UADA3D: Unsupervised Adversarial Domain Adaptation for 3D Object Detection With Sparse LiDAR and Large Domain Gaps

Maciej K. Wozniak, *Graduate Student Member, IEEE*, Mattias Hansson, Marko Thiel, and Patric Jensfelt, *Member, IEEE*

***Abstract*—In this study, we address a gap in existing unsupervised domain adaptation approaches on LiDAR-based 3D object detection, which have predominantly concentrated on adapting between established, high-density autonomous driving datasets. We focus on sparser point clouds, capturing scenarios from different perspectives: not just from vehicles on the road but also from mobile robots on sidewalks, which encounter significantly different environmental conditions and sensor configurations. We introduce Unsupervised Adversarial Domain Adaptation for 3D Object Detection (UADA3D). UADA3D does not depend on pre-trained source models or teacher-student architectures. Instead, it uses an adversarial approach to directly learn domain-invariant features. We demonstrate its efficacy in various adaptation scenarios, showing significant improvements in both self-driving car and mobile robot domains.**

***Index Terms*—Deep learning for visual perception, object detection, segmentation and categorization.**

I. Introduction

LIDAR-BASED perception systems are essential for the safe navigation of autonomous vehicles such as self-driving cars or mobile robots. A key challenge is the reliable detection and classification of objects within a vehicle's environment [1]. SOTA 3D object detection methods highly depend on quality and diversity of the datasets used for training, but also on how closely these datasets reflect inference conditions. Acquiring and annotating such data remains a significant technical, moral and practical challenge, being both time-consuming and labor-intensive [2]. This presents a major obstacle in development and deployment of 3D object detection models at scale.

A crucial technique to mitigate these challenges is domain adaptation (DA). DA addresses the problem of adapting models trained on a source domain with ample labeled data to a target domain where labels might be scarce (as in semi-supervised DA) or completely unavailable (as in unsupervised DA – UDA). UDA methods can substantially improve model performance in new, unfamiliar, or changing environments without the need to label new training samples. In the context of autonomous vehicles, discrepancies between source and target domains, often referred to as domain shift or domain gap, can be caused by changes in weather conditions [3], variations in object sizes [4], different sensor setups and deployment environments [1], [5] but also due to the transition from simulated to real-world environments [6].

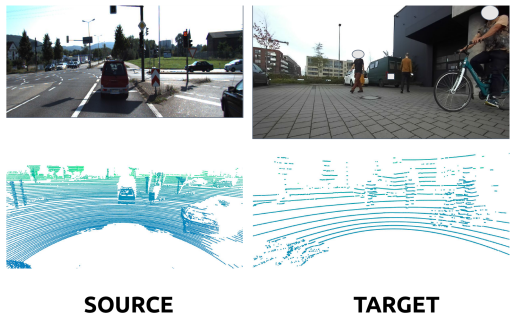


Fig. 1. Comparison of KITTI (source) and robot data (target). We observe that differences in operating environments, sensor positions, and LiDAR density create a large domain gap. This presents a significant challenge for LiDAR-based 3D object detectors, as well as for the task of domain adaptation.

UDA has received considerable attention in the field of computer vision. However, recent UDA approaches [4], [7], [8], [9], [10] for LiDAR-based 3D object detection primarily focus on automotive applications and corresponding datasets with dense LiDAR data, featuring 128, 64, or 32 layers [11], [12], [13]. We find a notable research gap when it comes to UDA for 3D object detection models explicitly addressing larger domain shifts than those associated with classical self-driving cars, such as last-mile delivery mobile robots. Those robots operate in an environment sharing many properties with that of self-driving cars, potentially allowing them to benefit from widely available datasets, yet they display significant differences: LiDAR sensors differ in both sensor position and resolution, resulting in sparser point clouds where the ground plane is located much closer to the sensor. Moreover, sidewalk environments are considerably different from roadways. While the same object classes are present, their distribution and relative distances to the sensor are distinctly different than from the car perspective, resulting in a different point density per object, as shown in Fig. 1 and further discussed in Section IV.

Received 1 July 2024; accepted 18 October 2024. Date of publication 29 October 2024; date of current version 7 November 2024. This article was recommended for publication by Associate Editor Matthew Gadd and Editor Abhinav Valada upon evaluation of the reviewers' comments. This work was supported by Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation. *(Corresponding author: Maciej K. Wozniak.)*

Maciej K. Wozniak, Mattias Hansson, and Patric Jensfelt are with the Division of Robotics, Perception, and Learning, KTH Royal Institute of Technology, 114 28 Stockholm, Sweden (e-mail: maciejw@kth.se).

Marko Thiel is with the Institute for Technical Logistics, Hamburg University of Technology, 21073 Hamburg, Germany.

Our code is open-source and will be available at https://maxiuw.github.io/uda.

Digital Object Identifier 10.1109/LRA.2024.3487489

We address UDA in scenarios involving sparse LiDAR and large domain shifts: 1) between widely used automotive datasets, 2) for sim-to-real tasks, and 3) for larger domain shifts using the last-mile delivery robot LAURA [14] with a 16-layer LiDAR sensor operating on a sidewalk and indoors.

Inspired by 2D image-based adversarial DA [15], [16], we propose a novel approach for 3D point cloud data: Unsupervised Adversarial Domain Adaptation for 3D Object Detection (**UADA3D**). Our method uses adversarial adaptation based on class-wise domain discriminators with a gradient reversal layer (GRL) to facilitate the learning of domain-invariant representations. The domain discriminator is trained to maximize its ability to distinguish between the target and source domains, while the model is trained to minimize this ability, resulting in domain-invariant feature learning.

The approach we present offers significant advantages over existing UDA methods for LiDAR-based 3D object detection: UADA3D does not require pre-trained source models, avoids the complexity of teacher-student architectures, and eliminates the inherent uncertainties of pseudo-labels. Instead, it directly learns features that are invariant across the source and target domains. Furthermore, our approach successfully adapts models to multiple object classes simultaneously (e.g. vehicle, pedestrian, and cyclist classes), a capability rarely demonstrated by other SOTA methods [4], [17], [18], [19]. Our approach is particularly well-suited for applications with larger domain shifts such as multiple object categories, dissimilar operation environments, and sparse LiDAR data.

Our main contributions are as follows: (i) we introduce UADA3D, an unsupervised adversarial domain adaptation approach for 3D LiDAR-based object detection (Section III); (ii) we test UADA3D with two widely used object detection models and achieve SOTA performance in various challenging UDA scenarios on autonomous driving and mobile robots datasets (Sections IV and V).

II. Related Work

Unsupervised Domain Adaptation for LiDAR-based 3D Object Detection: Numerous research projects have focused on adapting 3D object detection models between high-resolution LiDAR datasets, ranging from 128 to 32 layers, typically found in self-driving scenarios [4], [5], [8], [10], [19], [20], [21], [22], [23], [24], [25], [26]. However, little attention has been given to adapting these models to sparser LiDAR setups, which are common in small robotic platforms [14]. Peng et al. [5] focus specifically on model adaptation between LiDARs with 64 layers and 32 layers, and vice versa. Their results indicate that performance drastically decreases when models trained on 64-layer LiDAR data are adapted to a sparser 32-layer target domain. A similar pattern is observed in ST3D++[20], which leverages pseudo-labeling to generate labels for the target domain. MS3D++ [8] leverages multiple pre-trained detectors to achieve more accurate predictions, but again does not focus on very sparse data. Additionally, Cheng et al. [9] show that UDA methods does not maintain consistent performance across different object classes, and the majority of the methods adapt only between the Car/Vehicle class [8], [17]. Additionally, they do not focus on sparse LiDAR or different operating environments.

Recent works, such as DTS [18], which uses a student-teacher architecture with feature-level graph matching, show performance degradation with 16-layer LiDAR, although they do not report quantitative results. LiDAR Distillation (L.D.) [17] focuses exclusively on adapting models to sparser domains by using regression loss to minimize the difference between the BEV-feature maps predicted by student and teacher networks. However, their evaluations are conducted solely on autonomous driving datasets. While they extensively discuss the use of artificially downsampled 16-layer LiDAR data, they do not consider factors such as LiDAR sensor position (e.g., large height shift), domains that differ significantly (e.g., sidewalks versus streets), nor multiple classes. Most of the methods discussed above rely on a student-teacher approach, meaning they require a model pre-trained on the source domain to distill knowledge from the teacher to the student and/or to generate pseudo labels. In contrast, our method employs an adversarial learning approach. Consequently, we do not depend on a pre-trained teacher model to generate, for example, pseudo labels, which are not necessarily high-quality substitutes for ground truth and may even result in decreased model performance, as we later demonstrate in Section V.

Adversarial Domain Adaptation: Adversarial domain adaptation (ADA) relies on a discriminative network structure that leverages domain discriminators to achieve domain invariance. The GRL [15] reverses gradients linking the feature extractor θ_f and the discriminator θ_D. The total objective is to minimize: $L_{\text{total}} = L_{\text{task}}(\theta_f) - \lambda L_{\text{domain}}(\theta_f, \theta_D)$ where λ controls the weight of gradient reversal. Minimizing L_{task} ensures that the feature extractor learns task-specific features, and minimizing $-\lambda L_{\text{domain}}$ (due to the negative sign) ensures that the feature extractor learns domain-invariant features. This end-to-end training simultaneously employs source and target data, aligning features across domains while minimizing detection loss in the source data.

Following this, Chen et al. [16] introduced a two-stage adaptation method, incorporating image-level and instance-level adaptation. Saito et al. [27] proposed strong and weak adaptation, emphasizing local object feature alignment. He and Zhang [28] employed multiple GRLs for global adaptation at different convolutional layers. Xu et al. [29] introduced categorical regularization between image- and instance-level domain classification, using a regularization loss based on Euclidean distance. Li et al. [3] introduced AdvGRL, replacing the constant hyperparameter λ with an adaptive λ_{adv} to address challenging training samples. ADA has found a lot of use in object detection, classification, or segmentation in image space [15], [16], [30], [31]. SRDAN [32] or STAL3D [26] use adversarial approaches but their architecture is much different, however, they do not focus on per-class domain prediction, conditional adaptation or analyze different alignment strategies. They focus on adaptation between autonomous driving datasets, not LiDAR sparseness and distinct environments.

Our work concentrates on adversarial domain adaptation in LiDAR-based 3D object detection, distinct from 2D methods. Our approach tackles the unique challenges of spatial data handling, arising from different point cloud densities and operating environments, through data augmentation and method design. We address it particularly with our discriminator utilizing features masked by 3D bounding boxes. Extracting these features from sparser, irregularly spaced point clouds is significantly more challenging than from pixel grids. This often requires transitioning to representations like Bird-Eye-View (BEV) feature space, as illustrated in Fig. 2. Additionally, working with point clouds involves making predictions based on partially missing,

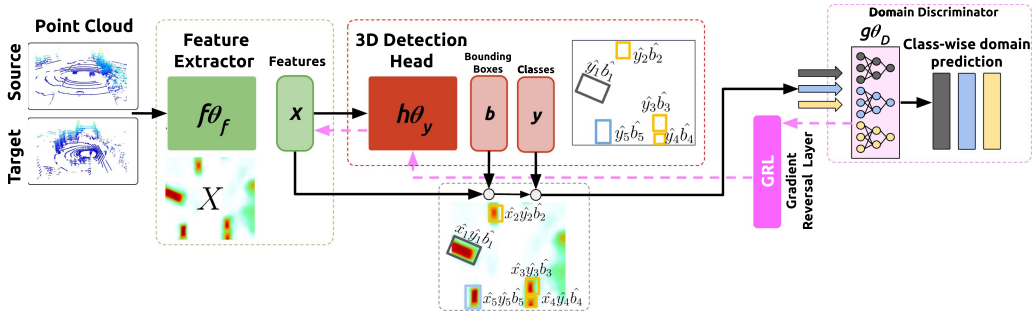


Fig. 2. An overview of **UADA3D** (**black** arrows show forwards, and **pink** backward pass). While the primary task of f_{θ_f} and h_{θ_y} is 3D object detection, the discriminator g_{θ_D} aims to classify the domain of each detected instance. Discriminator's loss, reversed by GRL, encourages the detector to learn features that are not only effective for object detection but also invariant across domains.

occluded, or incomplete data (please, refer to our website for qualitative results), caused by low angular resolution, potentially resulting in a small number of points per object, even at short distances to the object. In the realm of UDA for segmentation, existing methods employ adversarial approaches with marginal alignment or non-adversarial methods [31], [33]. Our approach utilizes conditional alignment, yielding superior performance for LiDAR-based 3D object detection (see Section V-A for a comparison).

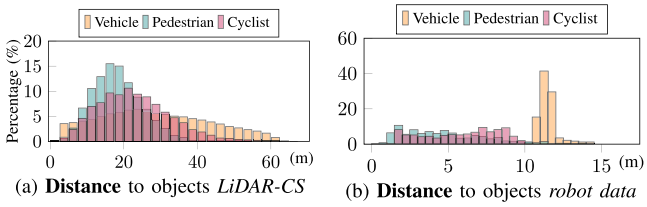
III. Method

A. Problem Formulation

Suppose that Q is a point cloud, and X is its feature representation learned by the feature extraction network f_{θ_f}. The detection head h_{θ_y} uses these features to predict $P(Y|X)$, where $Y = (y, b)$ are the category labels y and bounding boxes b. Q is sampled from the source domain $\mathcal{D}_s$ and target domain $\mathcal{D}_t$. The objective is to learn generalized weights θ_f and θ_y between domains such that $P(Y_s, X_s) \approx P(Y_t, X_t)$. Since $P(Y, X) = P(Y|X)P(X)$, the domain adaptation task for LiDAR-based object detectors is to align the marginal probability distributions $P(X_s)$ and $P(X_t)$ as well as the conditional probability distributions $P(Y_s|X_s)$ and $P(Y_t|X_t)$. Note that target labels Y_t are not available during training, thus we must use unsupervised domain adaptation.

B. Method Overview

Marginal adaptation, i.e., aligning $P(X)$, overlooks category and position labels, which can lead to uneven and biased adaptation. This may reduce the target domain's discriminative ability. Aligning $P(Y|X)$ places direct emphasis on the task-specific outcomes (class labels and bounding boxes) in relation to the features. By focusing on $P(Y|X)$, we hypothesize that the adaptation process also becomes more robust to variations in feature distributions across domains, concentrating on the essential task of detecting objects. Furthermore, in Figs. 3 and 4, we show that the distribution of points in different categories per object varies significantly between the datasets due to LiDAR density, position and operating environment, while the vehicle size slightly varies, depending on the dataset country of origin. While we chose to use conditional alignment, we compare our method with different alignment strategies: marginal and joint distribution alignment in Section V-A.



(a) **Distance** to objects *LiDAR-CS* (b) **Distance** to objects *robot data*

Fig. 3. Comparison of objects in LiDAR-CS and robot datasets.

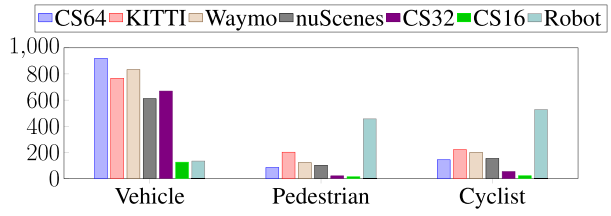


Fig. 4. Average number of points in an object per class.

Algorithm 1: UADA3D.

Input Labeled source dataset $\mathcal{D}_s : \{(Q^s, Y^s)\}^{N_s}$, unlabeled target dataset $\mathcal{D}_t : \{(Q^t)\}^{N_t}$
Output Weights: backbone θ_f, detection head θ_y, discriminators θ_D

$\theta_f, \theta_y, \theta_D \leftarrow WeightInitialization$
for $Q \in \mathcal{D}_s, \mathcal{D}_t$ **do**
  $X \leftarrow f_{\theta_f}(Q)$
  $\hat{Y} \leftarrow h_{\theta_y}(X) \triangleright \hat{Y} = (\hat{y}, \hat{b})$
  if source domain **then**
    $\theta_y \leftarrow UpdateWeights(\theta_y, \frac{\partial \mathcal{L}_{det}}{\partial \theta_y})$
    $\theta_f \leftarrow UpdateWeights(\theta_f, \frac{\partial \mathcal{L}_{det}}{\partial \theta_f})$
  end if
  $\mathcal{L}_C \leftarrow \frac{1}{N} \sum^{N} \hat{y}_{k,n} \odot (g_{\theta_D,k}(x_n, \hat{b_n}) - d)^2$
  $\theta_D \leftarrow UpdateWeights(\theta_D, \frac{\partial \mathcal{L}_C}{\partial \theta_D})$
  $\theta_y \leftarrow UpdateWeights(\theta_y, -\lambda \frac{\partial \mathcal{L}_C}{\partial \theta_y})$
  $\theta_f \leftarrow UpdateWeights(\theta_f, -\lambda \frac{\partial \mathcal{L}_C}{\partial \theta_f})$
end for
return $\theta_f^*, \theta_y^*, \theta_D^*$

Fig. 2 provides a schematic overview of our method UADA3D. In each iteration, a batch of samples Q from source $\mathcal{D}_s$ and target $\mathcal{D}_t$ domain is fed to the feature extractor f_{θ_f}. In each batch we sample from source and target training splits. The number of samples from the target and source data differ in each batch since the probability of drawing the sample is proportional to the dataset size and the model sees each sample once per epoch. Next, for each sample, features are extracted, and fed to the detection head h_{θ_y} that predicts 3D bounding boxes (lines 3-4in Algorithm 1). The object detection loss is calculated only for the labeled samples from source domain (lines 6-7). The probability distribution alignment branch uses the domain discriminator g_{θ_D} (line 9) to predict from which domain samples came from, based on the extracted features X and predicted labels $\hat{Y}$. The domain loss $\mathcal{L}_\mathcal{C}$ is calculated for all samples (line 9). Next, $\mathcal{L}_\mathcal{C}$ is backpropagated through the discriminators (line 10) and through the gradient reversal layer (GRL) with the coefficient λ, that reverses the gradient during backpropagation, to the detection head and feature extractor (lines 11-12). This adversarial training scheme works towards creating domain invariant features. Thus, our network learns how to extract features that will be domain invariant but also how to provide accurate predictions. Therefore, we seek for the optimal parameters θ_f^*, θ_y^*, and θ_D^*, that satisfy:

$$\begin{aligned} \theta_D^* &= \arg\min_{\theta_D} \mathcal{L}_C \\ (\theta_f^*, \theta_y^*) &= \arg\min_{\theta_f, \theta_y} \mathcal{L}_{det} - \lambda \mathcal{L}_C \end{aligned} \tag{1}$$

where $\mathcal{L}_{det}$ is the detection loss (described in [34], [35]) calculated only for the labeled source domain, $\mathcal{L}_C$ is the domain loss calculated for both source and target domains.

Our approach builds on the method by Ganin and Lempitsky [15]. However, what they proposed is more similar to the marginal adaptation approach. They do not mask the features with the predicted bounding boxes but predict the domain based on the global features. We introduce feature masking and a class-wise domain discriminator. Additionally, we use prediction confidence to weight discriminator loss, such that low confidence predictions do not influence the performance. These novel aspects and 3D application distinguish us from the approach proposed by Ganin and Lempitsky.

C. Feature Masking

Feature masking plays a crucial role in predicting the domain based on specific object features. Masking enables the model to focus solely on the features corresponding to each instance, thus enhancing the relevance and accuracy of the domain prediction. In Fig. 2, we show how features extracted from a point cloud Q are masked and used for domain prediction. The input to the class-wise domain discriminators $g_{\theta_{D,k}}$ is $(x, \hat{b})$, where x are masked features, $\hat{b}$ are predcited bounding boxes, and (a, b) denotes a concatenation. To obtain masked features x, we mask the feature map $X = f_{\theta f}(Q)$ with each predicted bounding box $\hat{b}_n$ creating corresponding masked features x_n. Finally, we concatenate x_n with the bounding box $\hat{b}_n$ and feed to the corresponding class-wise discriminator $g_{\theta_{D,k}}$. We further discuss the importance of feature masking in Section V-A.

D. Conditional Distribution Alignment

The conditional distribution alignment module, shown in Fig. 2, has the task of reducing the discrepancy between the conditional distribution $P(Y_s|X_s)$ of the source and $P(Y_t|X_t)$ of the target. As we highlighted in Section III-B, show in Fig. 4 and later discuss in Section IV-A, we can see a large difference between how objects from each category appear in different domains. Thus, instead of having one discriminator, we use $K = 3$ class-wise domain discriminators $g_{\theta_{D,k}}$, corresponding to vehicle, pedestrian, and cyclist classes. The conditional distribution alignment module is trained using the least-squares loss function:

$$\mathcal{L}_C = \frac{1}{N} \sum_{n=1}^{N} \hat{y}_{k,n} \odot (g_{\theta_{D,k}}(x_n, \hat{b}_n) - d)^2 \tag{2}$$

where N is the number of labels, $\hat{y}_{k,n}$ corresponding class confidence of instance n and $\odot$ is element-wise multiplication. The loss is backpropagated to the discriminators (line 9in Algorithm 1). Next, $\mathcal{L}_C$ is backpropagated through GRL to the detection head h_{θ_y} and the feature extractor f_{θ_f}.

E. Data Augmentation

Differences between LiDAR domains include different densities and object sizes. We use downsampling, a commonly used augmentation approach since the domain gap can be partially remedied by reducing LiDAR layers of the source data to 16 or 32 to better match the target domain LiDAR data as highlighted in [36]. LiDAR-CS [36], Waymo [12], and nuScenes [13] datasets contain vehicle sizes that correspond to the large vehicle sizes found in the USA, while our robot and KITTI [11] are collected in Europe where vehicles are smaller. Random object scaling (ROS) [20] is applied to source domain to address this vehicle size bias. While previous UDA methods on LiDAR-based 3D object detection often apply domain adaptation only to a single object category, we consider it a multiclass problem. Thus, we chose ROS with different scaling intervals for the three classes. The interested reader can find a more detailed analysis of the models' performance with ROS and downsampling on the project's webpage. Finally, points of both domains were shifted in vertical direction so that $z = 0$ at the position of the ground plane in sensor coordinates.

IV. Experiments Setup

We compare performance of IA-SSD [35] and Centerpoint [34] on large number of UDA scenarios using our method, UADA3D, against that of other SOTA unsupervised domain adaptation approaches (ST3D++ [20], DTS [18], L.D. [17], and MS3D++ [8]). We chose IA-SSD due to its efficiency, speed and low-memory requirements as a potential good fit for robotics application. Centerpoint is more robust detector achieving higher performance, however, with larger computational cost. In IA-SSD, the adaptive discriminator network is made of fully-connected layers that operate on down-sampled point features. In Centerpoint, the discriminator makes use of 2D convolutions with inputs from BEV feature maps. The most prominent distinction between the two networks is the point-based and view-based representations, which are handled by MLPs and 2D convolutions respectively. The ability to adapt

both of these models between the domains shows the modularity of our solution and adaptivity to different methods. With UADA3D, Centerpoint is trained for 40 and IA-SSD for 80 epochs using one-cycle Adam with a learning rate 0.003 and 0.01 respectively. The GRL-coefficient was set to a fixed value of 0.1. For all the SOTA methods, we train the source-only and oracle models with 40 (Centerpoint) and 80 (IA-SSD) epochs, then we adapt them for 40 and 80 accordingly.

A. Datasets

We use five datasets: LiDAR-CS [36] that provides multiple LiDAR resolutions (we use VLD-64 "*CS64*", VLD-32 "*CS32*", and VLP-16 "*CS16*"), KITTI [11] "*K*" (with HDL-64E), Waymo [12] "*W*" (HDL-64E), nuScenes [13] "*N*" (VLD-32) and data collected with the last mile delivery robot "*R*" (with VLP-16) in different locations in Europe.

First, we focus on UDA between autonomous driving datasets, from denser to sparser LiDARs, since this is where the other SOTA methods perform the worst. Even though these sensors operate similarly, differences arise from the point-cloud density or LiDARs azimuth and elevation angles. Second, we adapt to our robot data, which is particularly challenging due to different LiDAR positions and densities on the mobile robot, as well as the operating environment, sidewalk vs. street (the dataset will be publicly available). Finally, we adapt from sparser to denser domains and other popular UDA benchmarks used by SOTA.

In Fig. 3 we can see how big the gap between the self-driving datasets and robot data is, especially in distance to the objects (nuScenes and Waymo show a similar distribution to LiDAR-CS and KITTI). The distance to the objects encountered in the robot data, seen in Fig. 3(b), shows that the dataset only contains objects closer than 15 m, with a majority of pedestrians and cyclists closer than 10 m and all vehicles in the 10-15 m range, since the robot drives on a sidewalk. In contrast, in the self-driving cars dataset Fig. 3(a), objects are between 20-30 m away. Consequently, the average number of points per object (Fig. 4) is very different between the datasets. This is causing the instances to significantly differ between these domains. While the different vehicle sizes are often addressed in 3D domain adaptation approaches [20], to our knowledge, the differing number of points per object has not been examined by other UDA works before. KITTI dataset is limited by having labels only in the front camera FOV, while other datasets feature labels in a 360-degree FOV. This presents a significant challenge for domain adaptation.

B. Evaluation Metrics

We report mAP_{3D} and mAP_{BEV}, *closed gap* and change in mAP. mAP is the mean average precision over the three classes: Vehicle, Pedestrian, and Cyclist. *Closed gap* [37] is calculated as $\frac{mAP_{model}-mAP_{source}}{mAP_{oracle}-mAP_{source}} \times 100\%$ and tells how close we are to a model trained on the target domain (oracle).

V. Results

In Tables I and II, we compare our method with SOTA on a large number of UDA scenarios between source and target (S → T) data. In Table I we focus on dense to sparse scenarios for the two models: Centerpoint [34] and IA-SSD [35]. Our method, UADA3D, outperforms other SOTA approaches in most cases achieving much higher improvements, especially on the larger

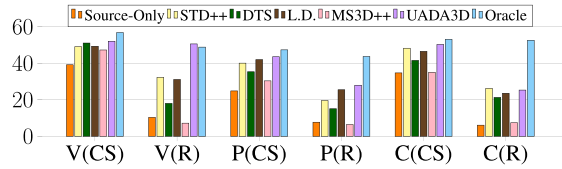


Fig. 5. Per class AP_{3D} in adaptation experiments centerpoint. **V**ehicle, **P**edestrian, and **C**yclist. **UADA3D** is our method.

domain gaps (adapting models towards mobile R(obot) or W→ N). By analyzing *Change* and *Closed Gap* columns, we can observe that Centerpoint shows higher adaptability and improvement (mAP_{3D} and mAP_{BEV}) than IA-SSD. That may come from the fact that, for IA-SSD, we have to fix a specific number of sampling points, which makes the model less flexible when adapting across different domains. Furthermore, even though other SOTA methods may outperform UADA3D on individual classes (only Cyclist in adaptation towards *R*), they perform worse in others categories (Fig. 5). Thus, UADA3D generalizes better across different detectors and classes.

We observe that our method handles diverse domain shifts effectively when adapting between autonomous driving datasets (Waymo, KITTI, nuScenes), simulation data (LiDAR-CS), and robot data (R). As mentioned in Section III, UADA3D does not need a pre-trained teacher model, as all the other approaches do. Instead, we can directly train the domain-adapted model, leveraging the GRL functionality which creates domain-invariant features. This allows our method to successfully train high-performing models on unlabeled target data, without depending on pseudo labels. We can observe in Table I that some of the methods perform even worse than the source-only approaches when tested on adaptation towards more challenging scenarios (e.g K → R), failing to generate accurate pseudo labels or distill teacher knowledge.Additionally, it is important to note that UADA3D never achieves lower performance than source-only models, on the adaptation task, regardless if the domain gap is big (e.g., K → R, W → N) or smaller (e.g., C64 → CS16). Notably, K→ R or CS16→R adaptations appear particularly difficult, as many methods perform worse than source-only, suggesting that adapting to the robot's environment poses distinct challenges. However, despite this, the gap in those cases seems easier to close than the domain gap between W → N, where improvement and *closed gap* were the lowest.

In Table II, we compare methods' performance on sparse to dense cases. UADA3D consistently outperforms other methods in all scenarios. All methods excel when adapting from sparse to dense data, indicating that target data is richer in information, compared to the source. This supports our hypothesis that adapting from dense to sparse is more challenging. Our method performs well on both: sparse to dense and dense to sparse, while some SOTA methods, like L.D. [17], only allow adaptation from dense to sparse LiDAR, highlighting our method's generalizability.

Additional Comparisons: While we use Centerpoint and IA-SSD, our approach works with other detectors. In Table II, we compare SECOND using weights provided by ST3D++, DTS, and MS3D++. The main limitation with using pre-trained models is that they are trained only on one category (car). Results highlight that UADA3D outperforms SOTA in these scenarios as well. Recently released CMDA [38] also uses an

TABLE I

COMPARISON OF PERFORMANCE OF DIFFERENT ADAPTATION METHODS ON SOURCE TO TARGET (S → T) DOMAIN ADAPTATION TASKS. WE REPORT mAP_{3D} AND mAP_{BEV} OVER VEHICLE, PEDESTRIAN AND CYCLIST. THE BEST SCORE IS **BOLD** AND THE SECOND BEST IS UNDERLINE

S→T	Methods	Models					
		IA-SSD [35]			Centerpoint [34]		
		3D/BEV	Change	Closed Gap	3D/BEV	Change	Closed Gap
W ↓ N	Source Only	1.5 / 4.5	-/-	-/-	15.96 / 17.87	-/-	-/-
	ST3D++ [20]	11.94 / 13.15	10.45 / 8.65	23.38 % / 20.09 %	17.66 / 22.83	1.71 / 4.97	4.6 % / 13.45 %
	DTS [18]	15.82 / 17.40	14.32 / 12.90	32.03 % / 29.97 %	16.24 / 17.89	0.28 / 0.02	0.78 % / 0.05 %
	L.D. [17]	17.57 / **19.33**	16.07 / **14.83**	35.95 % / **34.46** %	22.21 / 27.98	6.26 / 10.12	16.96 % / 27.41 %
	MS3D++ [8]	15.10 / 16.61	13.60 / 12.11	30.43 % / 28.15 %	19.71/ 21.69	3.75 / 3.82	10.17 % / 10.35 %
	UADA3D (ours)	**18.33** / 18.55	**16.83** / 14.05	**37.66** % / 32.66 %	**26.89 / 30.67**	**10.94 / 12.80**	**29.65 % / 34.68** %
	Oracle	46.20 / 47.53	-/-	-/-	52.84 / 54.77	-/-	-/-
CS64 ↓ CS16	Source Only	20.11 / 25.26	-/-	-/-	22.33 / 38.08	-/-	-/-
	ST3D++ [20]	30.11 / 40.09	10.00 / 14.83	36.09 % / 48.63	**43.90**/50.42	**21.57** / 12.34	**73.99** % / 57.40 %
	DTS [18]	33.41/42.62	13.29 / 17.36	47.98 % / 56.92 %	39.19 / 47.81	16.86 / 9.73	57.86 % / 45.26 %
	L.D. [17]	29.81 / 35.35	9.70 / 10.10	34.99 % / 33.11 %	42.66/ **54.07**	20.33 / **15.99**	69.76 % / **74.38** %
	MS3D++ [8]	32.79 / 37.21	12.68 / 11.95	45.77 % / 39.17 %	30.98 / 38.53	8.65 / 0.45	29.68 % / 2.08 %
	UADA3D (ours)	**35.32/ 45.53**	**15.21 / 20.27**	**54.87** % / **66.47** %	43.59 / 52.93	21.26/14.85	72.94% / 69.04%
	Oracle	47.82 / 55.76	-/-	-/-	51.48 / 59.58	-/-	-/-
CS64 ↓ R	Source Only	7.28 / 12.93	-/-	-/-	5.38 / 25.97	-/-	-/-
	ST3D++ [20]	31.79 / 42.19	24.50 / 29.26	66.53 % / 74.69 %	23.30 / 33.99	17.92 / 8.01	41.67 % / 23.88 %
	DTS [18]	**33.68 / 46.19**	**26.40 / 33.26**	**71.69** % / **84.90** %	16.87 / 32.13	11.50 / 6.16	26.73 % / 18.34 %
	L.D. [17]	20.70 / 32.10	13.41 / 19.17	36.42 % / 48.94 %	28.13 / 38.90	22.75 / 12.93	52.89 % / 38.51 %
	MS3D++ [8]	×	×	×	14.44 / 37.88	9.06 / 11.90	21.06 % / 35.47 %
	UADA3D (ours)	33.00 / 43.02	25.72 / 30.09	69.83 % / 76.81 %	**28.87 / 40.09**	**23.49 / 14.12**	**54.62** % / **42.07** %
K ↓ R	Source Only	×	×	×	24.40 / 33.81	-/-	-/-
	ST3D++ [20]	×	×	×	31.57 / 35.51	7.17 / 1.70	16.19 % / 4.58 %
	DTS [18]	×	×	×	23.24 / 23.56	-1.16 / -10.25	-2.63 % / -27.69 %
	L.D. [17]	×	×	×	39.46 / 41.47	15.06 / 7.66	33.98 % / 20.71 %
	MS3D++ [8]	×	×	×	21.58 / 25.72	-2.28 / -8.09	-6.36 % / -21.85 %
	UADA3D (ours)	×	×	×	**40.08 / 41.78**	**16.40 / 7.97**	**37.01 % / 21.54 %**
N ↓ R	Source Only	×	×	×	7.45 / 33.39	-/-	-/-
	ST3D++ [20]	×	×	×	13.69 / 38.58	6.24 / 5.19	15.24 % / 19.84 %
	DTS [18]	×	×	×	11.25 / 34.75	3.8 / 1.36	9.29 % / 5.20 %
	L.D. [17]	×	×	×	16.00 / 38.50	8.55 / 5.11	20.89 % / 19.54 %
	MS3D++ [8]	×	×	×	15.51 / 36.63	8.06 / 3.24	19.70 % / 12.39 %
	UADA3D (ours)	×	×	×	**16.77 / 38.73**	**9.32 / 5.34**	**22.76 % / 20.42 %**
CS32 ↓ R	Source Only	8.33 / 15.93	-/-	-/-	3.78 / 6.00	-/-	-/-
	ST3D++ [20]	21.57 / 31.04	13.25 / 15.11	37.02 % / 41.77 %	26.57 / 40.68	22.80 / 34.69	51.09 % / 64.79 %
	DTS [18]	15.37 / 26.94	7.05 / 11.01	19.69 % / 30.45 %	18.68 / 29.65	14.91 / 23.65	33.41 % / 44.18 %
	L.D. [17]	19.43 / 31.38	11.10 / 15.45	31.02 % / 42.70 %	26.89 / 36.85	23.12 / 30.85	51.82 % / 57.62 %
	MS3D++ [8]	×	×	×	4.45 / 42.46	0.68 / 36.46	1.52 % / 68.09 %
	UADA3D (ours)	**22.15/ 31.90**	**13.82 / 15.97**	**38.62** % / **44.14** %	**31.54/ 42.82**	**27.76 / 36.83**	**62.23** % / **68.78** %
CS16 ↓ R	Source Only	10.28 / 15.26	-/-	-/-	15.07 / 33.97	-/-	-/-
	ST3D++ [20]	9.58 / 29.83	-0.7 / 14.56	-2.07 % / 39.53 %	33.22 / 42.62	18.15 / 8.65	54.46 % / 33.84 %
	DTS [18]	11.95 / 27.59	1.67 / 12.32	4.93 % / 33.45 %	17.35 / 34.34	2.28 / 0.37	6.84 % / 1.45 %
	L.D. [17]	9.48 / 23.27	-0.81 / 8.01	-2.38 % / 33.45 %	27.13 / 40.19	12.06 / 6.22	36.19 % / 24.32 %
	MS3D++ [8]	×	×	×	2.16 / 32.30	-12.91 / -1.67	-38.75 % / -6.52 %
	UADA3D (ours)	**19.71 / 35.69**	**19.71 / 20.42**	**27.87** % / **55.44** %	**41.47 / 46.86**	**26.40 / 12.89**	**79.24** % / **50.43** %
Robot	Oracle (R)	44.11 / 52.10	-/-	-/-	48.39 / 59.54	-/-	-/-

TABLE II

COMPARISON OF PERFORMANCE OF DIFFERENT ADAPTATION SCENARIOS: CENTERPOINT ON SPARSE TO DENSE AND VARIOUS SCENARIOS ON SECOND [39], PREVIOUSLY USED BY SOTA AUTHORS. $mAP_{3D/BEV}$ REPORTED OVER 3 CLASSES: VEHICLE, PEDESTRIAN, AND CYCLIST ON CENTERPOINT, AND AP ON VEHICLE/CAR CLASS FOR SECOND. *NOTE: FOR SECOND WE USE WEIGHTS PROVIDED BY THE AUTHORS

	N →K* (SECOND [39])		W →N*(SECOND [39])		W →K*(SECOND [39])		N →W (Centerpoint [34])		N →K (Centerpoint [34])	
	3D/BEV	Closed Gap	3D/BEV	Closed Gap	3D/BEV	Closed Gap	3D/BEV	Closed Gap	3D/BEV	Closed Gap
Source	17.98/51.84	-/-	17.44/33.02	-/-	27.55/67.61	-/-	19.61/37.76	-/-	17.18/40.18	-/-
ST3D++	64.65/77.89	84.14% / 82.83%	22.03/36.71	26.34% / 19.56 %	**73.66**/83.37	91.08%/82.83%	29.37/40.07	26.32%/10.89%	20.37/44.02	11.66%/36.99%
DTS	64.97/80.23	84.71 % / 90.27%	22.83/40.62	30.92% / 40.29 %	71.18/83.91	91.08%/82.83%	32.33/41.10	34.31%/15.77%	18.90/46.20	6.30%/57.95%
MS3D++	X	X	23.15/44.01	32.75% / 58.28 %	X	X	21.82/39.92	5.97%/10.19%	18.00/40.61	2.98%/4.41%
UADA3D	**65.91/82.44**	86.41% / 97.30%	**24.84/45.51**	42.44% / 66.21 %	73.41/**86.53**	95.74%/96.14%	**35.68/42.78**	**43.33%/23.72%**	**21.97/48.52**	**17.51%/80.35%**
Oracle	73.45/83.29	-/-	34.87/51.88	-/-	75.45/87.29	-/-	56.69/58.96	-/-	44.51/50.56	-/-

adversarial component, however, it is trained on multimodal data, while we only use LiDAR. We can compare our results in Table II with CMDA results with SECOND reported on car: N-K (68.95/82.13) and W-N (24.64/42.81). Despite CMDA using additional modality, UADA3D performs on par with CMDA N-K and outperforms it on W-N.

A. Ablation Studies

First, we ask the question of which probability distribution alignment is the most beneficial for UADA3D. Next, we investigate the impact of discriminator design and analyze GRL parameters. Finally, we integrate other self-learning components to enhance our method further.

Probability Distribution Alignments: We explore the effectiveness of other probability distribution alignment strategies, shown in Fig. 6. In UADA3D with marginal distribution alignment, UADA3D$_{\mathcal{L}m}$, the discriminator gradient is backpropagated only to the feature extractor with loss $\mathcal{L}_m$, where $\mathcal{L}_m = \frac{1}{N}\sum_{n=1}^{N} d \cdot \log(g_{\theta_m}(X_n)) + (1-d) \cdot \log(1 - g_{\theta_m}(X_n))$, where d is 0 for source and 1 for target domains, and g_{θ_M} denotes the marginal discriminator network. In UADA3D (i.e., UADA3D$_{\mathcal{L}c}$) the discriminator gradient is backpropagated to the detection head through the whole model. UADA3D$_{\mathcal{L}mc}$ combines marginal and conditional alignment with $\mathcal{L}_{mc} = (\mathcal{L}_m + \mathcal{L}_C)$, where $\mathcal{L}_C$ is given in (2). UADA3D$_{\mathcal{L}m}$ uses feature maps directly, without features masking, to predict the domain and calculate the loss, while UADA3D

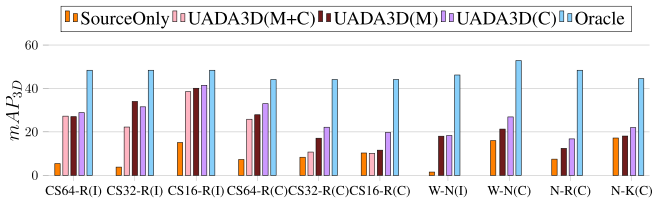
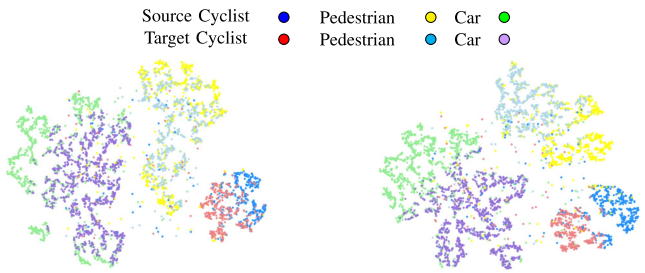


Fig. 6. Comparison UADA3D performance with different probability distribution alignments on IA-SSD (I) and centerpoint (C).



(a) UADA3D (b) UADA3D$_{\mathcal{L}m}$

Fig. 7. T-SNE analysis for N → robot with centerpoint.

TABLE III
CONTRIBUTION OF CLASS-CONFIDENCE AND MULTIPLE DISCRIMINATORS (AP_{3D}/AP_{BEV}). N_D IS A NUMBER OF CLASS-WISE DISCRIMINATORS, AND y DENOTES IF WE USE CLASS LABELS

	M	N_D	y	CS64 →CS16			CS64 →robot		
				$V_{3D/BEV}$	$P_{3D/BEV}$	$C_{3D/BEV}$	$V_{3D/BEV}$	$P_{3D/BEV}$	$C_{3D/BEV}$
IA	S			29.97 / 38.10	11.02 / 14.62	19.35 / 23.05	1.17 / 5.77	6.42 / 18.10	14.26 / 14.92
	(a)	1		42.55 / 61.42	17.36 / 32.85	35.29 / 39.14	52.20 / 59.60	8.28 / 20.65	15.47 / 23.37
[35]	(b)	1	✓	40.92 / 61.61	17.75 / 32.51	34.99 / 39.81	**59.85 / 67.66**	11.32 / 25.70	14.01 / 16.51
	(c)	3	✓	**42.96 / 62.08**	**25.08 / 34.09**	**37.91 / 40.42**	53.29 / 63.93	**14.20 / 28.58**	**31.51 / 36.55**
Cent	S			32.30 / 48.08	11.63 / 20.09	23.06 / 46.07	1.11 / 48.16	3.03 / 14.06	11.99 / 15.70
	(a)	1		**47.95** / 65.84	39.21 / 49.00	45.91 / 47.38	46.94 / 54.89	25.04 / 44.12	39.43 / 44.57
[34]	(b)	1	✓	47.62 / 65.43	**40.28** / 49.42	**46.08** / 47.41	**50.81 /58.72**	**27.68 / 45.39**	**47.88 / 51.56**
	(c)	3	✓	46.99 / **66.01**	38.22 / **49.47**	45.56 / **48.30**	44.53 / 53.11	20.65 / 33.50	21.43 / 33.67

employs masked features with class labels and bounding boxes. UADA3D with conditional probability distribution alignment (and feature masking) consistently yields higher-quality outcomes, however UADA3D$_{\mathcal{L}m}$ delivers comparable results in cross-sensor adaptation for self-driving cars, due to significant differences in the marginal probability distribution.

We also want to bring up the importance of feature masking and per-class discrimination in UADA3D. Using T-SNE analysis [40], we can observe in Fig. 7(a) that class wise discrimination with feature masking helps us achieve class-wise domain invariant features. If we compare this to features obtained with UADA3D$_{\mathcal{L}m}$ in Fig. 7(b), which uses the global features to predict the domain without feature masking we can observe that the class wise features are not well aligned. Consequently, we selected conditional alignment with feature masking as our preferred method.

Discriminator designs: In Table III we examine different discriminator designs: (a) single domain discriminator with the input of $(x_n, \hat{b}_n)$ (features masked by predicted bounding boxes), without label information $\hat{y}$, (b) a single domain discriminator with the input of $(x_n, \hat{b}_n, \hat{y})$, where the discriminator output is multiplied by the maximum predicted class confidence $\hat{y}_{\max,n} \odot g_{\theta_D}(x_n, \hat{b}_n)$, and (c) the default UADA3D setting with input of $(x_n, \hat{b}_n, \hat{y})$ and three class-wise domain discriminators, where the output of each discriminator is multiplied by the corresponding class confidence. (a) and (b) obtain higher adaptation scores in some cases, especially on Centerpoint CS64→R, but worse than (c) on IA-SSD. Comparing (a) and (b), we can see that the multiplication with $\hat{y}_{\max}$ (b) has a substantial impact. Centerpoint obtains better performance using one discriminator than IA-SSD, which is likely due to the easier transferability of BEV features compared to point features. AP_{BEV} scores of IA-SSD also improve more than the AP_{3D} for cases (a) and (b). Due to higher model complexity, IA-SSD appears to gain the most advantage from the use of multiple class-wise conditional discriminators (c). While IA-SSD performs the best with (c) and Centerpoint with (b), to be consistent, we chose option (c) as a default option for our method.

TABLE IV
GRL COEFFICIENTS TESTS OF IA-SSD, FOR *CS64 → CS16*

λ	UADA3D/UADA3D$_{\mathcal{L}m}$		
	$AP_{3D,V}$	$AP_{3D,P}$	$AP_{3D,C}$
Source	29.97	11.02	19.35
0.05	38.13/37.04	23.41/22.13	29.29/26.85
0.1	42.96/**48.48**	25.48/25.08	**38.12/37.91**
0.2	**43.24**/40.34	24.86/24.91	34.24/32.96
$\alpha = 0.1$	42.76/41.06	**26.43/26.80**	33.77/31.83
$\alpha = 0.2$	40.14/38.11	24.55/24.7	34.22/30.01
$\alpha = 0.5$	40.24/37.55	23.58/21.42	33.16/27.43
$\alpha = 1$	41.65/40.09	23.68/22.58	31.72/29.39
Oracle	58.62	35.57	49.27

Gradient reversal coefficient: We tested two different strategies for *GRL*-coefficient λ on UADA3D and UADA3D$_{\mathcal{L}m}$. Firstly, a constant $\lambda = 0.1$ was tested following the setting used for most adversarial UDA strategies in 2D object detection [16], [27]. Secondly, we follow other approaches [3], [15] where λ was increased over the training according to: $\lambda = \alpha(\frac{2}{1+exp(-\gamma p)} - 1)$, where $\alpha \in [0, 1]$ is a scaling factor that determines the final λ, $\gamma = 10$ and p is the training progress from start 0 to finish 1. α-values of $1, 0.5, 0.2, 0.1$ were tested. In this way, the gradient reversal is low for the first few iterations and goes towards α. Given that we do not use a pre-trained model, this approach enables the method to concentrate on the object detection task at the beginning of training, as the discriminator loss will initially be significantly smaller. This performance stems from the characteristics of each category. Pedestrians show more variability in posture and occlusion, while cars and cyclists are more consistent. We hypothesize that dynamic adaptation improves generalization for pedestrians, whereas for cars and cyclists, the impact of λ is less significant. In Table IV, we can observe that smaller λ values result in better adaptation performance. The best score for vehicles and cyclists was achieved with the constant $\lambda = 0.1$ while the dynamic λ achieved the best scores for pedestrians at $\alpha = 0.1$ respectively. Both $\lambda = 0.1$ and $\alpha = 0.1$ yielded good average scores across all three classes with minimal differences. We tested constant $\lambda \in [0.01, 0.5]$ and observed a slow decrease in performance towards source-only values for smaller values and low performance for higher values thus we opted for constant $\lambda = 0.1$ in our experiments.

VI. Conclusion and Limitations

We introduce UADA3D, a novel approach tailored for challenging UDA scenarios, specifically addressing sparser LiDAR data and mobile robots. UADA3D effectively navigates diverse environments, ensuring precise object detection and achieving SOTA performance across various domain adaptation scenarios. This way, the large number of existing data sets in the field of autonomous driving can also be leveraged for mobile robotics. However, domain gaps persist, notably in scenarios such as W $\rightarrow$ N or K $\rightarrow$ R, where improvements are constrained. Future work could explore adapting between multimodal scenarios, such as camera and LiDAR, radar and LiDAR, or adding more self-training components.

References

[1] M. K. Wozniak, V. Kårefjärd, M. Thiel, and P. Jensfelt, "Towards a robust sensor fusion step for 3D object detection on corrupted data," *IEEE Robot. Automat. Lett.*, vol. 8, no. 11, pp. 7018–7025, Nov. 2023.

[2] J. Wei, Y. Lin, and H. Caesar, "BaSAL: Size balanced warm start active learning for LiDAR semantic segmentation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 18258–18264.

[3] J. Li, R. Xu, J. Ma, Q. Zou, J. Ma, and H. Yu, "Domain adaptive object detection for autonomous driving under foggy weather," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2023, pp. 612–622.

[4] Y. Wang et al., "Train in Germany, test in the USA: Making 3D object detectors generalize," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11710–11720.

[5] X. Peng, X. Zhu, and Y. Ma, "CL3D: Unsupervised domain adaptation for cross-LiDAR 3D detection," in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, pp. 2047–2055.

[6] R. DeBortoli, L. Fuxin, A. Kapoor, and G. A. Hollinger, "Adversarial training on point clouds for sim-to-real 3D object detection," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 6662–6669, Oct. 2021.

[7] Z. Luo et al., "Unsupervised domain adaptive 3D detection with multi-level consistency," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 8846–8855.

[8] D. Tsai, J. S. Berrio, M. Shan, E. Nebot, and S. Worrall, "MS3D++: Ensemble of experts for multi-source unsupervised domain adaptation in 3D object detection," *IEEE Trans. Intell. Veh.*, Aug. 12, 2024, doi: 10.1109/TIV.2024.3441527.

[9] 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 *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 3691–3703.

[10] Z. Li, J. Guo, T. Cao, L. Bingbing, and W. Yang, "GPA-3D: Geometry-aware prototype alignment for unsupervised domain adaptive 3D object detection from point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 6371–6380.

[11] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, pp. 1231–1237, 2013.

[12] P. Sun et al., "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2443–2451.

[13] H. Caesar et al., "nuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11618–11628.

[14] M. Thiel, J. Ziegenbein, N. Blunder, M. Schrick, and J. Kreutzfeldt, "From concept to reality: Developing sidewalk robots for real-world research and operation in public space," *Logistics J.: Proc.*, vol. 2023, no. 1, 2023.

[15] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2015, pp. 1180–1189.

[16] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. V. Gool, "Domain adaptive faster R-CNN for object detection in the wild," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3339–3348.

[17] 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 *Proc. Eur. Conf. Comput. Vis.*, Springer, 2022, pp. 179–195.

[18] Q. Hu, D. Liu, and W. Hu, "Density-insensitive unsupervised domain adaption on 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 17556–17566.

[19] C. Huang, V. Abdelzad, S. Sedwards, and K. Czarnecki, "SOAP: Cross-sensor domain adaptation for 3D object detection using stationary object aggregation pseudo-labelling," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2024, pp. 3340–3349.

[20] J. Yang, S. Shi, Z. Wang, H. Li, and X. Qi, "ST3D++: Denoised self-training for unsupervised domain adaptation on 3D object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 6354–6371, May 2023.

[21] C. Saltori, S. Lathuiliere, N. Sebe, E. Ricci, and F. Galasso, "SF-UDA^{3D}: Source-free unsupervised domain adaptation for LiDAR-based 3D object detection," in *Proc. IEEE Int. Conf. 3D Vis.*, 2020, pp. 771–780.

[22] D. Tsai, J. S. Berrio, M. Shan, S. Worrall, and E. Nebot, "See eye to eye: A LiDAR-agnostic 3D detection framework for unsupervised multi-target domain adaptation," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 7904–7911, Jul. 2022.

[23] L. Kong, N. Quader, and V. E. Liong, "ConDA: Unsupervised domain adaptation for LiDAR segmentation via regularized domain concatenation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 9338–9345.

[24] J. Yuan et al., "Bi3D: Bi-domain active learning for cross-domain 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 15599–15608.

[25] Z. Zhang et al., "Pseudo label refinery for unsupervised domain adaptation on cross-dataset 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 15291–15300.

[26] Y. Zhang, C. Zhou, and D. Huang, "STAL3D: Unsupervised domain adaptation for 3D object detection via collaborating self-training and adversarial learning," *IEEE Trans. Intell. Veh.*, May 06, 2024, doi: 10.1109/TIV.2024.3397194.

[27] K. Saito, Y. Ushiku, T. Harada, and K. Saenko, "Strong-weak distribution alignment for adaptive object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 6949–6958.

[28] Z. He and L. Zhang, "Multi-adversarial faster-RCNN for unrestricted object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6667–6676.

[29] C.-D. Xu, X.-R. Zhao, X. Jin, and X.-S. Wei, "Exploring categorical regularization for domain adaptive object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11721–11730.

[30] T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez, "ADVENT: Adversarial entropy minimization for domain adaptation in semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2512–2521.

[31] M. Jaritz, T.-H. Vu, R. d. Charette, E. Wirbel, and P. Pérez, "xMUDA: Cross-modal unsupervised domain adaptation for 3D semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 12602–12611.

[32] W. Zhang, W. Li, and D. Xu, "SRDAN: Scale-aware and range-aware domain adaptation network for cross-dataset 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 6765–6775.

[33] T. Feng et al., "Open compound domain adaptation with object style compensation for semantic segmentation," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2024, vol. 36.

[34] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3D object detection and tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 11779–11788.

[35] Y. Zhang, Q. Hu, G. Xu, Y. Ma, J. Wan, and Y. Guo, "Not all points are equal: Learning highly efficient point-based detectors for 3D LiDAR point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 18931–18940.

[36] J. Fang, D. Zhou, J. Zhao, C. Tang, C.-Z. Xu, and L. Zhang, "LiDAR-CS dataset: LiDAR point cloud dataset with cross-sensors for 3D object detection," 2023, *arXiv:2301.12515*.

[37] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "STD: Sparse-to-dense 3D object detector for point cloud," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1951–1960.

[38] G. Chang et al., "CMDA: Cross-modal and domain adversarial adaptation for LiDAR-based 3D object detection," in *Proc. AAAI Conf. Artif. Intell.*, 2024, vol. 38, pp. 972–980.

[39] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, 2018, Art. no. 3337.

[40] L. V. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, 2008.