground-truth labels for all the points in the scene (target). We also evaluate the target upper bound (target) of our method obtained from the online finetuning of our source models over labeled target point clouds.

Benchmarking existing methods for SF-OUDA

Because our approach is the first that specifically tackles SF-OUDA in the context of 3D point cloud segmentation, we perform an in-depth analysis of the literature to identify previous adaptation methods that can be re-

purposed for SF-OUDA. Additionally, we experimentally evaluate their effectiveness on the considered datasets. We identify three categories of methods, as detailed below.

Batch normalization-based methods perform domain adaptation by considering different statistics for source and target samples within Batch Normalization (BN) layers. Here, we consider ADABN [102] and ONDA [112]. ADABN [102] is a source-free adaptation method which operates by updating the BN statistics, assuming that all target data are available (offline adaptation). ONDA [112] is the online version of ADABN [102], where the target BN statistics are updated online based on the target data within a mini-batch. This can be regarded as a SF-OUDA method. However, these approaches are general-purpose methods and have not been previously evaluated for 3D point cloud segmentation.

Prototype-based adaptation methods use class centroids, *i.e.*, prototypes, to generate target pseudo-labels that can be transferred to other samples via clustering. We implement SHOT [103] and ProDA [231]. SHOT [103] exploits Information Maximization (IM) to promote cluster compactness during offline adaptation. We implement SHOT by adapting the pre-trained model with the proposed IM loss online on each incoming target point cloud. ProDA [231] adopts a centroid-based weighting strategy to denoise target pseudo-labels, while also considering supervision from source data. We adapt ProDA to SF-OUDA by applying the same weighting strategy but removing source data supervision. We update target centroids at each incremental learning step. We refer to our SF-OUDA version of SHOT and PRODA as SHOT* and ProDA*, respectively.

Self-training-based methods exploit source model predictions to adapt to the target domain by re-training the model. We implement CBST [245] and TPLD [168]. CBST [245] relies on prediction confidence to select the most reliable pseudo labels. A confidence threshold is computed offline for

each target class to avoid class unbalance. Our implementation of CBST, which we denote as CBST*, uses the same class balance selection strategy but updates the thresholds online on each incoming frame. Moreover, no source data are considered as we are in a SF-OUDA setting. TPLD [168], originally designed for 2D semantic segmentation, uses the pseudo-label selection mechanism in [245] but introduces a pixel pseudo-label densification process. We implement TPLD by removing source supervision and replacing the densification procedure with a 3D spatial nearest-neighbor propagation. Our version of TPLD is denoted as TPLD*.

Besides re-purposing existing approaches for SF-OUDA, we also evaluate an additional baseline, *i.e.*, the rendering-based method RayCast [90]. This approach is based on the idea that target-like data can be obtained with photorealistic rendering applied to the source point clouds. Thus, adaptation is performed by simply training on target-like data. While RayCast can be regarded as an offline adaptation approach, we select it as it only requires the parameters of the real sensor to obtain target-like data from source point clouds.

Results

Evaluating GIPSO. Tab. 5.3, 5.4 and 5.5 report the results of our quantitative evaluation in the cases of Synth4D → SemanticKITTI, Synlidar → Semantic KITTI and Synth4D → nuScenes, respectively. The numbers in the tables indicate the improvement over the source model. GIPSO achieves an average IoU improvement of +4.31 on Synth4D → SemanticKITTI, +3.70 on Synlidar → SemanticKITTI and +0.85 on Synth4D → nuScenes. GIPSO outperforms both offline and online methods by a large margin on Synth4D → SemanticKITTI and Synlidar → Semantic KITTI, while it achieves a lower improvement over Synth4D → nuScenes. On SemanticKITTI, GIPSO can effectively improve *road*, *sidewalk*, *terrain*, *manmade* and *vege-*

Table 5.3: Synth4D → SemanticKITTI online adaptation. Source: pre-trained source model (lower bound). We report absolute mIoU for Source and mIoU relative to Source for the other methods. Key. SF: Source-Free. UDA: Unsupervised DA. O: Online.

Model	SF	UDA	O	vehicle	pedestrian	road	sidewalk	terrain	manmade	vegetation	Avg
Source				63.90	12.60	38.10	47.30	20.20	26.10	43.30	35.93
Target	✓		✓	+16.84	+5.49	+8.48	+34.44	+51.92	+45.68	+39.09	+28.85
ADABN [102]	✓	✓		-7.80	-2.00	-10.20	-18.60	-7.70	+5.80	-0.70	-5.89
RayCast [90]		✓		+3.80	-2.60	-3.10	-0.50	+7.30	+4.50	+0.20	+1.37
ProDA*	✓	✓	✓	-57.77	-12.34	-37.36	-46.95	-19.97	-25.62	-42.48	-34.64
SHOT*	✓	✓	✓	-62.44	-12.00	-28.27	-40.20	-20.00	-25.47	-42.55	-32.99
ONDA [112]	✓	✓	✓	-13.60	-1.70	-10.60	-20.00	-7.10	+3.90	-5.10	-7.74
CBST*	✓	✓	✓	-0.13	**+0.58**	-1.00	-1.12	+0.88	+1.69	+1.03	+0.28
TPLD*	✓	✓	✓	+0.36	+1.18	-0.76	-0.71	+0.95	+1.74	+1.15	+0.56
GIPSO (Ours)	✓	✓	✓	**+13.12**	-0.54	**+1.19**	**+2.45**	**+2.78**	**+5.64**	**+5.54**	**+4.31**

tation. *vehicle* is the best performing class, which can achieve a mIoU above +13. *pedestrian* is the worst performing class on all the datasets. *pedestrian* is a challenging class because it is significantly unbalanced compared to the others, also in the source domain. Although we attempted to mitigate the problem of unbalanced classes using adaptive thresholding and soft Dice loss, there are still situations that are difficult to address. Indeed, GIPSO limitations are related to geometric propagation and long-tailed classes. If objects of different classes share similar geometric structures, the geometric propagation may be deleterious. This can be mitigated by using another sensor modality (*e.g.* RGB) or by accounting for multi-scale signals to exploit context information. If severe class unbalance occurs, semantic segmentation accuracy may be affected, *e.g. pedestrian* class in Tabs. 5.3-5.5. This can be mitigated by re-weighting the loss through a class-balanced term (computed on the source). On nuScenes, the improvement is minor because its lower resolutions make patterns less distinguishable and more difficult to segment.

Evaluating state-of-the-art methods. We also analyze the performance of the existing methods. Batch-normalisation based methods perform poorly

Table 5.4: SynLiDAR → SemanticKITTI online adaptation. Source: pre-trained source model (lower bound). We report absolute mIoU for Source and mIoU relative to Source for the other methods. Key. SF: Source-Free. UDA: Unsupervised DA. O: Online.

Model	SF	UDA	O	vehicle	pedestrian	road	sidewalk	terrain	manmade	vegetation	Avg
Source				59.80	14.20	34.90	53.50	31.00	37.40	50.50	40.19
Target	✓		✓	+21.32	+8.09	+11.51	+28.13	+40.46	+33.67	+30.63	+24.83
ADABN [102]	✓	✓		+3.90	-6.40	-0.20	-3.70	-5.70	+1.40	+0.30	-1.49
RayCast [90]		✓		-	-	-	-	-	-	-	-
ProDA*	✓	✓	✓	-53.30	-13.79	-33.83	-52.78	-30.52	-36.68	-49.29	-38.60
SHOT*	✓	✓	✓	-57.83	-12.64	-24.80	-46.02	-30.80	-36.83	-49.32	-36.89
ONDA [112]	✓	✓	✓	-2.90	-6.40	-2.20	-8.80	-7.60	-1.20	-6.70	-5.11
CBST*	✓	✓	✓	+0.99	-0.83	+0.55	+0.20	+0.74	-0.07	+0.38	+0.28
TPLD*	✓	✓	✓	+0.90	-0.48	+0.59	+0.33	+0.84	+0.07	+0.37	+0.37
GIPSO (Ours)	✓	✓	✓	**+13.95**	-6.76	**+3.26**	**+5.01**	**+3.00**	**+3.34**	**+4.08**	**+3.70**

on all the datasets, with only ADABN [102] showing a minor improvement on nuScenes. We argue that non-i.i.d. batch samples arising in the online setting are playing an important role in this degradation, as they can have detrimental effects on models with BN layers [169]. SHOT* and ProDA* perform poorly in almost all the experiments, except on Synth4D → nuScenes where ProDA* achieves +0.29. This minor improvement may be due to the short sequences of nuScenes (40 frames) making centroids less likely to drift. This does not occur in SemanticKITTI where the long sequence causes a rapid drift (see detailed in Sec. 5.1.4). CBST* and TPLD* improve on SemanticKITTI and perform poorly on nuScenes. This can be ascribed to the noisy pseudo-labels that are selected using their confidence-based filtering approach. Lastly, RayCast [90] achieves +1.37 on Synth4D → SemanticKITTI, but underperform on Synth4D → nuScenes with a degradation of -3.46. RayCast was originally proposed for real-to-real adaptation, therefore we believe that its performance may be affected by the large difference in point cloud resolution between Synth4D and nuScenes. RayCast underperforms GIPSO in the online setup, thus showing how offline solutions can fail in dynamic domains. Note that Ray-

Table 5.5: Synth4D → nuScenes online adaptation. Source: pre-trained source model (lower bound). We report absolute mIoU for Source and mIoU relative to Source for the other methods. Key. SF: Source-Free. UDA: Unsupervised DA. O: Online.

Model	SF	UDA	O	vehicle	pedestrian	road	sidewalk	terrain	manmade	vegetation	Avg
Source				22.54	14.38	42.03	28.39	15.58	38.18	54.14	30.75
Target	✓		✓	+3.76	+0.92	+9.41	+16.95	+19.79	+10.92	+10.71	+10.35
ADABN [102]	✓	✓		+1.23	-2.74	-1.24	+0.14	+0.53	+0.70	+4.03	+0.38
RayCast [90]		✓		-1.36	-9.69	-3.53	-3.42	-2.77	-2.54	-0.91	-3.46
ProDA*	✓	✓	✓	+0.57	**-1.40**	+0.73	+0.09	+0.71	+0.40	+0.91	+0.29
SHOT*	✓	✓	✓	**+0.82**	-1.77	+0.68	-0.05	-0.70	-0.54	+1.09	-0.07
ONDA [112]	✓	✓	✓	+0.34	-1.90	-1.19	-0.62	+0.18	-0.40	+0.58	-0.43
CBST*	✓	✓	✓	+0.37	-2.61	-1.35	-0.79	+0.19	-0.36	-0.45	-0.71
TPLD*	✓	✓	✓	+0.65	-1.90	-0.96	-0.39	+0.43	+0.07	+0.86	-0.18
GIPSO (Ours)	✓	✓	✓	+0.55	-3.76	**+1.64**	**+1.72**	**+2.28**	**+1.18**	**+2.36**	**+0.85**

Cast cannot be evaluated using Synlidar, because Synlidar does not provide odometry information.

5.1.4 In-depth analyses

Method components

Tab. 5.8 shows the results of our ablation study on Synth4D → Semantic KITTI. When we use only the adaptive pseudo-label selection (A) we can achieve +1.07 compared to the source. When we combine A with the temporal regularization (T) we can further improve by +3.65. Then, we can achieve our best performance through the geometric propagation (P) of the pseudo labels.

Method parameters

We perform ablations for different parameters of GIPSO on Synth4D → SemanticKITTI. First, we study the propagation size K by increasing it up to 100 for each seed pseudo-label. Then, we study how GIPSO performs by varying the time window w. Results report the performance on Source

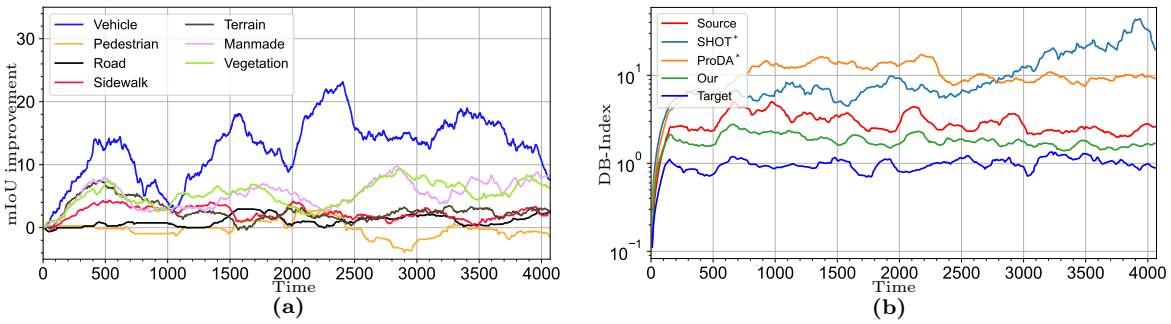


Figure 5.5: (a) Per-class improvement of GIPSO over time on Synth4D → SemanticKITTI. (b) DB-Index over time on Synth4D → SemanticKITTI. The lower the DB-Index, the better the class separation of the features.

Table 5.6: Online adaptation on Synth4D → SemanticKITTI with different propagation size K.

Model	K	**vehicle**	**pedestrian**	**road**	**sidewalk**	**terrain**	**manmade**	**vegetation**	**Avg**
Source	-	22.54	14.38	42.03	28.39	15.58	38.18	54.14	30.75
Target	-	+3.76	+0.92	+9.41	+16.95	+19.79	+10.92	+10.71	+10.35
Ours	1	+14.18	-1.13	+1.08	+2.11	+2.74	+5.49	+5.39	+4.27
Ours	5	+13.42	-0.51	+0.91	+2.16	+2.66	+5.54	+5.62	+4.26
Ours	10	+13.12	-0.54	+1.19	+2.45	+2.78	+5.64	+5.54	+4.31
Ours	50	+12.01	-1.00	+0.73	+2.01	+3.02	+5.51	+5.66	+3.99
Ours	100	+12.25	-2.49	+0.62	+1.93	+3.39	+5.99	+5.68	+3.91

(gray) in absolute mIoU while the others are reported as relative mIoU improvement over the Source model. Target is the supervised upper bound of our task in our setting.

Propagation size We study the effect of different propagation steps by using our geometry-based propagation. Tab. 5.6 shows the results with a K of $1, 5, 10, 50, 100$. We can see that mIoU starts to decrease when a higher number of propagation steps are used, *i.e.*, $K = 50$, whereas we reach the best improvement of +4.31 with $K = 10$. These results show that K should be set such that to both preserve pseudo-labelling accuracy while propagating seed labels towards new informative points.

Time-window length We study the effect of different time window length w in our self-supervised temporal consistency loss. Tab. 5.7 shows that w

Table 5.7: Online adaptation on Synth4D → SemanticKITTI with a different time window w.

Model	w	vehicle	pedestrian	road	sidewalk	terrain	manmade	vegetation	Avg
Source	-	22.54	14.38	42.03	28.39	15.58	38.18	54.14	30.75
Target	-	+3.76	+0.92	+9.41	+16.95	+19.79	+10.92	+10.71	+10.35
Ours	1	+9.73	-0.63	+0.56	+1.79	+2.86	+4.88	+4.27	+3.35
Ours	2	+11.76	-1.09	+0.78	+1.97	+2.50	+5.01	+5.23	+3.74
Ours	3	+12.89	-0.37	+0.79	+1.84	+2.70	+5.20	+5.12	+4.02
Ours	4	+13.84	-0.84	+0.94	+2.24	+2.57	+5.37	+5.49	+4.23
Ours	5	+13.12	-0.54	+1.19	+2.45	+2.78	+5.64	+5.54	+4.31
Ours	6	+13.95	-0.48	+0.95	+2.01	+2.77	+5.69	+5.93	+4.40
Ours	7	+13.32	-0.90	+1.11	+2.16	+3.14	+5.43	+5.74	+4.28
Ours	8	+13.16	-1.16	+0.95	+1.88	+2.67	+5.75	+6.20	+4.21

should be selected neither too large ($w = 8$) nor too small ($w = 1$) for the best performance. The time window w should be set based on the sampling rate of the sensor and the overlap between adjacent frames.

Oracle study

We analyze the importance of using a reliable pseudo-label selection metric. Tab. 5.9 shows the pseudo-label accuracy as a function of the points that are selected as the K-th best candidates based on the distance from their centroids (as proposed in [231]), confidence (as proposed in [245]) and uncertainty (ours). Centroid-based selection shows a low accuracy even at $K = 1$, which tends to worsen as K increases. Confidence-based selection is more reliable than the centroid-based selection. We found uncertainty-based selection to be more reliable at smaller values of K, which we deem to be more important than having more pseudo-labels but less reliable.

Per-class temporal behavior

Fig. 5.5a shows the mIoU over time for each class on Synth4D → SemanticKITTI. We can observe that six out of seven classes have a steady

Table 5.8: Synth4D → SemanticKITTI ablation study of GIPSO: (A) Adaptive thresholding; (A+T) A + Temporal consistency; (A+T+P) A+T + geometric Propagation.

Source	Target	A	A+T	A+T+P
35.95	+28.85	+1.07	+3.65	+4.31

Table 5.9: Oracle study on Synth4D → SemanticKITTI that compares the accuracy of different pseudo-label selection metrics: Centroid, Confidence and Uncertainty.

	Centroid	Confidence	Uncertainty
Top-1	38.1	66.7	76.1
Top-10	43.8	61.4	69.7

improvement: *vehicle* is the best performing class, followed by *vegetation* and *manmade*. Drops in mIoU are typically due to sudden geometric variations of the point cloud, *e.g.*, a road junction after a straight road, or a jammed road after an empty road. *pedestrian* confirms to be the most challenging class.

Temporal compactness of features

We assess how well points are organized in the feature space over time. We use the DB Index (DBI) that is typically used in clustering to measure the feature intra- and inter-class distances [35]. The lower the DBI, the better the quality of the features. We use SHOT* and ProDA* as comparisons with our method, and the source and target models as references. Fig. 5.5b shows the DBI variations over time. SHOT* behavior is typical of a drift, as features of different classes become interwoven. ProDA* does not drift, but it produces features that are worse than the source model. Our approach is between source and target models, with a tendency to get closer to target.

Different 3D local descriptors

We assess the effectiveness of different 3D local descriptors. We test FPFH [147] (handcrafted) and FCGF [28] (deep learning) descriptors. GIPSO achieves +3.56 mIoU with FPFH, +4.12 mIoU with FCGF and +4.31 mIoU with DIP. This is in line with the experiments shown in [130],

where DIP shows a superior generalization capability across domains than FCGF.

Performance with global features. We assess the GIPSO performance on Synth4D→SemanticKITTI when the global temporal consistency loss proposed in STRL [67] is used instead of our per-point loss (Eq. 5.5). This variation achieves +1.74 mIoU, showing that per-point temporal consistency is key.

Improving state-of-the-art with GIPSO

We show that our proposed modules also improve state-of-the-art methods, such as CBST [245], ProDA [231] and, TPLD [231], providing additional evidence that our propositions are steps forward in SF-OUDA not just in GIPSO. First, we show that our adaptive sampling strategy can be used in state-of-the-art methods to obtain more reliable pseudo-labels. Second, we propose modifications to further improve baselines performance in SF-OUDA. We propose the following modifications:

- CBST* uses a confidence based sampling strategy to select class-balanced pseudo-labels. We improve CBST* by using our adaptive selection strategy based on uncertainty;
- TPLD* builds upon CBST* by increasing pseudo-label number through densification and voting. We improve TPLD* with our more robust adaptive pseudo-label selection and substitute the spatial nearest neighbor with our geometrically informed propagation strategy.
- ProDA* exploits a centroid-based weighting strategy to denoise pseudo-labels. Moreover, momentum update is performed between source $F_{\mathcal{S}}$ and target model $F_{\mathcal{T}}$. We improve ProDA* in its three main parts. First, we remove source model momentum update as it promotes domain drift. Second, we substitute pseudo-labelling with our iterative

Table 5.10: Improvement of state-of-the-art methods by using GIPSO adaptive selection strategy and propagation strategy on Synth4D → SemanticKITTI.

Model	**vehicle**	**pedestrian**	**road**	**sidewalk**	**terrain**	**manmade**	**vegetation**	**Avg**
Source	22.54	14.38	42.03	28.39	15.58	38.18	54.14	30.75
Target	+3.76	+0.92	+9.41	+16.95	+19.79	+10.92	+10.71	+10.35
ProDA*	-58.92	-12.08	-36.74	-45.32	-15.46	-20.69	-39.24	-32.63
CBST*	-0.13	0.58	-1.00	-1.12	0.88	1.69	1.03	0.28
TPLD*	0.36	1.18	-0.76	-0.71	0.95	1.74	1.15	0.56
ProDA* (Ours)	2.04	4.40	0.24	0.62	0.29	1.07	1.71	1.48
CBST* (Ours)	2.72	-2.53	-0.19	0.56	1.48	3.02	2.46	1.07
TPLD* (Ours)	2.81	-2.33	-0.05	0.65	2.30	3.44	2.82	1.38

dropout based pseudo-labeling strategy. Third, we compute more robust centroids by considering the mean of point-features in our iterative pseudo-labelling strategy.

Tab. 5.10 shows that GIPSO components can be used to successfully improve the performance of existing methods. ProDA* improves from -32.63 to $+1.48$, we deem this is due to the more robust centroid computation and to the lower adaptation drift obtained with a non-updated source model. CBST* benefits from a better pseudo-label selection improving from $+0.28$ to $+1.07$. TPLD* benefits from a better pseudo-labels and the geometrically informed propagation improving from $*0.56$ to $+1.38$.

Qualitative results

Fig. 5.6 shows the comparison between GIPSO and the source model on on Synth4D→SemanticKITTI. The first row shows frame 178 of SemanticKITTI with an improvement of +27.14 mIoU (large). The classes *vehicle*, *sidewalk* and *terrain* are incorrectly segmented by the source model, we can see a significant improvement in segmentation on these classes after adaptation. The second and third rows show frame 1193 and frame 2625 with an improvement of +10.00 mIoU (medium) and +4.99 mIoU (small).

Improvements are visible after adaptation in the classes *vehicle*, *sidewalk* and *road*. The last row shows a segmentation drift for *road* that is caused by incorrect pseudo-labels.

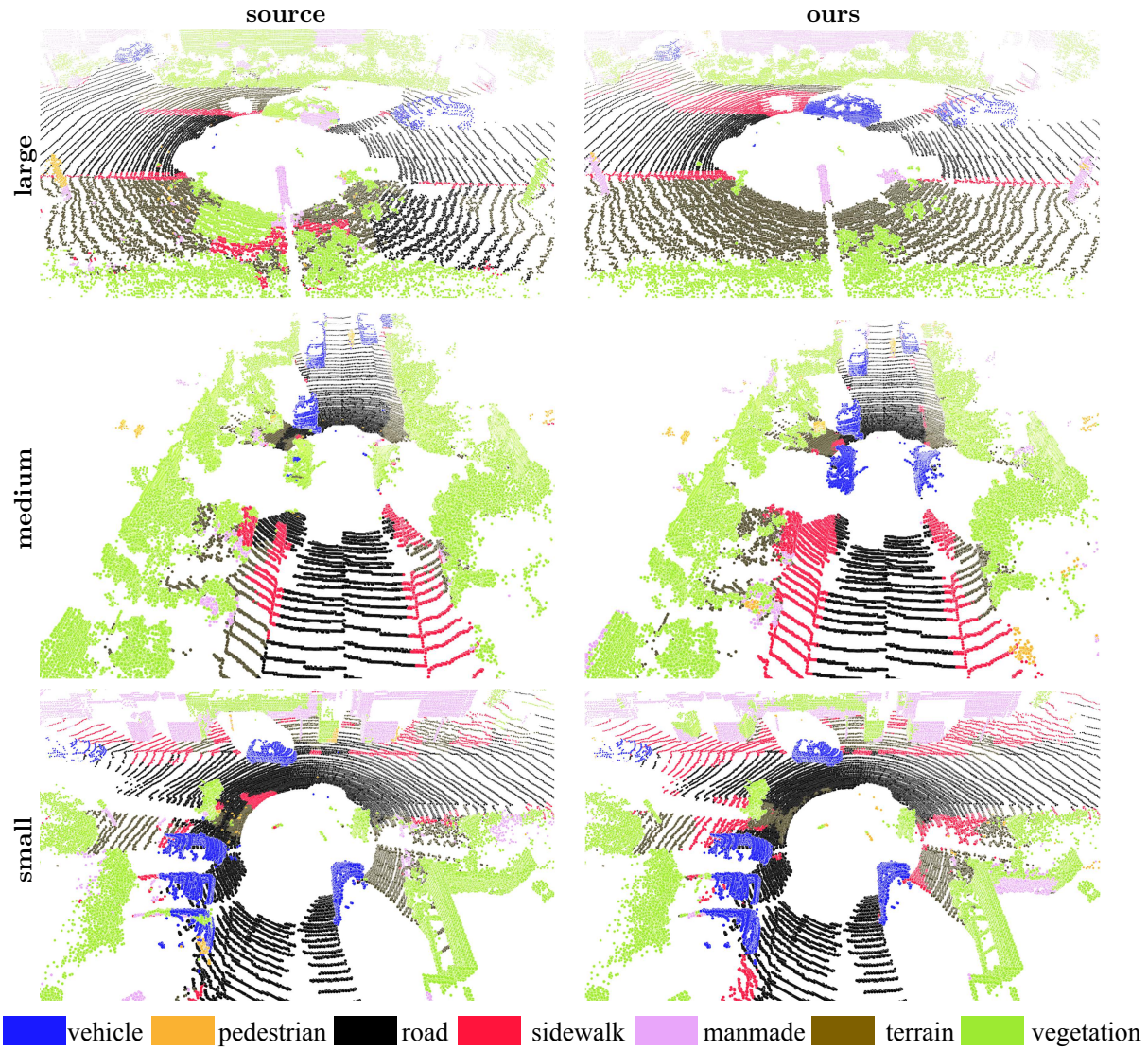


Figure 5.6: Results on Synth4D→SemanticKITTI with three different ranges of mIoU improvements, *i.e.*, large (+27.2), medium (+10.0) and small (+5.1).

Chapter 6

Setting 5: Unknown target

In this sixth chapter, we delve into the final setting, "Setting 5: Unknown target". This setting is one of the most difficult studied in this thesis. Indeed, we are given an annotated source dataset, but we miss any information about the future target domains. This specific experimental setup is also known as the domain generalization task. In this chapter, we study this setting for the first time in 3D semantic segmentation, specifically in LiDAR point clouds. Tackling this task is not trivial as domain shift needs to be solved in advance, aiming for generalization capabilities during source training. We address this problem by introducing a simple and robust method that leverages a dense auxiliary task during point cloud segmentation learning.

6.1 Walking your LiDOG: A journey through multiple domains for LiDAR semantic segmentation

We address the challenge of achieving accurate *and* robust semantic segmentation of LiDAR point clouds. LiDAR semantic segmentation (LSS) is one of the most fundamental perception problems in mobile robot navigation, with applications ranging from mapping [98], localization [110], and

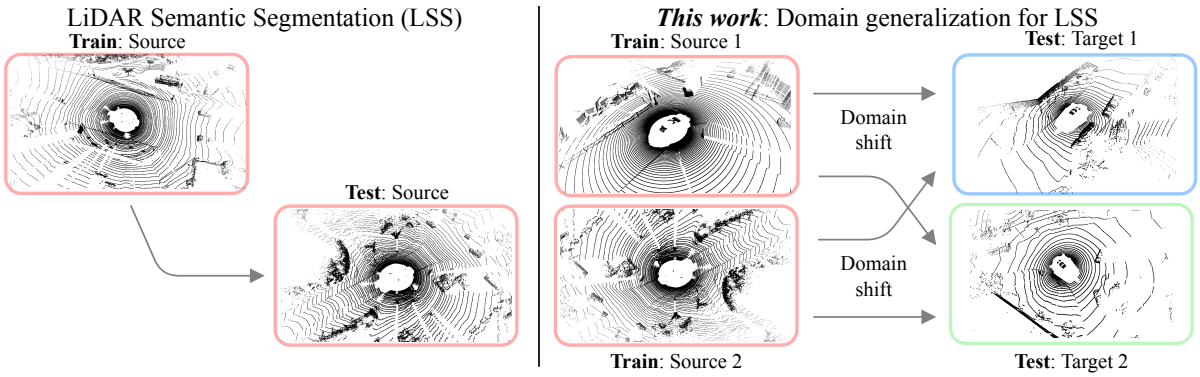


Figure 6.1: Domain Generalization for LiDAR Semantic Segmentation (DG-LSS). *Left*: Existing LSS methods are trained and evaluated on point clouds drawn from the same domain. *Right*: We focus on studying LSS under domain shifts, where the test samples are drawn from a different data distribution. This chapter aims to address the generalization aspect of this task.

online dynamic situational awareness [179].

Status Quo. State-of-the-art LSS methods [27, 66, 178, 243] perform well when trained *and* evaluated using the same sensory setup and environment (*i.e.*, source domain, Fig. 6.1, *left*). However, their performance degrades significantly in the presence of domain shifts (Fig. 6.1, *right*), commonly caused by differences in sensory settings, *e.g.*, a new type of sensor, or recording environments, *e.g.*, geographic regions with different road layouts or types of vehicles. One way to mitigate this is to collect multi-domain datasets for pre-training, similar to what has been done in the image domain using inexpensive cameras that are widely available [40, 105]. However, building a crowd-sourced collection of multi-domain LiDAR datasets is currently not feasible.

Stirring the pot. As a first step towards LiDAR segmentation models that are robust to domain shifts, we present the first-of-its-kind experimental test-bed for studying *Domain Generalization* (DG) in the context of *LiDAR Semantic Segmentation* (LSS). In our DG-LSS setup, we train and evaluate models on different domains, including two synthetic [154] and

two real-world densely labeled datasets [10, 8, 48], recorded in different geographic regions with different sensors. This evaluation setup reveals a significant gap in terms of mean intersection-over-union between models trained on the source and target domains: for example, a model transferred from SemanticKITTI to nuScenes dataset obtains 26.53 mIoU compared 48.49 mIoU obtained by the model fully trained on the target domain. *Could this gap be alleviated with DG techniques?*

Insights. To address this challenge, we propose *LiDOG* (**Li**DAR **DO**main **G**eneralization) as a simple yet effective method specifically designed for DG-LSS. In addition to reasoning about the scene semantics in 3D space, LiDOG projects features from the sparse 3D decoder onto the 2D bird's-eye-view (BEV) plane along the vertical axis and learns to estimate a dense 2D semantic layout of the scene. In this way, LiDOG encourages the 3D network to learn features that are robust to variations in, *e.g.*, type of sensor or geo-locations and thereby can be transferred across different domains. This directly leads to increased robustness toward domain shifts and yields +8.35 mIoU improvement on the target domain, confirming the efficacy of our approach. Our experimental evaluation confirms this approach is consistently more effective compared to prior efforts in data augmentations [230, 123, 153], domain adaptation techniques [196, 89], and image-based DG techniques [26, 125], applied to the LiDAR semantic segmentation.

Contributions. We make the following key contributions:

- We present the *first* study on *domain generalization* in the context of *LiDAR semantic segmentation*.
- We carefully construct a test-bed for studying DG-LSS using two synthetic and two densely-labeled real-world datasets recorded in different cities with different sensors.

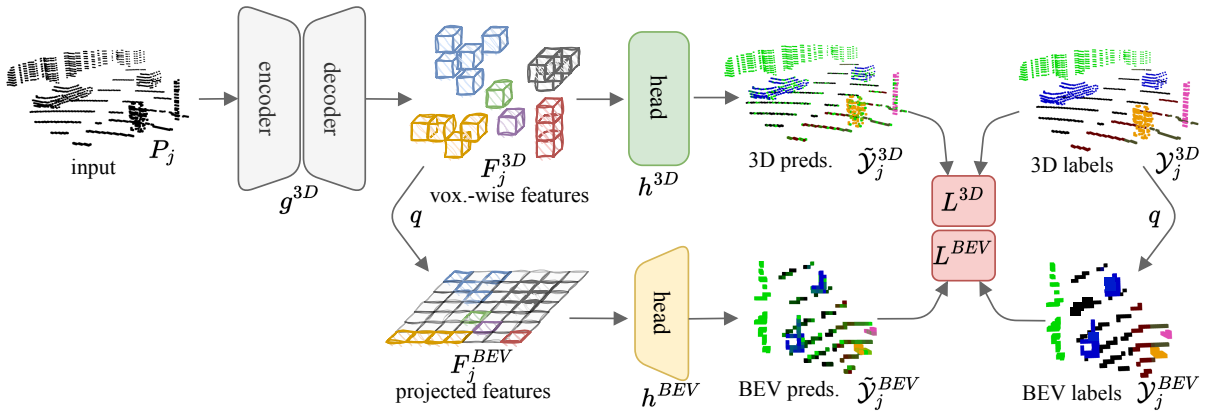


Figure 6.2: LiDOG overview. We encode our input LiDAR scan P_j using the 3D backbone g^{3D} to learn the occupied voxels' feature representations F^{3D}. (*Upper branch - main task*) We apply a sparse segmentation head on F^{3D} and supervise with 3D semantic labels, $\mathcal{Y}_j^{3D}$. (*Lower branch - auxiliary task*) We project those features along the height-axis to obtain a dense 2D bird's-eye (BEV) view features F^{BEV}, and apply several 2D convolutional layers to learn the 2D BEV representation. We supervise the BEV auxiliary task by using BEV-view of semantic labels, $\mathcal{Y}_j^{BEV}$. We train jointly on both L^{3D} and L^{BEV}.

- We rigorously study how prior efforts proposed in related domains can be used to tackle domain shift
- We propose LiDOG, a simple yet strong baseline that learns robust, generalizable, and domain-invariant features by learning semantic priors in the 2D birds-eye view. Despite its simplicity, we achieve state-of-the-art performance in all the generalization directions.

6.1.1 Method

Preliminaries and definitions

In this section, we first recap the standard LiDAR Semantic Segmentation (LSS) setting for completeness. Next, we formally introduce the problem of Domain Generalization for LiDAR Semantic Segmentation (DG-LSS), which we study in this chapter.

LiDAR semantic segmentation. In LSS, we are given a set of labeled training

instances in the form of point clouds $\mathcal{T}_{train} = \{P_j = \{\mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^{N_j}\}$. Each point cloud $P_j \sim \mathcal{D}$ consists of N_j points, and is drawn from a certain point cloud distribution $\mathcal{D}$. Moreover, point clouds in the training set are labeled according to a predefined semantic class vocabulary $\mathcal{K} = \{1, \ldots, K\}$ of K categorical labels. Given such a training set $\mathcal{T}_{train}$ along with class vocabulary $\mathcal{K}$, the goal of LSS is to learn a function $f(\mathbf{p}_i) \rightarrow \mathcal{K}$, that maps each point $\mathbf{p}_i \in P_j, \forall j$, to one of K semantic classes, such that the prediction error on the *same domain* $\mathcal{D}$ is minimized. In other words, LSS models are trained *and* evaluated on (disjunct) sets of point clouds that are drawn from the same data distribution $\mathcal{D}$. We note that it is standard practice for LSS methods to be evaluated on multiple datasets. However, standard multi-dataset evaluation follows the setting described above: models are trained *and* evaluated for *each* domain *separately*.

Domain generalization for LiDAR semantic segmentation. In this chapter, we release the assumption that train and test splits were sampled from the same domain and explicitly study cross-*domain generalization* in LSS. We are given M labeled datasets $\mathcal{T}_m \sim \mathcal{D}_m$, $m \in \{1, \ldots, M\}$ sampled from different data distributions $\mathcal{D}_m$. Importantly, all datasets must be labeled according to a common semantic vocabulary $\mathcal{K}$, *i.e.*, we are studying DG in closed-set conditions [238]. Then, the task can be defined as follows: we use $\mathcal{T}_{\{m\}}$ datasets for training, and the held-out datasets $\mathcal{T}_{\{m'\}}$, $\{m'\} \cap \{m\} = \emptyset$, solely for testing.

The goal of DG-LSS is, therefore, to train a mapping $f(\mathbf{p}_i) \rightarrow \mathcal{K}$, such that the prediction performance on the target datasets $\mathcal{T}_{\{m'\}} \sim \mathcal{D}_{\{m'\}}$ is maximized. We only have access to data and labels from the target datasets during evaluation. The model has never *seen any target data* before evaluation, neither labeled nor unlabeled.

Overview

We provide a schematic overview of LiDOG: *Li*dar *DO*main *G*eneralization network in Fig. 6.2. LiDOG leverages a 3D sparse-convolutional encoder-decoder neural network (Fig. 6.2, g^{3D}) that encodes the input point cloud P_j into sparse 3D features F^{3D} (Fig. 6.2). In order to tackle the domain shift problem, we propose to augment this network with a dense top-down prediction auxiliary task during model training. In particular, we project sparse 3D features F^{3D} from the decoder network along the height axis to obtain a 2D bird's-eye view (BEV) representation F^{BEV} of the features learned by the 3D network. Then, we jointly optimize the losses for the 3D and BEV network heads. Importantly, the auxiliary BEV head is not required during model inference. Its sole purpose is to effectively learn domain-agnostic features during training.

3D segmentation branch

The upper branch of LiDOG (Fig. 6.2) aims to learn the main segmentation task, *i.e.*, it learns the mapping function $f(\mathbf{p}_i) \rightarrow \mathcal{K}$. We first process the input LiDAR point cloud P_j into a 3D occupancy (voxel) grid V_j. During this phase, the continuous 3D input space is evenly sampled into discrete 3D cells. Points falling into the same cell are merged. We then learn a feature representation of the occupancy grid V_j via a 3D encoder-decoder network g^{3D} based on the well-established sparse-convolutional backbone [27]. The encoder consists of a series of sparse 3D convolutional downsampling layers that learn a condensed 3D representation, while the decoder upsamples the features back to the original resolution with sparse convolutional upsampling layers. We use sparse batch-normalization layers. The output of g^{3D} is a set of voxel-wise features $F_j^{3D} = g^{3D}(V_j)$, where each feature vector is associated with a voxel cell in V_j. We employ a

sparse segmentation head to obtain semantic posteriors that predict the voxel-wise categorical distribution from the voxel features F_j^{3D}: $\tilde{\mathcal{Y}_j^{3D}} = p(\mathcal{K}|V_j) = \sigma(h^{3D}(F_j^{3D}))$, where σ denotes the softmax activation function.

Dense auxiliary task

In the lower branch of LiDOG, we add the auxiliary task that encourages the network to learn a representation that is robust to domain shifts. We transform 3D features to bird's-eye view (BEV) space, and train a network to estimate BEV semantic scene layout.

Sparse-to-dense feature projection. We start our dense-to-sparse segmentation task by projecting voxel-wise features F_j^{3D} into dense BEV-features feature space F_j^{BEV}. Given a voxel $v_i \in V_j$ center coordinates (x_i, y_i, z_i), we compute its corresponding BEV coordinates (x_i^{BEV}, z_i^{BEV}) as:

$$q(x_i, z_i) = (x_i^{BEV}, z_i^{BEV}) = \left(\left\lfloor \frac{x_i}{x_q} \right\rfloor, \left\lfloor \frac{z_i}{z_q} \right\rfloor \right), \tag{6.1}$$

where q is the projection function, and x_q and z_q are the quantization parameters. We project only voxels within projection bounds $B^{3D} = [(-b_x, b_x), (-b_z, b_z)]$, where b_x and b_z are the bound values along the x and z axis, and shift projected coordinates to fit in the BEV-features height and width. Consequently, we define BEV features F_j^{BEV} by initializing them to zero and by filling non-empty cells as $F_j^{BEV}(x_i^{BEV}, z_i^{BEV}) = F_i^{3D}$. When multiple voxels project to the same cell, we randomly keep one.

Sparse-to-dense label projection. We obtain source labels by following the same procedure used for F_j^{BEV}. After voxelization, each voxel v_i is associated to a label category $y_i^{3D} \in \mathcal{Y}_j^{3D}$. We use Eq. 6.1 and the same bounds B^{3D} to compute the label coordinates (x_i^{BEV}, z_i^{BEV}) of BEV image pixels. Then, we define BEV labels as: $\mathcal{Y}_j^{BEV}(x_i^{BEV}, z_i^{BEV}) = y_i^{3D}$.

BEV predictions. We predict BEV semantic posteriors by employing a

Figure 6.3: LiDAR point clouds and their corresponding BEV views: SemanticKITTI (*left*) and nuScenes (*right*). After projection, BEV images are geometrically more similar.

standard 2D segmentation head h^{BEV} which predicts semantic classes in the dense 2D BEV plane and takes as input BEV features F_j^{BEV}. First, we reduce the dimensionality of F_j^{BEV} through a pooling operation. Then, we feed h^{BEV} with F_j^{BEV} and obtain BEV semantic predictions: $\tilde{\mathcal{Y}_j^{BEV}} = \sigma(h^{BEV}(F_j^{BEV}))$, where σ denotes the softmax activation function.

Losses

Given 3D semantic predictions $\tilde{\mathcal{Y}_j^{3D}}$ and labels $\mathcal{Y}_j^{3D}$, as well as BEV semantic predictions $\tilde{\mathcal{Y}_j^{BEV}}$ and labels $\mathcal{Y}_j^{BEV}$, we train both LiDOG network heads jointly using soft DICE loss [71], *i.e.*, $L^{3D} = dice(\tilde{\mathcal{Y}_j^{3D}}, \mathcal{Y}_j^{3D})$, and $L^{BEV} = dice(\tilde{\mathcal{Y}_j^{BEV}}, \mathcal{Y}_j^{BEV})$. Finally, we average contributions of both losses: $L_{tot} = \frac{1}{2}(L^{BEV} + L^{3D})$. DICE loss has been shown in literature [71, 153] to perform favorably for rarer classes compared to the cross-entropy loss.

6.1.2 Why our approach works

We demonstrate empirically in Sec. 6.1.3 that the dense BEV prediction proxy task significantly improves model performance in terms of cross-domain generalization. Why does this simple objective yield, in practice, more robust feature representations? There are several possible sources of domain shift, *e.g.*, different sensors, different environments, and scans are visibly different (Fig. 6.3). After the projection, we obtain a denser label

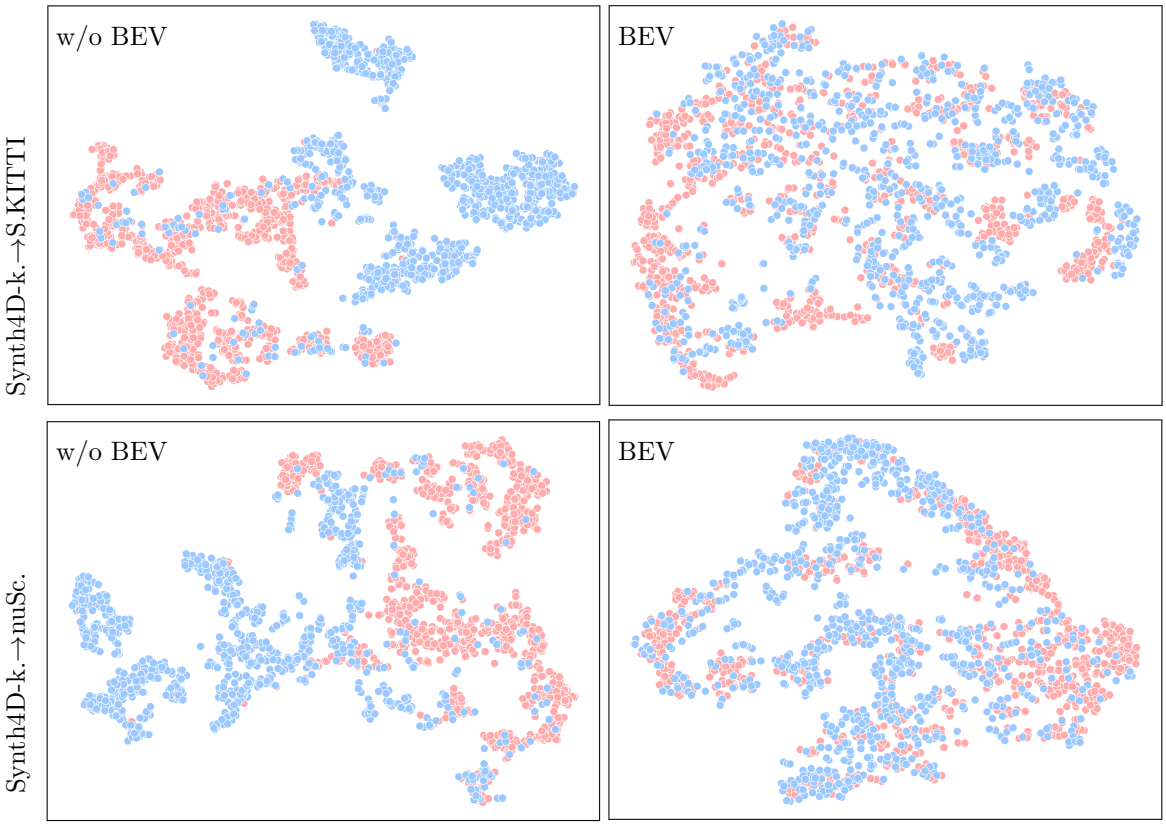


Figure 6.4: Feature visualization: t-SNE visualization [42] of the point embeddings for the *road* class, obtained by training our network without (w/o BEV, *left*) and with BEV task (BEV, *right*). *Top:* Synth4D-KITTI→SemanticKITTI. *Bottom:* Synth4D-KITTI→nuScenes. As shown, projected features are well-aligned when training the network with the auxiliary BEV task.

space that is less sensitive to individual sensor characteristics: in the BEV space scans from different domains look *more alike*, Fig. 6.3. We effectively regularize the model training by promoting the network robustness to changes in the sensor acquisition patterns. We validate this intuition in Fig. 6.4 by visualizing learned point embeddings [42] for the *road* class with and without BEV auxiliary task, across multiple domains. As can be seen, projected embeddings are consistently closer when employing BEV auxiliary task.

6.1.3 Experiments

In this section, we report our experimental evaluation. First, we describe our evaluation protocol, datasets and baselines. Second, we provide implementation details. Then, we report our *synth-to-real* and *real-to-real* results, and discuss qualitative results. Finally, we ablate our proposed approach.

Experimental protocol

We base our evaluation protocol on two synthetic (Synth4D [154]) and two real-world datasets (SemanticKITTI [10] and nuScenes [48]). We study (i) generalization from *synthetic→real data* in *single-* and *multi-source* settings, *i.e.*, trained on one or multiple source datasets, as well as (ii) *real→real* generalization.

Synthetic datasets. Synth4D [154] is a synthetic dataset generated using CARLA [44] simulator. It provides two synthetic source domains, each one with $20k$ labeled scans, emulating Velodyne64HDLE and Velodyne32HDLE sensors, similar to those used to record SemanticKITTI (*Synth4D-KITTI*) and nuScenes-lidarseg datasets (*Synth4D-nuScenes*). These datasets were generated in the same synthetic environment, mimicking different sensor types. We use the official training splits in all experiments.

Real datasets. SemanticKITTI [10] was recorded in various regions of Karlsruhe, Germany, with Velodyne64HDLE sensor, and provides $20k$ labeled scans. nuScenes [48] was recorded in Boston, USA, and Singapore, with a sparser Velodyne32HDLE sensor and provides $40k$ labeled LiDAR scans. We use the official training and validation splits for both datasets in all experiments.

Class vocabulary. To ensure this task is well-defined, we formalize cross-dataset consistent and compatible semantic class vocabulary, which ensures

there is a one-to-one mapping between all semantic classes. We follow [154] which provides a common label space among Synth4D, SemanticKITTI and nuScenes-lidarseg consisting of seven semantic classes: `vehicle`, `person`, `road`, `sidewalk`, `terrain`, `manmade` and `vegetation`. We perform all evaluation with respect to these classes.

Baselines. We are the first in studying DG-LSS, therefore, we construct our set of baselines by considering the previous efforts in bridging the domain shift in other LiDAR tasks as well DG methods for images.

Data augmentation baselines (Aug). Data augmentations are used in supervised learning to improve model generalization and, in DA, to reduce domain shift. We re-implement and evaluate in the DG context three data augmentation approaches: Mix3D [123], PointCutMix [230] and CoSMix [153]. Mix3D concatenates both points and labels of different scenes, obtaining a single mixed-up scene. PointCutMix and CoSMix follow the same strategy but mix either patches or semantic regions, respectively. We follow the official implementations and apply these data augmentation strategies during source training. For single-source training, we mix between samples of the same domain, while for multi-source training, we mix between samples of the two source domains.

Image-based baselines (2D DG). Image-based DG is a widely explored field. However, not all the methods can be extended to DG-LSS due to their image-related assumptions, *e.g.*, style consistency. We identify and re-implement two image-based approaches that can be extended to DG-LSS: IBN [125] and RobustNet [26]. IBN makes use of both batch and instance normalization layers in the network blocks to boost generalization performance. We follow the IBN-C block scheme and use it to build the sparse model. RobustNet [26] makes use of IBN blocks and introduces an instance whitening loss. Similarly, we start from our IBN implementation and use their instance-relaxed whitening loss [26] during training.

UDA for LiDAR baselines (3D UDA). Most of the UDA for LSS approaches assume (unlabeled) target data available for training/adaptation [215], hence, we cannot adapt and evaluate them in the DG setting fairly. We identify two UDA baselines that assume only weak supervision from the target domain, *e.g.*, vehicle dimensions or sensor specs. We re-implement and extend SN [196] and RayCast [90]. SN [196] uses the knowledge of average vehicle dimensions from source and target domains to re-scale source instances. We employ DBSCAN to isolate vehicle instances in both source and target domains and estimate the ratio between their dimensions. At training time, we follow their pipeline and train our segmentation model on the re-scaled point clouds. RayCast [89] employs ray casting to re-sample source data by mimicking target sensor sampling. We use the official code to re-sample source data and use mapped data during training. Notice that both these baselines use a-priori knowledge from the target domain, *i.e.*, vehicle dimensions or target sensor specifications, and can thus be considered as weakly-supervised baselines.

Implementation details

We implement our method and all the baselines by using the PyTorch framework. We use MinkowskiNet [27] as a sparse convolutional backbone in all our experiments and train until convergence with voxel size 0.05m, total batch size of 16, learning rate 0.01 and ADAM optimizer [82]. In the experiments, we use random rotation, scaling, and downsampling for better convergence of baselines and LiDOG. We set rotation bounds between $[-\pi/2, \pi/2]$, scaling between $[0.95, 1.05]$, and perform random downsampling for 80% of the patch points. We set projection bounds B^{3D} based on the input resolution. We set b_x and b_z to $50m$ in the denser source domains of Synth4D-KITTI and SemanticKITTI and b_x and b_z to $30m$ in the sparser domains of Synth4D-nuScenes and nuScenes. Quantization pa-

Table 6.1: Synth4D-KITTI→Real, single-source. Our approach (LiDOG) improves upon *Source* model on both real datasets: +19.49 mIoU for SemanticKITTI and +16.52 mIoU for nuScenes, outperforming all baselines. Lower bound (*red*): a model trained on the source domain without the help of DG techniques. Upper bound (*blue*): model directly trained on the target data.

S	T	Info	Method	Vehicle	Person	Road	Sidewalk	Terrain	Manmade	Vegetation	mIoU
Synth4D-KITTI	SemanticKITTI	Lower bound	Source	34.33	4.47	36.38	13.27	19.48	23.1	41.81	24.69
		Aug	Mix3D [123]	59.99	14.02	55.75	17.71	25.67	39.26	53.00	37.92
			PointCutMix [230]	58.26	14.35	67.74	13.91	**27.26**	**49.21**	**64.22**	42.14
			CoSMix [153]	43,01	14.16	61.84	12.38	18.84	18.00	57.56	32.26
		2D DG	IBN [125]	24.95	9.18	56.82	17.85	7.21	21.59	44.65	26.04
			RobustNet [26]	50.85	14.97	58.71	7.83	19.96	42.58	44.52	34.20
		3D UDA	SN [196]	49,38	14.83	68.53	18.45	25.62	37.49	59.48	39.11
			RayCast [90]	51.73	4.33	56.43	18.07	23.91	36.23	40.52	33.03
		3D DG	Ours (LiDOG)	**72.86**	**17.1**	**71.85**	**28.48**	11.53	46.02	61.42	**44.18**
		Upper bound	Target	81.57	23.23	82.09	57.64	46.84	64.14	75.21	61.53
Synth4D-KITTI	nuScenes	Lower bound	Source	19.99	6.80	32.85	6.81	11.95	45.07	20.86	20.62
		Aug	Mix3D [123]	26,8	**22.68**	41.90	7.31	13.54	48.17	52.09	30.36
			PointCutMix [230]	23,97	19.49	44.27	6.29	12.14	48.85	55.44	30.06
			CoSMix [153]	16.94	15.78	52.97	2.09	5.55	15.6	43.85	21.83
		2D DG	IBN [125]	21,32	11.96	36.83	5.39	11.25	35.91	37.46	22.87
			RobustNet [26]	26.19	7.47	43.85	2.29	13.93	43.63	46.32	26.24
		3D UDA	SN [196]	23.14	14.08	51.60	11.38	14.02	46.91	50.83	30.28
			RayCast [90]	19.54	11.78	56.53	6.66	8.45	45.66	36.99	26.52
		3D DG	Ours (LiDOG)	**31.29**	19.62	**64.66**	**14.21**	**15.63**	**57.3**	**57.27**	**37.14**
		Upper bound	Target	47.42	20.18	80.89	37.02	34.99	64.27	54.67	48.49

rameters x_q and z_q are computed based on the spatial bounds in order to obtain BEV labels of $168x168$ pixels. For downsampling dense features, we use a max pooling layer with window size 5, stride 3, and padding 1. The LiDOG 2D decoder is implemented with a series of three 2D convolutional layers interleaved by batch normalization layers. ReLU activation function is used on all the layers while softmax activation is applied on the last layer.

Table 6.2: Synth4D-nuScenes→Real, single-source. We train our model on Synth4D-nuScenes and test on SemanticKITTI and nuScenes. LiDOG improves over the *source* models by +15.08 mIoU on SemanticKITTI and by +9.21 mIoU on nuScenes. LiDOG outperforms all the compared baselines. Lower bound (*red*): a model trained n the source domain without the help of DG techniques. Upper bound (*blue*): a model directly trained on target data.

S	T	Info	Method	Vehicle	Person	Road	Sidewalk	Terrain	Manmade	Vegetation	mIoU
Synth4D-nuScenes	SemanticKITTI	Lower bound	Source	14.54	2.41	32.78	14.86	6.4	30.89	36.07	19.71
		Aug	Mix3D [123]	37.38	7.26	56.90	21.06	10.60	34.46	53.79	31.64
			PointCutMix [230]	27.51	4.32	56.04	21.37	7.02	24.38	45.35	26.57
			CoSMix [153]	16.13	6.42	39.71	14.63	13.04	23.54	30.57	20.58
		2D DG	IBN [125]	47.15	7.81	50.74	4.51	15.20	29.53	51.35	29.47
			RobustNet [26]	21.19	8.56	44.46	10.80	15.06	11.95	30.59	20.37
		3D UDA	SN [196]	11.56	2.38	37.50	10.86	5.19	20.70	39.23	18.20
			RayCast [90]	28,89	6.34	53.59	12.94	**15.86**	21.74	41.85	25.89
		3D DG	Ours	**55.08**	**11.42**	**59.48**	**26.10**	2.78	**34.83**	**53.82**	**34.79**
		Upper bound	Target	81.57	23.23	82.09	57.64	46.84	64.14	75.21	61.53
Synth4D-nuScenes	nuScenes	Lower bound	Source	17.73	8.94	36.53	7.28	5.78	52.13	43.62	24.57
		Aug	Mix3D [123]	**33.83**	17.85	47.74	8.93	9.71	56.35	44.18	31.23
			PointCutMix [230]	21.51	13.12	53.72	8.86	8.45	54.83	48.81	29.90
			CoSMix [153]	22.00	15.43	57.40	8.86	9.08	56.24	47.16	30.88
		2D DG	IBN [125]	23.07	14.13	44.96	7.10	9.83	53.70	**49.49**	28.90
			RobustNet [26]	21.59	12.29	48.52	8.14	6.22	51.33	47.68	27.97
		3D UDA	SN [196]	24.71	8.45	5.,08	5.66	**11.04**	47.00	39.05	26.57
			RayCast [90]	19.65	12.24	58.08	7.58	9.71	46.43	41.37	27.86
		3D DG	Ours (LiDOG)	26.79	**18.68**	**63.28**	**15.81**	6.57	**58.8**	46.5	**33.78**
		Upper bound	Target	47.42	20.18	80.89	37.02	34.99	64.27	54.67	48.49

Synth → Real evaluation

In this section, we train our model on one or more synthetic source domains and evaluate on both real target domains, *i.e.*, SemanticKITTI and nuScenes-lidarseg. We report two models as lower and upper bounds in all the tables for reference: *source* (*i.e.*, a model trained on the source domain without the help of DG techniques) and *target* (*i.e.*, model directly trained on the target data).

Single-source. In Tab. 6.1, we report the results for single-source *Synth4D-KITTI→Real*. We observe a 36.84 mIoU gap (SemanticKITTI) between

the *source* (24.69 mIoU) and *target* (61.53 mIoU) models. Among the baselines, augmentation-based methods are the most effective: PointCutMix and Mix3D achieve 42.14 mIoU and 30.36 mIoU on *Synth4D-KITTI→Real*, respectively. Surprisingly, these approaches outperform 3D DA baselines, that leverage prior information about the target domain, *e.g.*, vehicle dimensions (SN) or target sensor specs (RayCast), confirming the efficacy of data augmentations for DG. LiDOG significantly reduces the domain gap between *source* and *target* models and outperforms all the compared baselines in all the scenarios. For example, on *Synth4D-KITTI→Real* (Tab. 6.1), we obtain 44.18 mIoU, a +19.49 improvement over the *source* model. In Tab. 6.2, we report the results for single-source *Synth4D-nuScenes → Real*. We observe a 41.82 mIoU gap (SemanticKITTI) between the *source* (19.71 mIoU) and *target* (61.53 mIoU) models on Synth4D-nuScenes → SemanticKITTI and a 23.92 mIoU gap (nuScenes), between *source* (24.57 mIoU) and *target* (48.49 mIoU) on Synth4D-nuScenes→nuScenes. Among the baselines, augmentation-based methods are the most effective, with Mix3D achieving 31.64 mIoU (SemanticKITTI) and 31.23 mIoU (nuScenes). LiDOG reduces the domain gap between *source* and *target* models and outperforms all the compared baselines in all the scenarios. For example, on *Synth4D-nuScenes→Real* (Tab. 6.2), we obtain 34.79 mIoU, a +15.08 improvement over the *source* model.

Multi-source. In Tab. 6.3, we report the results for the multi-source training, where we train our model on both synthetic datasets, and evaluate the model on both real datasets. Compared to the single source (Tab. 6.1), in this setting, we train models on synthetic data that mimic two different sensor types. The *source* model improves from 24.69 → 31.82 mIoU on SemanticKITTI and from 20.62 → 25.60 mIoU on nuScenes as a result of training on multiple synthetic datasets. As can be seen, LiDOG consistently improves over top-scoring baselines, RobustNet (+3.04 mIoU)

Table 6.3: (Synth4D-nuScenes + Synth4D-KITTI)→Real, multi-source. Baselines significantly improve performance relative to the *source* model. Specifically, with LiDOG we observe +10.62 mIoU improvement on SemanticKITTI and +14.63 mIoU on nuScenes. Our approach (LiDOG) outperforms all the compared approaches. Lower bound (*red*): a model trained on the source domain without the help of DG techniques. Upper bound (*blue*): model directly trained on the target data.

S	T	Info	Method	Vehicle	Person	Road	Sidewalk	Terrain	Manmade	Vegetation	mIoU
Synth4D-nuScenes+Synth4D-KITTI	SemanticKITTI	Lower bound	Source	45.14	8.39	41.74	17.28	18.44	33.81	57.93	31.82
		Aug	Mix3D [123]	45.64	7.72	59.11	22.12	16.86	**50.03**	51.59	36.15
			PointCutMix [230]	48.38	6.28	60.8	20.53	7.56	33.62	54.32	33.07
			CoSMix [153]	45.81	0.93	59.28	22.81	18.13	43.14	40.91	33.00
		2D DG	IBN [125]	58.9	12.02	67.09	27.89	7.53	33.91	57.62	37.85
			RobustNet [26]	59.28	13.11	66.55	7.75	**30.87**	35.46	**62.75**	39.40
		3D UDA	SN [196]	35.07	5.95	60.36	**27.96**	13.3	26.88	54.88	32.06
			RayCast [90]	58.06	3.3	63.41	25.33	22.03	35.39	41.64	35.59
		3D DG	Ours (LiDOG)	**66.23**	**18.87**	**67.39**	24.13	15.22	46.21	59.03	**42.44**
		Upper bound	Target	81.57	23.23	82.09	57.64	46.84	64.14	75.21	61.53
Synth4D-nuScenes+Synth4D-KITTI	nuScenes	Lower bound	Source	20.64	13.4	28.44	7.61	10.23	52.88	46.01	25.60
		Aug	Mix3D [123]	28.35	24.6	57.16	13.96	9.48	60.19	55.98	35.67
			PointCutMix [230]	25.6	14.8	54.29	11.58	6.66	58.09	51.53	31.79
			CoSMix [153]	23.13	8.26	52.24	11.04	10.88	58.29	53.43	31.04
		2D DG	IBN [125]	26.11	16.95	55.16	13.59	12.67	56.64	52.92	33.43
			RobustNet [26]	27.03	16.48	50.61	6.66	15.73	57.68	**56.47**	32.95
		3D UDA	SN [196]	19.08	13.89	53.18	14.73	9.69	53.54	45.39	29.93
			RayCast [90]	25.37	9.03	60.27	10.03	11.15	54.72	45.56	30.87
		3D DG	Ours (LiDOG)	**30.78**	**25.3**	**73.2**	**19.83**	**16.03**	**62.65**	53.80	**40.23**
		Upper bound	Target	47.42	20.18	80.89	37.02	34.99	64.27	54.67	48.49

and Mix3D (+4.56 mIoU). On average, LiDOG significantly improves over the *source* model with +10.62 mIoU and +14.63 mIoU on SemanticKITTI and nuScenes, respectively. We conclude that LiDOG is consistently a top-performer for classes and scenarios over the *source* model, except for the *terrain* class. This may be a limitation of our BEV projection. We assume it occurs when multiple classes are spatially overlapping due to the top-down projection, *e.g.*, mixing *terrain* with *vegetation*.

Table 6.4: SemanticKITTI→nuScenes, single-source. We train our model on SemanticKITTI and evaluate it on the nuScenes dataset. LiDOG improves over the *source* model by +8.35 mIoU. Lower bound (*red*): a model trained on the source domain without the help of DG techniques. Upper bound (*blue*): model directly trained on the target data.

S	T	Info	Method	Vehicle	Person	Road	Sidewalk	Terrain	Manmade	Vegetation	mIoU
SemanticKITTI	nuScenes	Lower bound	Source	22.92	0.03	63.33	16.09	7.42	35.40	40.53	26.53
		Aug	Mix3D [123]	33.74	11.22	58.54	12.91	5.28	**50.36**	48.59	31.52
			PointCutMix [230]	22.75	2.68	59.37	10.47	7.04	27.9	42.74	24.71
			CoSMix [153]	**35.91**	0.00	58.13	11.57	8.95	45.17	49.11	29.83
		2D DG	IBN [125]	29.93	0.01	56.77	18.70	12.09	37.67	33.83	27.00
			RobustNet [26]	25.50	0.03	62.40	15.12	9.40	29.58	43.62	26.52
		3D UDA	SN [196]	21.35	0.01	60.48	15.06	6.16	31.85	45.69	25.80
			RayCast [90]	28.82	0.00	59.25	16.08	12.51	49.72	49.82	30.89
		3D DG	Ours (LiDOG)	23.97	**14.86**	**70.63**	**24.59**	**13.97**	45.27	**50.85**	**34.88**
		Upper bound	Target	47.42	20.18	80.89	37.02	34.99	64.27	54.67	48.49

Real → Real evaluation

In this evaluation, we only train and evaluate our models on real-world recordings. We report *SemanticKITTI→nuScenes* results in Tab. 6.4 and *nuScenes→SemanticKITTI* results in Tab. 6.5. Overall, we observe that the performance gap of *Real→Real* (Tab. 6.4-6.5) is lower compared to the gap in *Synth→Real* setting (Tab. 6.1-6.3).

SemanticKITTI→nuScenes. We observe a performance gap of 21.96 mIoU between *source* (26.53 mIoU) and *target* (48.49 mIoU) models. All the baselines consistently improve over the *source* model, however, by a smaller margin as compared to *Synth→Real*. Again, LiDOG improves on all the classes compared to the *source* performance and achieves top generalization performance on all the classes except for *vehicle* and *manmade*. On average, LiDOG obtains 34.88 mIoU and obtains +8.35 mIoU improvement over the *source* model.

nuScenes→SemanticKITTI. We observe a performance gap of 31.98 mIoU

Table 6.5: nuScenes→SemanticKITTI, single-source. We train our model on nuScenes and evaluate it on the SemanticKITTI dataset. LiDOG improves over the *source* model by +11.67 mIoU. Lower bound (*red*): a model trained on the source domain with-out the help of DG techniques. Upper bound (*blue*): model directly trained on the target data.

S	T	Info	Method	Vehicle	Person	Road	Sidewalk	Terrain	Manmade	Vegetation	mIoU
nuScenes	SemanticKITTI	Lower bound	Source	27.16	6.12	29.19	8.75	22.87	51.27	61.47	29.55
		Aug	Mix3D [123]	37.86	6.74	41.95	5.73	27.59	41.21	65.41	32.36
			PointCutMix [230]	55.22	**13.92**	45.96	5.43	30.47	**56.50**	**70.98**	39.78
			CoSMix [153]	44.58	13.88	36.10	10.19	29.32	54.43	69.08	36.80
		2D DG	IBN [125]	22.00	11.32	37.24	0.21	13.11	21.76	50.33	22.28
			RobustNet [26]	32.94	10.98	39.85	14.70	28.27	50.42	58.47	33.66
		3D UDA	SN [196]	25.69	5.46	19.59	2.17	23.47	27.65	61.07	23.58
			RayCast [90]	28.30	16.09	45.80	9.44	20.56	38.56	61.83	31.51
		3D DG	Ours (LiDOG)	**60.07**	9.03	**47.44**	**16.40**	**32.58**	54.21	68.82	**41.22**
		Upper bound	Target	81.57	23.23	82.09	57.64	46.84	64.14	75.21	61.53

with *source* and *target* models achieving 29.55 mIoU and 61.53 mIoU, respectively. Augmentation-based methods are once again the most effective baselines, with PointCutMix achieving 39.78 mIoU. LiDOG outperforms all the compared baselines, achieves 41.22 mIoU and obtains +11.67 over the *source* model.

6.1.4 Ablation studies

BEV auxiliary task

We study whether LiDOG generalization capabilities are due to the additional segmentation head, or specifically due to the dense BEV segmentation head. The use of multiple prediction heads for ensembling is a well-known practice for improving prediction robustness [19, 47]. To this purpose, we remove the lower branch of LiDOG and attach an additional 3D branch. The resulting architecture has two sparse 3D heads h^{3D} taking as input F^{3D}. During inference, predictions are obtained by averaging the predictions from both heads. Fig. 6.5 reports the generalization perfor-

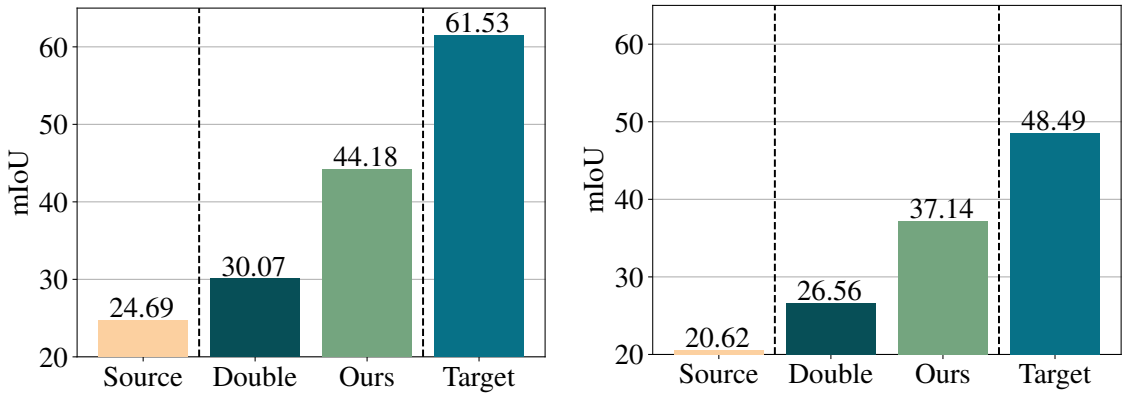


Figure 6.5: Effectiveness of the BEV head: We compare 2D BEV decoder (*Ours*) to simply adding an additional 3D segmentation head (*Double*) on SemanticKITTI (*left*) and nuScenes (*right*). Source: $Synth4D - KITTI$.

mance of this alternate version (*Double*) compared to LiDOG (*Ours*). As can be seen, by simply learning two decoders and ensembling predictions we obtain lower performance as compared to utilizing the proposed BEV network heads to improve the generalization.

BEV prediction area

First, we study the impact on the generalization ability of LiDOG w.r.t. BEV area size in Synth4D-KITTI→Real setting. We experiment with projection bounds B^{3D} ranging from $10x10$ to $60x60m$. As can be seen in Fig. 6.6, there is a near-linear relation between the BEV area and generalization performance.

BEV resolution

We study the impact of BEV image resolution on the LiDOG performance. We adapt the feature resolution by applying several pooling steps and the label resolution via the quantization step size. In Fig. 6.7, we report the results obtained on Synth4D-KITTI→Real, when using BEV features with 50% and 75% of the initial resolution (100%). Interestingly, on Synth4D-KITTI→nuScenes, we observe a slight improvement with 75% resolution.

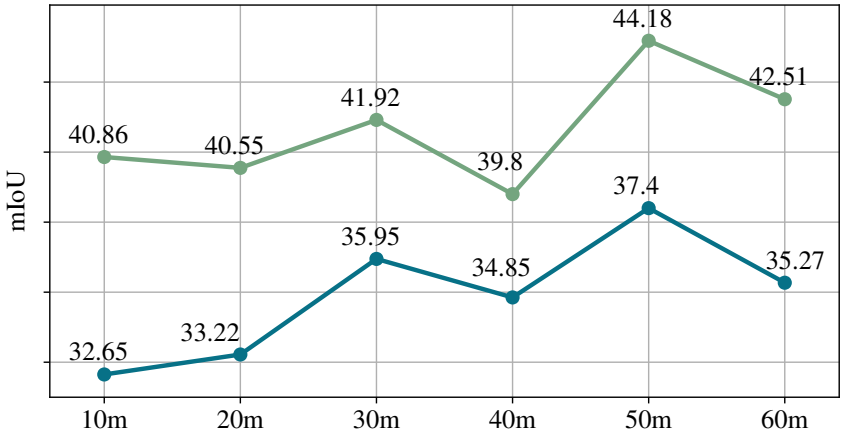


Figure 6.6: BEV prediction area: We study the impact of BEV area size on SemanticKITTI (*top*) and nuScenes (*bottom*). $50x50m$ area is consistently the best-performing option on both datasets. Source: $Synth4D - KITTI$.

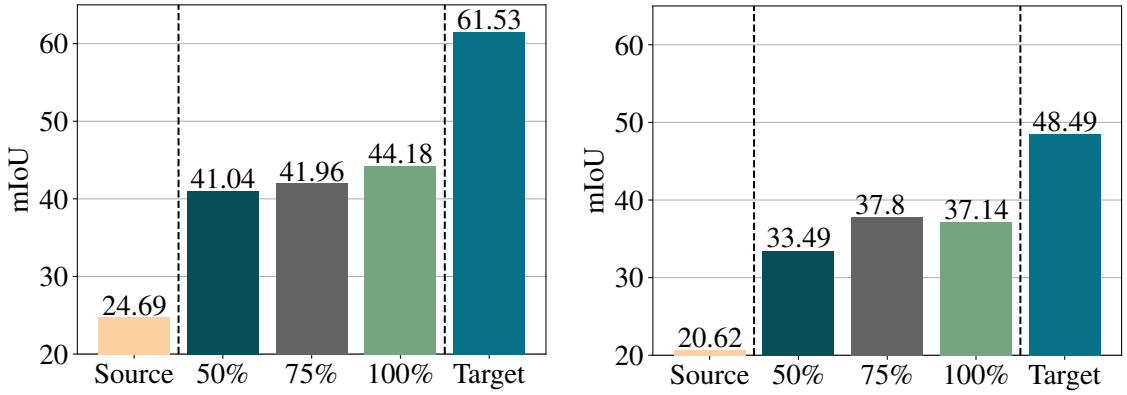


Figure 6.7: BEV image resolution: We compare the performance while changing the BEV image resolution on SemanticKITTI (*left*) and nuScenes (*right*), Source: *Synth4D-KITTI*.

However, we obtain consistently top performance on both domains with the full resolution (100%).

Qualitative comparison with Mix3D

In Fig. 6.8, we discuss qualitative results, comparing the *source* model, the top-performing baseline, Mix3D, our proposed LiDOG, and the ground-truth labels. As we can see, the *source* model predictions are often incorrect and mingled in many small and large regions. Mix3D brings (limited)

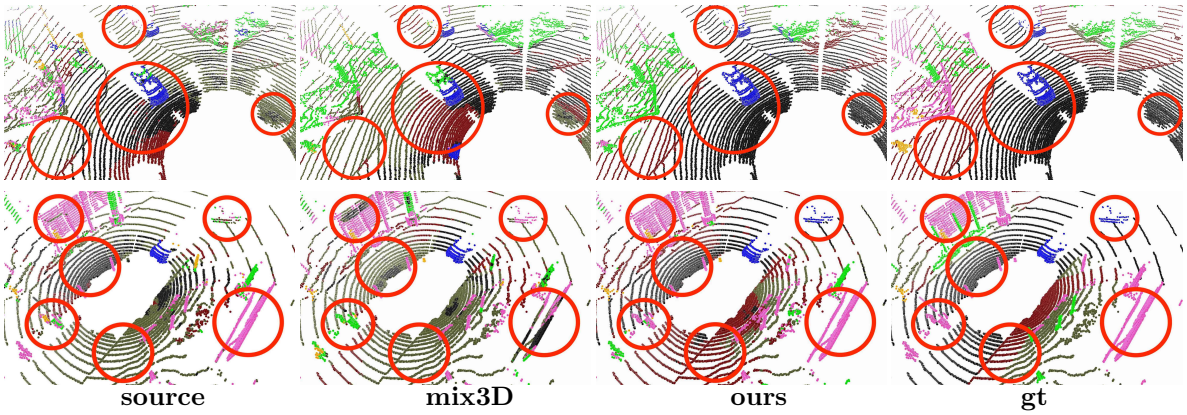


Figure 6.8: Qualitative results. *Top:* Synth4D-KITTI→SemanticKITTI, *bottom:* Synth4D-KITTI→nuScenes. LiDOG improves consistently improves results over the *source* model and outperforms Mix3D [123] with more homogeneous predictions.

improvements. For example, notice how *road* (Fig. 6.8 *top*) is incorrectly segmented. LiDOG achieves overall top performance, improving on both small (*vehicle*) and large (*building* and *road*) areas.

Additional qualitative results

We report additional qualitative results for each baseline and in all the studied generalization directions. In Fig. 6.9-6.11 we show qualitative results in the Synth→Real setting: Synth4D-kitti→Real (Fig. 6.9), Synth4D-nuScenes→Real (Fig. 6.10) and Synth4D-kitti+Synth4D-nuScenes→Real (Fig. 6.11). In Fig. 6.12-6.13 we show qualitative results in the Semantic KITTI → nuScenes and nuScenes→SemanticKITTI directions, respectively. Source predictions are often incorrect and spatially inconsistent. Baselines consistently improve over the source model performance. LiDOG achieves the overall best performance with improved and more precise predictions. This can be seen in all the reported results, both *synth→real* and *real→real*.

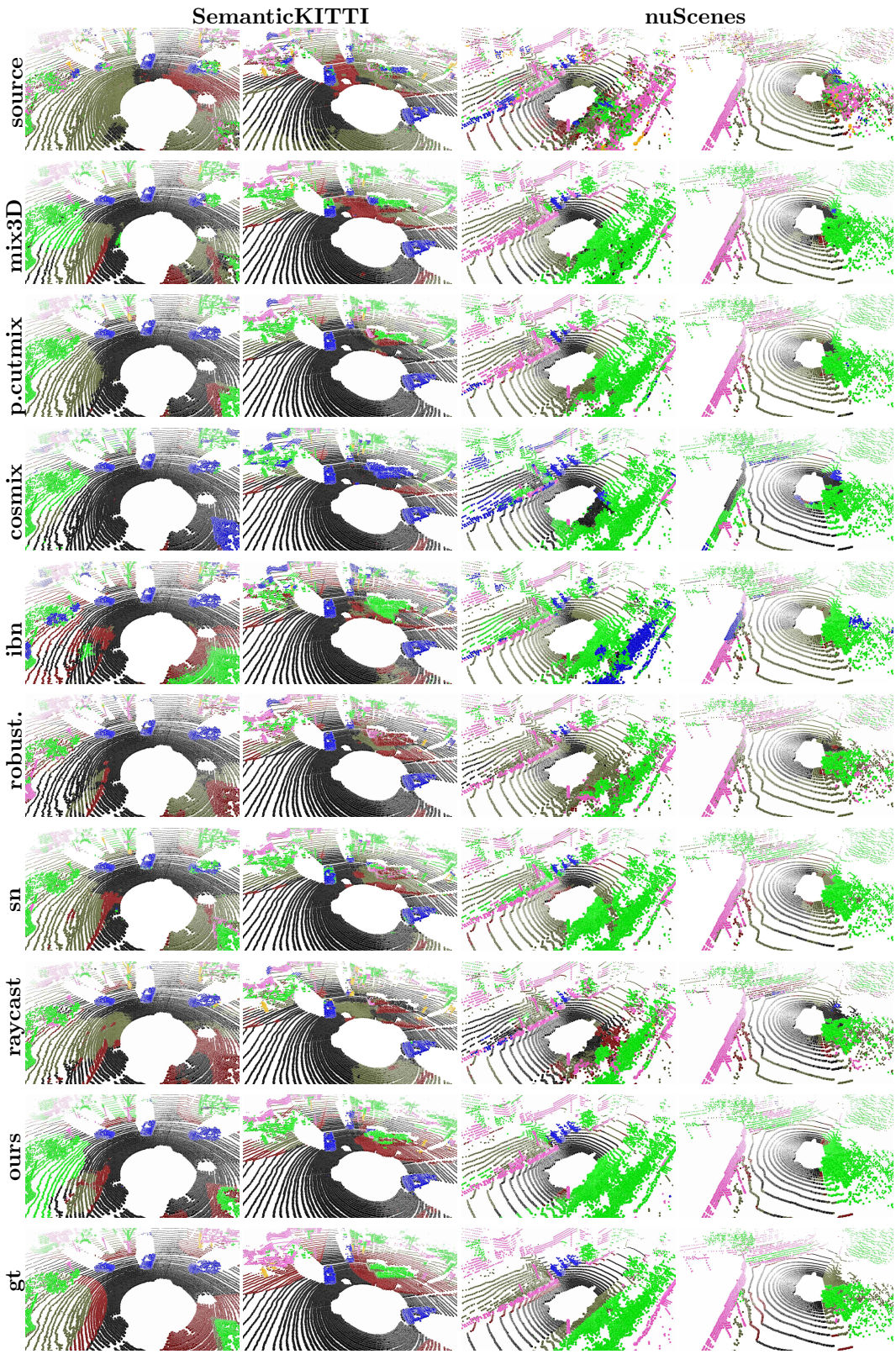


Figure 6.9: Qualitative results. *Left:* Synth4D-kitti→SemanticKITTI, *right:* Synth4D-kitti→nuScenes.

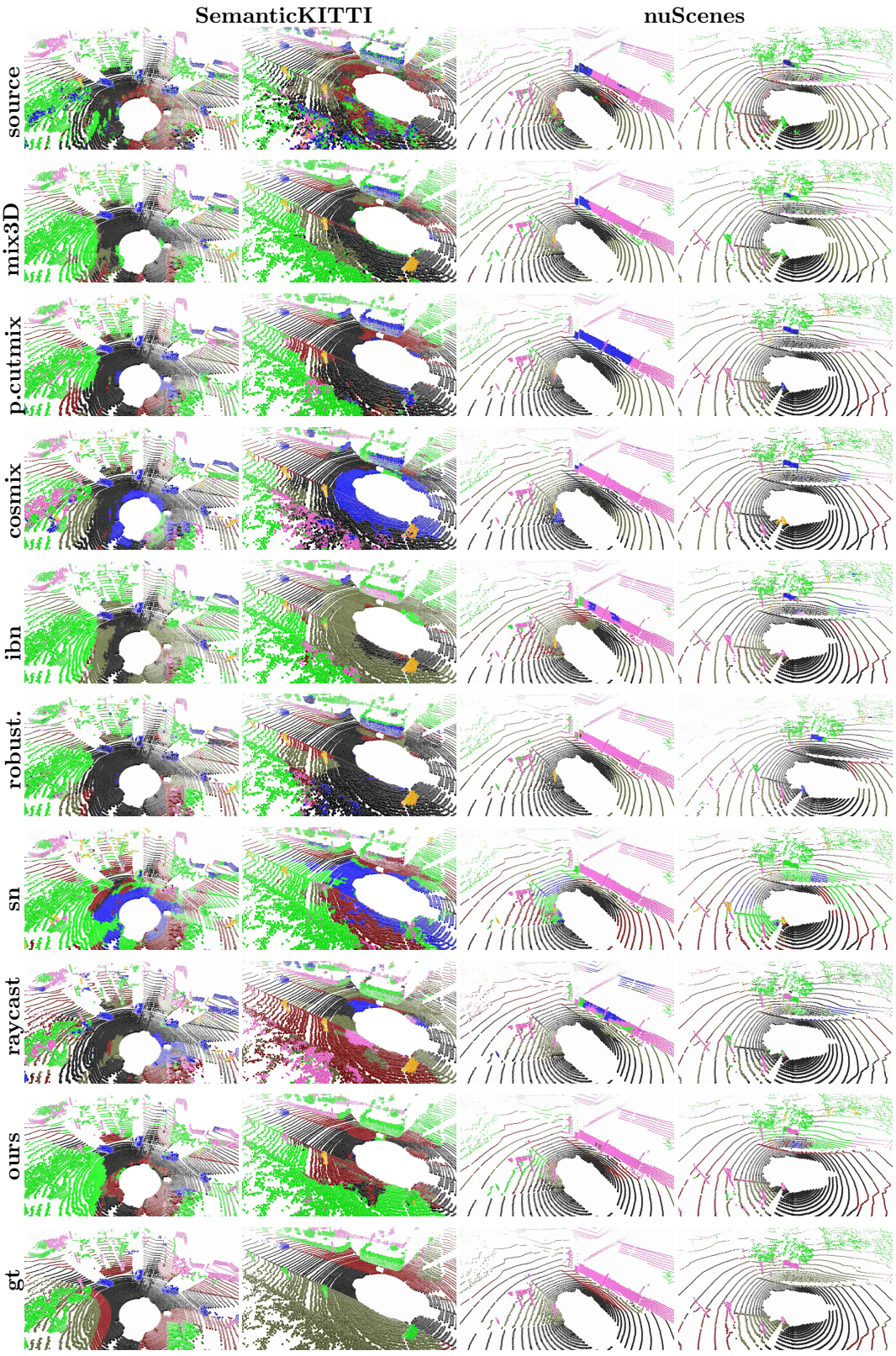


Figure 6.10: Qualitative results. *Left:* Synth4D-nuScenes→SemanticKITTI, *right:* Synth4D-nuScenes→nuScenes.

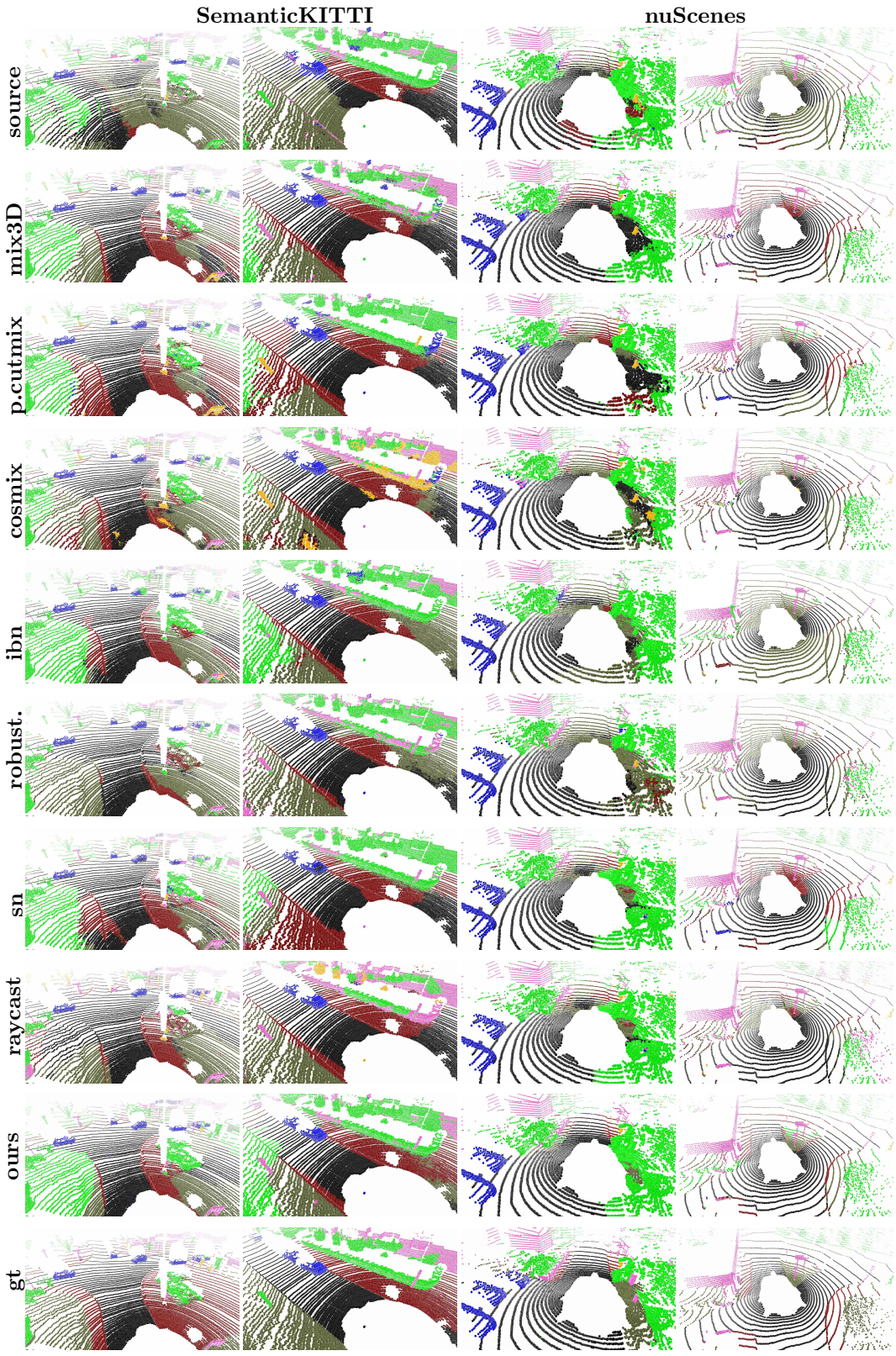


Figure 6.11: Qualitative results. *Left:* Synth4D-kitti + Synth4D-nuScenes → SemanticKITTI, *right:* Synth4D-kitti + Synth4D-nuScenes → nuScenes.

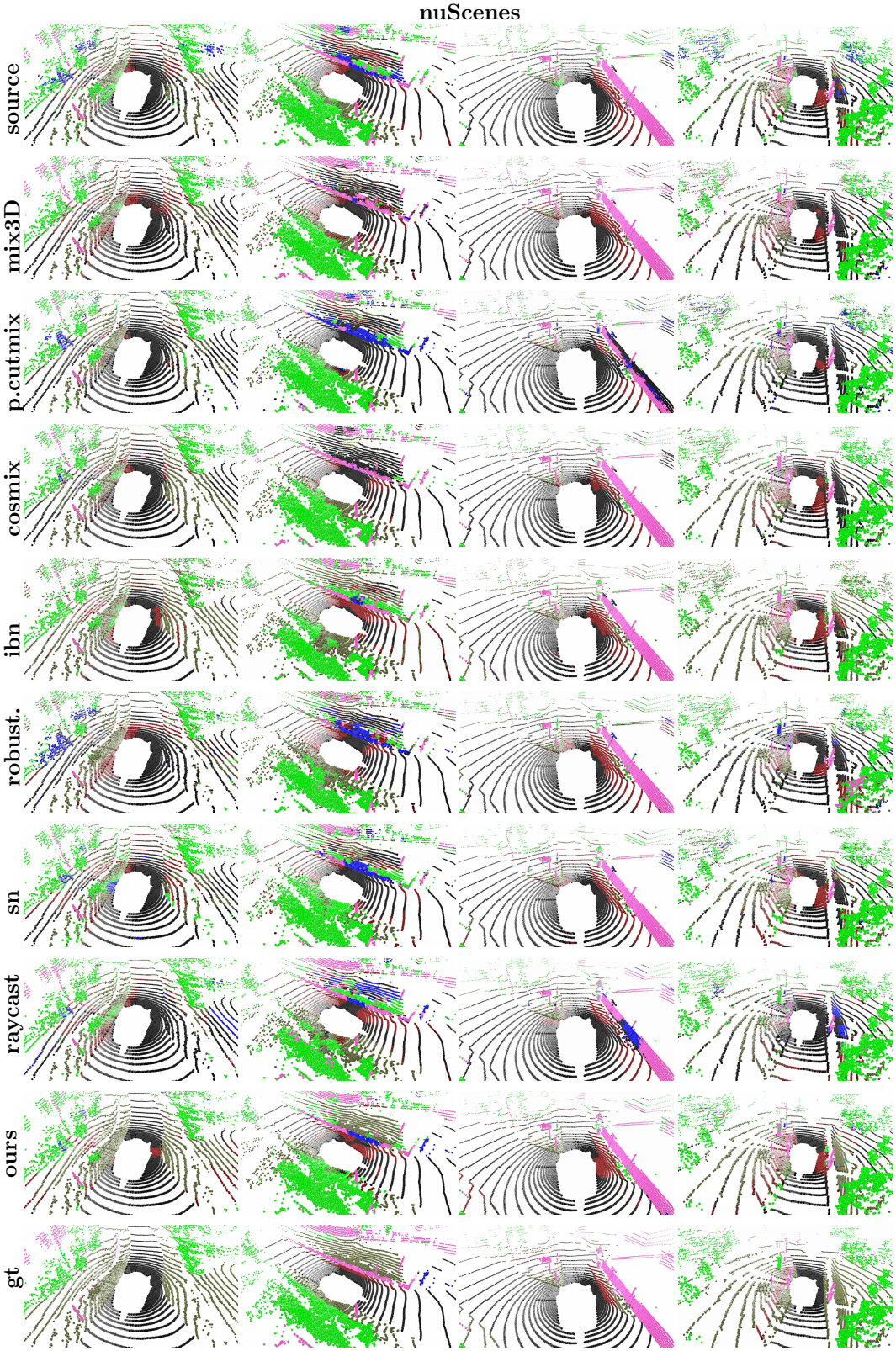


Figure 6.12: Qualitative results. SemanticKITTI→nuScenes.

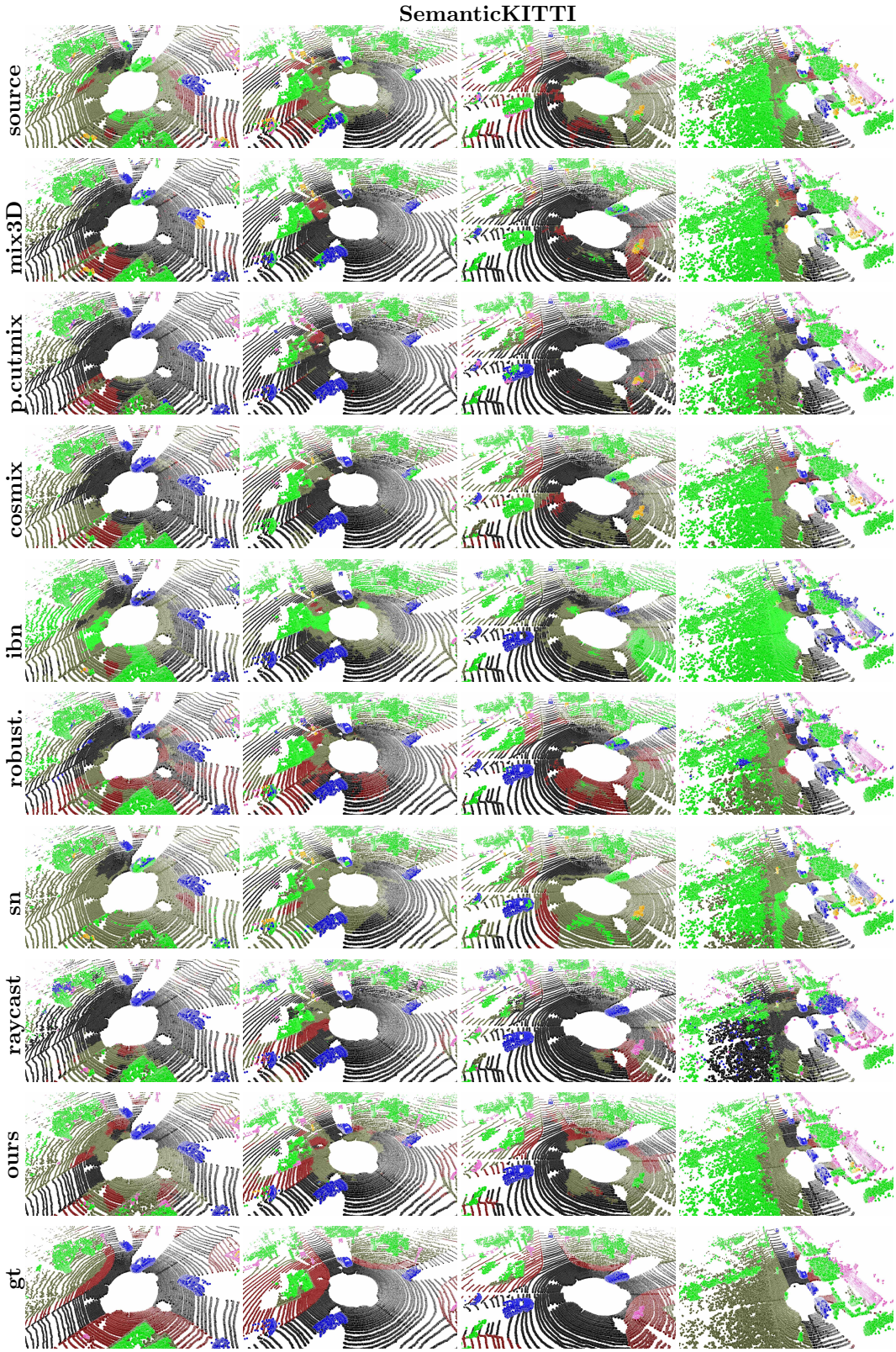


Figure 6.13: Qualitative results. nuScenes→SemanticKITTI.

Chapter 7

Conclusions

In this thesis, we addressed the challenge of domain shift in 3D point cloud perception, exploring mitigation strategies across five distinct settings that vary in target supervision, target data availability, and source availability. Our study ranged from semi-supervised and unsupervised domain adaptation to source-free and online adaptation to domain generalization.

In particular, Chapter 3 focused on domain shift with annotated source and partially annotated or unlabeled target domains. We introduced a novel mixing-based strategy for point cloud domain adaptation in these settings. In Chapter 4, we presented the first study on source-free adaptation, mitigating domain shift from a pre-trained source model without access to source data during adaptation. Our approach employed pseudo-annotations, reversible scale transformations, and motion coherency. Chapter 5 and Chapter 6 delved into the novel settings of online target data and no target data, respectively. In Chapter 5, we introduced the pioneering study of source-free online unsupervised domain adaptation for point clouds and the first method for this setting leveraging pseudo-annotations and geometric feature propagation. Additionally, we introduced a synthetic dataset for 3D semantic segmentation simulating widely used LiDAR sensors. Finally, in Chapter 6, we explored handling domain shift towards

unknown target domains. After studying existing work in related fields, we proposed the first approach for domain generalization in point clouds, using an auxiliary dense BEV task.

7.1 Future directions

The progress in methods for domain adaptation and domain generalization for 3D perception has been driven by assumptions that facilitate the study of this issue, yet these assumptions underlie the main limitations of these approaches.

Current approaches rely on the close-set class assumption, assuming the same label space in source and target domains. While practical in controlled scenarios, this assumption is challenging to verify in real-world perception. A first promising research direction is the development of adaptation or generalization approaches between different label spaces across domains. Addressing domain shift while recognizing diverse or novel classes presents a valuable opportunity for real-world applications. While novel class discovery in 3D semantic segmentation has gained attention [140], existing approaches focus on the same domain. Consequently, the challenge of handling domain shift with different label spaces in 3D perception still needs to be explored.

A second promising direction is the study of label shift in recent open-vocabulary 3D perception methods. Recent advances in multi-modal learning [137] have demonstrated extraordinary results, unlocking the potential of diverse modalities for open vocabulary perception [127, 174]. However, their behavior across different domains remains relatively unexplored, presenting a promising direction for future studies. These methods typically leverage self-supervised pre-training with image and text modalities, wherein semantic concepts are implicitly learned from the text modality

and inherited post-distillation into the 3D modality [127, 174]. Nevertheless, the identification of seen and unseen classes and the impact on open-vocabulary and out-of-domain robustness are yet to be explored comprehensively.

A third promising research direction involves the use of multiple modalities during adaptation or generalization training. Connected with multimodal learning, the inclusion of additional modalities in the adaptation process has shown potential for enhancing adaptation performance and generalization capabilities, as evidenced by some recent works [74]. However, the exploration of how these additional modalities can improve generalization or adaptation in the context of 3D perception remains relatively unknown.

In conclusion, existing domain adaptation or generalization approaches need a comprehensive exploration of domain shift among different input representations. Specifically, methods using projection-based architectures often avoid the inclusion of point or voxel-based architectures and vice versa. This creates a significant gap, as certain phenomena and domain shift causes may be representation-dependent, presenting an opportunity for cross-representation learning. Our work in [156] took a step in this direction by introducing BEV maps to enhance generalization robustness. Nevertheless, the inclusion of other point cloud representations during training and the investigation of domain shift among input representations remains an interesting and unexplored direction for future research.

Bibliography

[1] A.Geiger, P.Lenz, C.Stiller, and R.Urtasun. Vision meets robotics: The kitti dataset. *IJRR*, 2013.

[2] I. Alonso, L. Riazuelo, L. Montesano, and A.C. Murillo. 3d-mininet: Learning a 2d representation from point clouds for fast and efficient 3d lidar semantic segmentation. In *IROS*, 2020.

[3] N. Alyshai. Machine learning 101: Supervised, unsupervised, reinforcement learning explained. `https://datasciencedojo.com/blog/machine-learning-101/`, 2023.

[4] S. Ao, Q. Hu, B. Yang, A. Markham, and Y. Guo. SpinNet: Learning a General Surface Descriptor for 3D Point Cloud Registration. In *CVPR*, 2021.

[5] A.Torralba and A.A.Efros. Unbiased look at dataset bias. In *CVPR*, 2011.

[6] M. Aygun, A. Osep, M. Weber, M. Maximov, C. Stachniss, J. Behley, and L. Leal-Taixé. 4d panoptic lidar segmentation. In *CVPR*, 2021.

[7] Y. Balaji, S. Sankaranarayanan, and R. Chellappa. Metareg: Towards domain generalization using meta-regularization. *NeurIPS*, 2018.

[8] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, J. Gall, and C. Stachniss. Towards 3d lidar-based semantic scene understanding of 3d point cloud sequences: The semantickitti dataset. *IJRL*, 2021.

[9] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *ICCV*, 2019.

[10] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *ICCV*, 2019.

[11] N. Bisagno, C. Saltori, B. Zhang, FGB. De Natale, and N. Conci. Embedding group and obstacle information in lstm networks for human trajectory prediction in crowded scenes. *CVIU*, 2021.

[12] C. M. Brown. Computer vision and natural constraints. *Science*, 1984.

[13] S. Bucci, A. D'Innocente, Y. Liao, F.M. Carlucci, B. Caputo, and T. Tommasi. Self-supervised learning across domains. *T-PAMI*, 2021.

[14] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual kitti 2. *arXiv*, 2020.

[15] H. Caesar, V. Bankiti, A.H. Lang, S. Vora, V.E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[16] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[17] F.M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi. Domain generalization by solving jigsaw puzzles. In *CVPR*, 2019.

[18] F.M. Carlucci, L. Porzi, B. Caputo, E. Ricci, and S. Rota Bulo. Autodial: Automatic domain alignment layers. In *ICCV*, 2017.

[19] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *NeurIPS*, 2020.

[20] N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *T-IT*, 2004.

[21] C. Chen, Z. Fu, Z. Chen, S. Jin, Z. Cheng, X. Jin, and X.-S. Hua. Homm: Higher-order moment matching for unsupervised domain adaptation. In *AAAI*, 2020.

[22] LC. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and AL. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *ICLR*, 2014.

[23] X. Chen and K. He. Exploring simple siamese representation learning. In *CVPR*, 2021.

[24] Y. Chen, S. Liu, X. Shen, and J. Jia. Fast point r-cnn. In *ICCV*, 2019.

[25] Z. Chen, Y. Luo, Z. Wang, M. Baktashmotlagh, and Z. Huang. Revisiting domain-adaptive 3d object detection by reliable, diverse, and class-balanced pseudo-labeling. In *ICCV*, 2023.

[26] S. Choi, S. Jung, H. Yun, J.T. Kim, S. Kim, and J. Choo. Robustnet: Improving domain generalization in urban-scene segmentation via instance selective whitening. In *CVPR*, 2021.

[27] C. Choy, JY. Gwak, and S. Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019.

[28] C. Choy, J. Park, and V. Koltun. Fully convolutional geometric features. In *ICCV*, 2019.

[29] C.Liu, S.Lee, S.Varnhagen, and H.E.Tseng. Path planning for autonomous vehicles using model predictive control. In *IV*, 2017.

[30] T. Cortinhal, G. Tzelepis, and E. E. Aksoy. Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds. In *ISVC*, 2020.

[31] G. Csurka. Domain adaptation for visual applications: A comprehensive survey. *arXiv*, 2017.

[32] A. Dai, A. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017.

[33] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *ICCV*, 2017.

[34] X. Danfei, A. Dragomir, and J. Ashesh. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *CVPR*, 2018.

[35] D.Davies and D.W.Bouldin. A cluster separation measure. *T-PAMI*, 1979.

[36] D.Dolgov, S.Thrun, M.Montemerlo, and J.Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *IJRR*, 2010.

[37] R. DeBortoli, L. Fuxin, A. Kapoor, and G. A. Hollinger. Adversarial training on point clouds for sim-to-real 3d object detection. *RAL*, 2021.

[38] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza. Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset. In *ICRA*, 2019.

[39] N. Demmel, C. Sommer, D. Cremers, and V. Usenko. Square root bundle adjustment for large-scale reconstruction. In *CVPR*, 2021.

[40] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

[41] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *AAAI*, 2021.

[42] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 2008.

[43] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2020.

[44] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *ACRL*, 2017.

[45] A. D'Innocente and B. Caputo. Domain generalization with domain-specific aggregation modules. In *GCPR*, 2019.

[46] L. Fan, X. Xiong, F. Wang, N. Wang, and Z. Zhang. Rangedet: In defense of range view for lidar-based 3d object detection. In *ICCV*, 2021.

[47] E. Fini, E. Sangineto, S. Lathuilière, Z. Zhong, M. Nabi, and E. Ricci. A unified objective for novel class discovery. In *ICCV*, 2021.

[48] W. K. Fong, R. Mohan, J. V. Hurtado, L. Zhou, H. Caesar, O. Beijbom, and A. Valada. Panoptic nuscenes: A large-scale benchmark for lidar panoptic segmentation and tracking. *RAL*, 2021.

[49] L. Gao, J. Zhang, L. Zhang, and D. Tao. Dsp: Dual soft-paste for unsupervised domain adaptive semantic segmentation. In *ACMM*, 2021.

[50] C. Ge, J. Chen, E. Xie, Z. Wang, L. Hong, H. Lu, Z. Li, and P. Luo. Metabev: Solving sensor failures for 3d detection and map segmentation. In *ICCV*, 2023.

[51] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *CVPR*, 2012.

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

[53] M. Ghifary, D. Balduzzi, B.W. Kleijn, and M. Zhang. Scatter component analysis: A unified framework for domain adaptation and domain generalization. *T-PAMI*, 2016.

[54] GISGeography. What is photogrammetry? `https://gisgeography.com/what-is-photogrammetry/`.

[55] Z. Gojcic, C. Zhou, J.D. Wegner, and W. Andreas. The perfect match: 3D point cloud matching with smoothed densities. In *CVPR*, 2019.

[56] B. Graham, M. Engelcke, and L. van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018.

[57] B. Graham and L. van der Maaten. Submanifold sparse convolutional networks. *arXiv*, 2017.

[58] D. Griffiths and J. Boehm. SynthCity: A large scale synthetic point cloud. In *arXiv*, 2019.

[59] S. Guan, J. Xu, Y. Wang, B. Ni, and X. Yang. Bilevel online adaptation for out-of-domain human mesh reconstruction. In *CVPR*, 2021.

[60] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. Deep learning for 3d point clouds: A survey. *T-PAMI*, 2020.

[61] M. Hahner, C. Sakaridis, M. Bijelic, F. Heide, F. Yu, D. Dai, and L. Van Gool. Lidar snowfall simulation for robust 3d object detection. In *ICCV*, 2022.

[62] C. He, H. Zeng, J. Huang, XS. Hua, and L. Zhang. Structure aware single-stage 3d object detection from point cloud. In *CVPR*, 2020.

[63] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[64] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv*, 2015.

[65] J. Hoffman, E. Tzeng, T. Park, JY. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, 2018.

[66] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham. RandLA-Net: Efficient semantic segmentation of large-scale point clouds. In *CVPR*, 2020.

[67] S. Huang, Y. Xie, SC. Zhu, and Y. Zhu. Spatio-temporal self-supervised representation learning for 3d point clouds. In *ICCV*, 2021.

[68] Z. Huang, H. Wang, E. P. Xing, and D. Huang. Self-challenging improves cross-domain generalization. In *ECCV*, 2020.

[69] B. Hurl, K. Czarnecki, and S. Waslander. Precise synthetic image and lidar (PreSIL) dataset for autonomous vehicle perception. In *IVS*, 2019.

[70] A. Irschara, C. Zach, JM. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *CVPR*, 2009.

[71] S. Jadon. A survey of loss functions for semantic segmentation. In *CIBCB*, 2020.

[72] J. Janai, F. Guney, A. Behl, and A. Geiger. Computer vision for autonomous vehicles problems datasets and state of the art. *arXiv*, 2019.

[73] M. Jaritz, T.-H. Vu, R. De Charette, E. Wirbel, and P. Pérez. Cross-modal learning for domain adaptation in 3d semantic segmentation. *T-PAMI*, 2022.

[74] M. Jaritz, T.-H. Vu, R. de Charette, E. Wirbel, and P. Pérez. xMUDA: Cross-modal unsupervised domain adaptation for 3d semantic segmentation. In *CVPR*, 2020.

[75] J.Levinson, J.Askeland, J.Becker, J.Dolson, D.Held, S.Kammel, J.Z.Kolter, D.Langer, O.Pink, V.Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *IV*, 2011.

[76] H. Katti, M. Peelen, and SP. Arun. Machine vision benefits from human contextual expectations. *Scientific Reports*, 2019.

[77] A. Kendall, V. Badrinarayanan, and R. Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. In *BMVC*, 2017.

[78] A. Khosla, T. Zhou, T. Malisiewicz, A. Efros A, and A. Torralba. Undoing the damage of dataset bias. In *ECCV*, 2012.

[79] A. Kim, G. Brasó, A. Ošep, and L. Leal-Taixé. Polarmot: How far can geometric relations take us in 3d multi-object tracking? In *ECCV*, 2022.

[80] H. Kim, Y. Kang, C. Oh, and K.-J. Yoon. Single domain generalization for lidar semantic segmentation. In *CVPR*, 2023.

[81] T. Kim and C. Kim. Attract, perturb, and explore: Learning a feature alignment network for semi-supervised domain adaptation. In *ECCV*, 2020.

[82] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[83] L. Kong, Y. Liu, R. Chen, Y. Ma, X. Zhu, Y. Li, Y. Hou, Y. Qiao, and Z. Liu. Rethinking range view representation for lidar segmentation. In *ICCV*, 2023.

[84] L. Kong, Y. Liu, X. Li, R. Chen, W. Zhang, J. Ren, L. Pan, K. Chen, and Z. Liu. Robo3d: Towards robust and reliable 3d perception against corruptions. In *ICCV*, 2023.

[85] W.M. Kouw and M. Loog. A review of domain adaptation without target labels. *T-PAMI*, 2021.

[86] K.Saleh, A.Abobakr, M.Attia, J.Iskander, D.Nahavandi, M.Hossny, and S.Nahvandi. Domain adaptation for vehicle detection from bird's eye view lidar point cloud data. In *ICCV-W*, 2019.

[87] X. Lai, Y. Chen, F. Lu, J. Liu, and J. Jia. Spherical transformer for lidar-based 3d recognition. In *CVPR*, 2023.

[88] A. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.

[89] F. Langer, A. Milioto, A. Haag, J. Behley, and C. Stachniss. Domain Transfer for Semantic Segmentation of LiDAR Data using Deep Neural Networks. In *IROS*, 2020.

[90] F. Langer, A. Milioto, A. Haag, J. Behley, and C. Stachniss. Domain transfer for semantic segmentation of LiDAR data using deep neural networks. In *IROS*, 2021.

[91] M. Laskin, A. Srinivas, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *ICML*, 2020.

[92] T. Le and Y. Duan. Pointgrid: A deep network for 3d shape understanding. In *CVPR*, 2018.

[93] A. Lehner, S. Gasperini, A. Marcos-Ramiro, M. Schmidt, M.N. Mahani, N. Navab, B. Busam, and F. Tombari. 3d-vfield: Adversarial augmentation of point clouds for domain generalization in 3d object detection. In *CVPR*, 2022.

[94] J. Levinson et al. Towards fully autonomous driving: Systems and algorithms. In *IV*, 2011.

[95] D. Li and T. Hospedales. Online meta-learning for multi-source and semi-supervised domain adaptation. In *ECCV*, 2020.

[96] D. Li, Y. Yang, and Y.-Z. Song T. Hospedales. Learning to generalize: Meta-learning for domain generalization. In *AAAI*, 2018.

[97] M. Li, Y. Zhang, X. Ma, Y. Qu, and Y. Fu. Bev-dg: Cross-modal learning under bird's-eye view for domain generalization of 3d semantic segmentation. In *ICCV*, 2023.

[98] Q. Li, Y. Wang, Y. Wang, and H. Zhao. Hdmapnet: An online hd map construction and evaluation framework. In *ICRA*, 2022.

[99] Y. Li and J. Ibanez-Guzman. Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems. *IEEE Signal Process. Mag.*, 2020.

[100] Y. Li and J. Ibanez-Guzman. Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems. *IEEE SPM*, 2020.

[101] Y. Li, X. Tian, M. Gong, Y. Liu, T. Liu, K. Zhang, and D. Tao. Deep domain generalization via conditional invariant adversarial networks. In *ECCV*, 2018.

[102] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou. Revisiting batch normalization for practical domain adaptation. *arXiv*, 2016.

[103] J. Liang, D. Hu, and J. Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *ICML*, 2020.

[104] Velodyne Lidar. VelodyneLidar. `https://velodynelidar.com`, 2021.

[105] T.Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[106] Q. Liu, Q. Dou, L. Yu, and A.P. Heng. Ms-net: multi-site network for improving prostate segmentation with heterogeneous mri data. *T-MI*, 2020.

[107] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

[108] M. Long, Z. Cao, J. Wang, and M. Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, 2018.

[109] Z. Luo, Z. Cai, C. Zhou, G. Zhang, H. Zhao, S. Yi, S. Lu, H. Li, S. Zhang, and Z. Liu. Unsupervised domain adaptive 3d detection with multi-level consistency. In *ICCV*, 2021.

[110] W.C. Ma, I. Tartavull, I. A. Bârsan, S. Wang, M. Bai, G. Mattyus, N. Homayounfar, S. K. Lakshmikanth, A. Pokrovsky, and R. Urtasun. Exploiting sparse semantic hd maps for self-driving vehicle localization. In *IROS*, 2019.

[111] M. Mancini, Z. Akata, E. Ricci, and B. Caputo. Towards recognizing unseen categories in unseen domains. In *ECCV*, 2020.

[112] M. Mancini, H. Karaoguz, E. Ricci, P. Jensfelt, and B. Caputo. Kitting in the wild through online domain adaptation. In *IROS*, 2018.

[113] J. Mao, Y. Xue, M. Niu, H. Bai, J. Feng, X. Liang, H. Xu, and C. Xu. Voxel transformer for 3d object detection. In *ICCV*, 2021.

[114] F. Marra, C. Saltori, G. Boato, and L. Verdoliva. Incremental learning for the detection and classification of gan-generated images. In *WIFS*, 2019.

[115] G. Mei, F. Poiesi, C. Saltori, J. Zhang, E. Ricci, and N. Sebe. Overlap-guided gaussian mixture models for point cloud registration. In *WACV*, 2023.

[116] G. Mei, C. Saltori, F. Poiesi, E. Ricci, Q. Wu, and J. Zhang. Unsupervised point cloud representation learning by clustering and neural rendering. In *Under Peer-Review*, 2023.

[117] G. Mei, C. Saltori, F. Poiesi, J. Zhang, E. Ricci, N. Sebe, and Q. Wu. Data augmentation-free unsupervised learning for 3d point cloud understanding. *BMVC*, 2022.

[118] T. Meinhardt, A. Kirillov, L. Leal-Taixé, and C. Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *CVPR*, 2022.

[119] B. Michele, A. Boulch, G. Puy, T.-H. Vu, R. Marlet, and N. Courty. Saluda: Surface-based automotive lidar unsupervised domain adaptation. *3DV*, 2024.

[120] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *IROS*, 2019.

[121] JH. Moon, D. Das, and CS. Lee. Multi-step online unsupervised domain adaptation. In *ICASSP*, 2020.

[122] P. Morerio, J. Cavazza, and V. Murino. Minimal-entropy correlation alignment for unsupervised deep domain adaptation. *arXiv*, 2017.

[123] A. Nekrasov, J. Schult, O. Litany, B. Leibe, and F. Engelmann. Mix3D: Out-of-Context Data Augmentation for 3D Scenes. In *3DV*, 2021.

[124] V. Olsson, W. Tranheden, J. Pinto, and L. Svensson. Classmix: Segmentation-based data augmentation for semi-supervised learning. In *WACV*, 2021.

[125] X. Pan, P. Luo, J. Shi, and X. Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *ECCV*, 2018.

[126] Y. Pan, B. Gao, J. Mei, S. Geng, C. Li, and H. Zhao. Semanticposs: A point cloud dataset with large quantity of dynamic instances. *arXiv*, 2020.

[127] S. Peng, K. Genova, C. Jiang, A. Tagliasacchi, M. Pollefeys, T. Funkhouser, et al. Openscene: 3d scene understanding with open vocabularies. In *CVPR*, 2023.

[128] A. Petrovskaya and S. Thrun. Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots*, 2009.

[129] F. Poiesi and D. Boscaini. Distinctive 3D local deep descriptors. In *ICPR*, 2021.

[130] F. Poiesi and D. Boscaini. Learning general and distinctive 3D local deep descriptors for point cloud registration. *T-PAMI*, 2022.

[131] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, 2018.

[132] CR. Qi, O. Litany, K. He, and LJ. Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, 2019.

[133] C.R. Qi, H. Su, K. Mo, and L.J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[134] C.R. Qi, L. Yi, H. Su, and L.J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv*, 2017.

[135] G. Qian, Y. Li, H. Peng, J. Mai, H. Hammoud, M. Elhoseiny, and B. Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *NeurIPS*, 2022.

[136] R. Qian, X. Lai, and X. Li. Badet: Boundary-aware 3d object detection from point clouds. *Pattern Recognition*, 2022.

[137] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *ICLR*, 2021.

[138] M.A. Rahman and Y. Wang. Optimizing intersection-over-union in deep neural networks for image segmentation. In *ISVC*, 2016.

[139] C. B. Rist, M. Enzweiler, and D. M. Gavrila. Cross-sensor deep domain adaptation for lidar detection and segmentation. In *IV*, 2019.

[140] L. Riz, C. Saltori, E. Ricci, and F. Poiesi. Novel class discovery for 3d point cloud semantic segmentation. In *CVPR*, 2023.

[141] L. Riz, C. Saltori, Y. Wang, E. Ricci, and F. Poiesi. Novel class discovery meets foundation models for 3d semantic segmentation. In *Under Peer-Review*, 2023.

[142] M. Rochan, S. Aich, E. R. Corral-Soto, A. Nabatchian, and B. Liu. Unsupervised domain adaptation in lidar semantic segmentation with self-supervision and gated adapters. In *ICRA*, 2022.

[143] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[144] F. Rosique, P.J. Navarro, C. Fernández, and A. Padilla. A systematic review of perception system and simulators for autonomous vehicles research. *Sensors*, 2019.

[145] S. Roy, W. Menapace, S. Oei, B. Luijten, E. Fini, C. Saltori, et al. Deep learning for classification and localization of covid-19 markers in point-of-care lung ultrasound. *T-MI*, 2020.

[146] S. Roy, A. Siarohin, E. Sangineto, S. R. Bulo, N. Sebe, and E. Ricci. Unsupervised domain adaptation using feature-whitening and consensus loss. In *CVPR*, 2019.

[147] R.B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3D registration. In *ICRA*, 2009.

[148] R. Sabzevari and D. Scaramuzza. Multi-body motion estimation from monocular vehicle-mounted cameras. *T-RO*, 2016.

[149] A. Sahoo, R. Shah, R. Panda, K. Saenko, and A. Das. Contrast and mix: Temporal contrastive video domain adaptation with background mixing. *NeurIPS*, 2021.

[150] K. Saito, D. Kim, S. Sclaroff, T. Darrell, and K. Saenko. Semi-supervised domain adaptation via minimax entropy. In *ICCV*, 2019.

[151] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, 2018.

[152] C. Saltori, F. Galasso, G. Fiameni, N. Sebe, F. Poiesi, and E. Ricci. Compositional semantic mix for domain adaptation in point cloud segmentation. *T-PAMI*, 2023.

[153] C. Saltori, F. Galasso, G. Fiameni, N. Sebe, E. Ricci, and F. Poiesi. Cosmix: Compositional semantic mix for domain adaptation in 3d lidar segmentation. In *ECCV*, 2022.

[154] C. Saltori, E. Krivosheev, S. Lathuiliére, N. Sebe, F. Galasso, G. Fiameni, E. Ricci, and F. Poiesi. Gipso: Geometrically informed propagation for online adaptation in 3d lidar segmentation. In *ECCV*, 2022.

[155] C. Saltori, S. Lathuiliére, N. Sebe, E. Ricci, and F. Galasso. Sf-uda3d: Source-free unsupervised domain adaptation for lidar-based 3d object detection. *3DV*, 2020.

[156] C. Saltori, A. Osep, E. Ricci, and L. Leal-Taixé. Walking your lidog: A journey through multiple domains for lidar semantic segmentation. In *ICCV*, 2023.

[157] C. Saltori, P. Rota, N. Sebe, and J. Almeida. Low-budget label query through domain alignment enforcement. *CVIU*, 2022.

[158] C. Saltori, S. Roy, N.Sebe, and G. Iacca. Regularized evolutionary algorithm for dynamic neural topology search. In *ICIAP*, 2019.

[159] J. Sanchez, JE. Deschaud, and F. Goulette. Domain generalization of 3d semantic segmentation in autonomous driving. In *ICCV*, 2023.

[160] J. Schlosser, C. Chow, and Z. Kira. Fusing lidar and images for pedestrian detection using convolutional neural networks. In *ICRA*, 2016.

[161] S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge. Improving robustness against common corruptions by covariate shift adaptation. *NeurIPS*, 2020.

[162] J. Schult, F. Engelmann, A. Hermans, O. Litany, S. Tang, and B. Leibe. Mask3d: Mask transformer for 3d semantic instance segmentation. In *ICRA*, 2023.

[163] S. Shankar, V. Piratla, S. Chakrabarti, S. Chaudhuri, and P. Jyothi. Generalizing across domains via cross-gradient training. *ICLR*, 2018.

[164] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *CVPR*, 2020.

[165] S. Shi, L. Jiang, J. Deng, Z. Wang, C. Guo, J. Shi, X. Wang, and H. Li. Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. *IJCV*, 2023.

[166] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019.

[167] W. Shi and R. Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *CVPR*, 2020.

[168] I. Shin, S. Woo, F. Pan, and I.S. Kweon. Two-phase pseudo label densification for self-training based domain adaptation. In *ECCV*, 2020.

[169] S.Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. *arXiv*, 2017.

[170] R. Strudel, R. Garcia, I. Laptev, and C. Schmid. Segmenter: Transformer for semantic segmentation. In *ICCV*, 2021.

[171] B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV*, 2016.

[172] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020.

[173] Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020.

[174] A. Takmaz, E. Fedele, R. W. Sumner, M. Pollefeys, F. Tombari, and F. Engelmann. Openmask3d: Open-vocabulary 3d instance segmentation. In *NeurIPS*, 2023.

[175] H. Tang*, Z. Liu*, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *ECCV*, 2020.

[176] D. Tanneberg, J. Peters, and E. Rueckert. Efficient online adaptation with stochastic recurrent neural networks. In *Humanoids*, 2017.

[177] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017.

[178] H. Thomas, C.R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and J. L. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019.

[179] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, and G. Hoffmann. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 2006.

[180] M. Toldo, A. Maracani, U. Michieli, and P. Zanuttigh. Unsupervised domain adaptation in semantic segmentation: a review. *Technologies*, 2020.

[181] A. Tompkins, R. Senanayake, and F. Ramos. Online domain adaptation for occupancy mapping. *arXiv*, 2020.

[182] A. Tonioni, F. Tosi, M. Poggi, S. Mattoccia, and L. Di Stefano. Real-time self-adaptive deep stereo. In *CVPR*, 2019.

[183] A. Torralba and A. Oliva. Depth estimation from image structure. *T-PAMI*, 2002.

[184] W. Tranheden, V. Olsson, J. Pinto, and L. Svensson. Dacs: Domain adaptation via cross-domain mixed sampling. In *WACV*, 2021.

[185] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017.

[186] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of field Robotics*, 2008.

[187] U.Rosolia, S.De Bruyne, and A.G.Alleyne. Autonomous vehicle control: A nonconvex approach for obstacle avoidance. *T-CST*, 2016.

[188] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *NeurIPS*, 2017.

[189] P. Voigtlaender and B. Leibe. Online adaptation of convolutional neural networks for video object segmentation. *arXiv*, 2017.

[190] R. Volpi, P. De Jorge, D. Larlus, and G. Csurka. On the road to online adaptation for semantic image segmentation. In *CVPR*, 2022.

[191] T.U. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *CVPR*, 2019.

[192] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell. Tent: Fully test-time adaptation by entropy minimization. *ICLR*, 2021.

[193] G. Wang, H. Han, S. Shan, and X. Chen. Cross-domain face presentation attack detection via multi-domain disentangled representation learning. In *CVPR*, 2020.

[194] H. Wang, Z. He, Z.C. Lipton, and E.P. Xing. Learning robust representations by projecting superficial statistics out. *ICLR*, 2019.

[195] Q. Wang, D. Dai, L. Hoyer, L. Van Gool, and O. Fink. Domain adaptive semantic segmentation with self-supervised depth estimation. In *ICCV*, 2021.

[196] Y. Wang, X. Chen, Y. You, L. Erran, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao. Train in germany, test in the usa: Making 3d object detectors generalize. In *CVPR*, 2020.

[197] Y. Wang and JM. Solomon. Deep closest point: Learning representations for point cloud registration. In *ICCV*, 2019.

[198] Y. Wang, J. Yin, W. Li, P. Frossard, R. Yang, and J. Shen. Ssda3d: Semi-supervised domain adaptation for 3d object detection from point cloud. In *AAAI*, 2023.

[199] S. Weber, N. Demmel, TC. Chan, and D. Cremers. Power bundle adjustment for large-scale 3d reconstruction. In *CVPR*, 2023.

[200] Y. Wei, Z. Wei, Y. Rao, J. Li, J. Zhou, and J. Lu. Lidar distillation: Bridging the beam-induced domain gap for 3d object detection. In *ECCV*, 2022.

[201] X. Weng, J. Wang, D. Held, and K. Kitani. 3D Multi-Object Tracking: A Baseline and New Evaluation Metrics. *IROS*, 2020.

[202] Xinshuo Weng and Kris Kitani. A baseline for 3d multi-object tracking. *arXiv*, 2019.

[203] B. Wu, A. Wan, X. Yue, and K. Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *ICRA*, 2018.

[204] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *ICRA*, 2019.

[205] X. Wu, Y. Lao, L. Jiang, X. Liu, and H. Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. *NeurIPS*, 2022.

[206] Y. Wu, D. Inkpen, and Ahmed A. El-Roby. Dual mixup regularized learning for adversarial domain adaptation. In *ECCV*, 2020.

[207] A. Xiao, J. Huang, D. Guan, K. Cui, S. Lu, and L. Shao. Polarmix: A general data augmentation technique for lidar point clouds. *NeurIPS*, 2022.

[208] A. Xiao, J. Huang, D. Guan, F. Zhan, and S. Lu. Synlidar: Learning from synthetic lidar sequential point cloud for semantic segmentation. In *AAAI*, 2022.

[209] C. Xu and J.J. Corso. Libsvx: A supervoxel library and benchmark for early video processing. *IJCV*, 2016.

[210] M. Xu, J. Zhang, B. Ni, T. Li, C. Wang, Q. Tian, and W. Zhang. Adversarial domain adaptation with domain mixup. In *AAAI*, 2020.

[211] Q. Xu, Y. Zhou, W. Wang, C. R. Qi, and D. Anguelov. Spg: Unsupervised domain adaptation for 3d object detection via semantic point generation. In *ICCV*, 2021.

[212] Y. Xu, A. Osep, Y. Ban, R. Horaud, Laura L. Leal-Taixé, and X. Alameda-Pineda. How to train your deep multi-object tracker. In *CVPR*, 2020.

[213] X.Yue, B.Wu, S.A.Seshia, K.Keutzer, and A.L.Sangiovanni-Vincentelli. A LiDAR point cloud generator: from a virtual world to autonomous driving. In *ICMR*, 2018.

[214] Y. Yan, Y. Mao, and B. Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018.

[215] J. Yang, S. Shi, Z. Wang, H. Li, and X. Qi. St3d: Self-training for unsupervised domain adaptation on 3d object detection. In *CVPR*, 2021.

[216] J. Yang, S. Shi, Z. Wang, H. Li, and X. Qi. St3d++: Denoised self-training for unsupervised domain adaptation on 3d object detection. *T-PAMI*, 2022.

[217] S. Yang, J. van de Weijer, L. Herranz, S. Jui, et al. Exploiting the intrinsic neighborhood structure for source-free domain adaptation. *NeurIPS*, 2021.

[218] Z. Yang, Y. Sun, S. Liu, and J. Jia. 3dssd: Point-based 3d single stage object detector. In *CVPR*, 2020.

[219] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *ICCV*, 2019.

[220] Y.Chen, W.Li, C.Sakaridis, D.Dai, and L.Van Gool. Domain adaptive faster r-cnn for object detection in the wild. In *CVPR*, 2018.

[221] L. Yi, B. Gong, and T. Funkhouser. Complete & label: A domain adaptation approach to semantic segmentation of lidar point clouds. *arXiv*, 2021.

[222] Z. Yihan, C. Wang, Y. Wang, H. Xu, C. Ye, Z. Yang, and C. Ma. Learning transferable features for point cloud detection via 3d contrastive co-training. *NeurIPS*, 2021.

[223] T. Yin, X. Zhou, and P. Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, 2021.

[224] Y.LeCun, Y.Bengio, and G.Hinton. Deep learning. *Nature*, 2015.

[225] Y.Liu, W.Zhang, and J.Wang. Source-free domain adaptation for semantic segmentation. In *CVPR*, 2021.

[226] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019.

[227] Y.Zhou and O.Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018.

[228] X. Zhan, J. Xie, Z. Liu, Y.S Ong, and C.C. Loy. Online deep clustering for unsupervised representation learning. In *CVPR*, 2020.

[229] H. Zhang, M. Cisse, Y.N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

[230] J. Zhang, L. Chen, B. Ouyang, B. Liu, J. Zhu, Y. Chen, Y. Meng, and D. Wu. Pointcutmix: Regularization strategy for point cloud classification. *Neurocomputing*, 2021.

[231] P. Zhang, B. Zhang, T. Zhang, D. Chen, Y. Wang, and F. Wen. Prototypical pseudo label denoising and target structure learning for domain adaptive semantic segmentation. In *CVPR*, 2021.

[232] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *CVPR*, 2020.

[233] Z. Zhang, S. Lathuilière, A. Pilzer, N. Sebe, E. Ricci, and J. Yang. Online adaptation through meta-learning for stereo depth estimation. *arXiv*, 2019.

[234] H. Zhao, L. Jiang, J. Jia, P. Torr, and V. Koltun. Point transformer. In *ICCV*, 2021.

[235] S. Zhao, Y. Wang, B. Li, B. Wu, Y. Gao, P. Xu, T. Darrell, and K. Keutzer. epointda: An end-to-end simulation-to-real domain adaptation framework for lidar point cloud segmentation. *arXiv*, 2020.

[236] Y. Zhao, Z. Zhong, N. Zhao, N. Sebe, and G.H. Lee. Style-hallucinated dual consistency learning for domain generalized semantic segmentation. In *ECCV*, 2022.

[237] W. Zheng, W. Tang, L. Jiang, and CW. Fu. Se-ssd: Self-ensembling single-stage object detector from point cloud. In *CVPR*, 2021.

[238] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C.C. Loy. Domain generalization in vision: A survey. *T-PAMI*, 2021.

[239] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and CC. Loy. Domain generalization: A survey. *T-PAMI*, 2022.

[240] K. Zhou, Y. Yang, T. Hospedales, and T. Xiang. Deep domain-adversarial image generation for domain generalization. In *AAAI*, 2020.

[241] Y. Zhou, Y. Wang, F. Poiesi, Q. Qin, and Y. Wan. Loop closure detection using local 3D deep descriptors. *RAL*, 2022.

[242] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[243] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *CVPR*, 2021.

[244] L. Zou, H. Tang, K. Chen, and K. Jia. Geometry-aware self-training for unsupervised domain adaptation on object point clouds. In *CVPR*, 2021.

[245] Y. Zou, Z. Yu, BVK. Kumar, and J. Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *ECCV*, 2018.

[246] Y. Zou, Z. Yu, X. Liu, B. Kumar, and J. Wang. Confidence regularized self-training. In *ICCV*, 2019.