2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

CTS: Sim-to-Real Unsupervised Domain Adaptation on 3D Detection

Meiying Zhang[1], Weiyuan Peng[1], Guangyao Ding[2], Chenyang Lei[2], Chunlin Ji[3], Qi Hao[2]

***Abstract*— Simulation data can be accurately labeled and have been expected to improve the performance of data-driven algorithms, including object detection. However, due to the various domain inconsistencies from simulation to reality (sim-to-real), cross-domain object detection algorithms usually suffer from dramatic performance drops. While numerous unsupervised domain adaptation (UDA) methods have been developed to address cross-domain tasks between real-world datasets, progress in sim-to-real remains limited. This paper presents a novel Complex-to-Simple (CTS) framework to transfer models from labeled simulation (source) to unlabeled reality (target) domains. Based on a two-stage detector, the novelty of this work is threefold: 1) developing fixed-size anchor heads and RoI augmentation to address size bias and feature diversity between two domains, thereby improving the quality of pseudo-label; 2) developing a novel corner-format representation of aleatoric uncertainty (AU) for the bounding box, to uniformly quantify pseudo-label quality; 3) developing a noise-aware mean teacher domain adaptation method based on AU, as well as object-level and frame-level sampling strategies, to migrate the impact of noisy labels. Experimental results demonstrate that our proposed approach significantly enhances the sim-to-real domain adaptation capability of 3D object detection models, outperforming state-of-the-art cross-domain algorithms, which are usually developed for real-to-real UDA tasks.**

I. Introduction

Unsupervised domain adaptation (UDA) research in 3D object detection has yielded outstanding results in various real-world datasets [1]–[8]. By contrast, the sim-to-real domain adaptation has not made much progress yet. This is primarily due to the point cloud generated in commonly used simulation environments, such as CARLA [9], have limitations including: 1) ideal and densely collected with minimal noise; 2) significant statistical disparities from real–world data, as simulated assets are limited in types and sizes; and 3) insufficient diversity in object features. These limits degrade the sim-to-real domain adaptation performance in 3D object detection.

Generally, UDA methods in 3D object detection can be divided into two main categories: 1) domain-invariant feature learning [1]–[4], which learns domain-invariant features by minimizing the distance of feature distribution between the source and target domains; 2) pseudo-label guided methods [5]–[8], which enhance transfer performance by generating pseudo-labels in the target domain and further training using these labels. While the former requires specific feature information of two domains, the latter provides a more general and flexible cross-domain framework. However, these methods are not directly applicable to sim-to-real scenarios. A fully functional pseudo-label guided approach to sim-to-real UDA should be able to address the following issues:

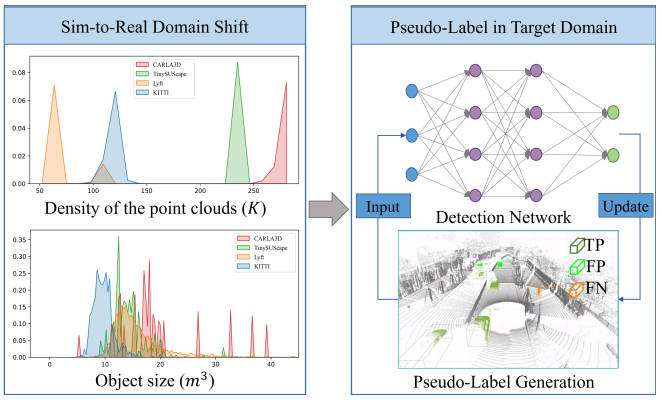

Fig. 1: An illustration of unsupervised sim-to-real domain adaptation guided by pseudo-label, which aims to minimize domain shifts arising from the simulator (*e.g.*, CARLA [9]) to the real-world datasets (*e.g.*, KITTI [10], Lyft [11] and TinySUSCape [12]).

- **Generation of High-quality Pseudo-label.** The object size bias and distribution differences between the simulated and real data, as shown in Fig 1, easily lead to inconsistent regression results (*i.e.*, low-quality pseudo-labels). How to mitigate these biases in detection is important for generating high-quality pseudo-labels.
- **Uniform Quantification of Pseudo-label Quality.** The generated pseudo-labels include true positive (TP), false positive (FP), and false negative (FN), as shown in Fig. 1. In general, TP labels have high quality, while FP ones have low quality and FN ones are missing labels. How to uniformly quantify the quality of pseudo-labels is critical for subsequent sampling of high-quality labels.
- **Target Data Sampling with High-quality Pseudo-labels.** In most UDA methods guided by pseudo-labels, all pseudo-labels are packaged into the target domain training stage. However, FP and FN pseudo-labels introduce extra noise into this process and degrade model performance. How to smartly sample the target data with high-quality pseudo-labels is crucial to improve cross-domain performance.

Meiying Zhang and Weiyuan Peng are co-first authors; Corresponding author: Qi Hao (e-mail: hao.q@sustech.edu.cn)

[1] Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology (SUSTech), China; [2] Department of Computer Science and Engineering, SUSTech; [3] Kuang-Chi Institute of Advanced Technology, China.

This work is jointly supported by the National Natural Science Foundation of China (62261160654), the Shenzhen Fundamental Research Program (JCYJ20220818103006012, KJZD20231023092600001), and the Shenzhen Key Laboratory of Robotics and Computer Vision (ZDSYS20220330160557001).

To reduce the domain gap arising from object bias, current methods primarily focus on point cloud preprocessing in the source domain. However, these methods can barely reduce domain inconsistencies between two domains [7], [8], [13]. Furthermore, methods that use a complex two-stage UDA design show limited performance in sim-to-real tasks [6], [13]. Meanwhile, various methods have been proposed to achieve high-quality pseudo-label guidance, including multi-output fusion techniques, such as fusing multi-modality outputs for 2D-3D data [12], or fusing multi-pass outputs to maintain "high stochastic" [14]. The mean teacher scheme can also generate more accurate pseudo-labels in target domains [6], [14], [15]. However, its performance can be much degraded by the data noise in sim-to-real tasks.

This paper proposes a mean teacher-based Complex-to-Simple (CTS) framework, focusing on the second stage design, for sim-to-real UDA, with novel techniques to mitigate object bias, enhance pseudo-label quality, and optimize target domain data sampling for pseudo-label guidance. The main contributions include:

- Development of localization refinement techniques including RoI random scaling and fixed-size anchor heads to address domain inconsistencies and produce high-quality pseudo-labels.
- Development of a uniform corner-format measure for aleatoric uncertainty (AU) estimation to evaluate the quality of pseudo-labels accurately.
- Development of two AU-based sampling strategies in the mean teacher domain adaptation process to select point cloud frames and labels with adequate quality.
- Release of CTS code, alongside the CARLA3D simulated dataset, for further research[1].

II. Related Work

A. UDA for 3D object detection

Some previous works have well explored the usage of UDA in 3D object detection [6]–[8], [13], [14]. One common challenge of UDA in 3D object detection is the object size bias when cross-domains. Wang et al. [13] propose statistical normalization (SN) to align object sizes utilizing statistical information from target domain data. ST3D [7] and ST3D++ [8] employ data augmentations during source domain training to improve the model's incorporation of diverse size information. Besides mitigating object size bias, using pseudo-label guided methods in UDA emphasizes improving the quality of pseudo-labels. JST [12] enhances pseudo-label quality through 2D and 3D joint refinement, aligning outcomes from both modalities. ST3D [7] integrates an additional IoU regression head to assess prediction quality, facilitating selective updates of the pseudo-label pool. Building upon ST3D, ST3D++ [8] further refines pseudo-labels using a quality-aware denoising pipeline. MLC-Net [6] also employs the mean teacher scheme to ensure target domain consistency between teacher and student modules at both point and instance levels, which is similar to our method but involves higher complexity using UDA design for both stages. Although having significant improvements in real-to-real tasks, existing UDA methods often experience serious performance degradation in sim-to-real tasks. Therefore, based on the analysis of simulation and reality differences, our study concentrates on the quality enhancement, evaluation and selection for pseudo-labels to achieve higher sim-to-real performance.

B. Uncertainty Estimation in 3D Object Detection

Uncertainty can serve as a valuable metric for quantifying both data and model noise within deep neural networks (DNNs) [16]–[20]. Uncertainty estimation methods typically address two main sources: epistemic uncertainty (EU) and aleatoric uncertainty (AU). EU is represented by a posterior distribution over model parameters [16], [17], [19], providing insights into the models' uncertainty; AU is represented a distribution over model outputs [18], [20], reflecting intrinsic data stochastic. Notably, AU varies with the quality of input data, suitable for quantifying the noise level of input data. Within the context of 3D detection tasks, several methodologies have integrated aleatoric uncertainty (AU) due to its ability to enhance detection performance [21], [22]. Meyer et al. [21] employ a mixture of Laplace distributions to fit the variances for each predefined regression variable, including box center positions, sizes, and orientation. Feng et al. [22] model AU using multivariate Gaussian distributions, with independent variables representing three distinct sets, *i.e.*, RoI positions, bounding box positions, and orientation. However, few methods have leveraged the Aleatoric Uncertainty (AU) estimated from 3D detection results for the evaluation of data noise. Besides, existing approaches represent uncertainties using non-uniform variables, adding a complexity to further utilization. Therefore, this study proposes a uniform corner-based representation for bounding boxes with uncertainties, easy for the quality evaluation of the predicted pseudo-labels.

III. System Setup

In a standard two-stage detector like PointRCNN [23], the first stage roughly detects objects across a frame and the second stage refines localization. Directly applying PointRCNN for sim-to-real tasks led to a 60% decrease in Average Precision (AP) at an IoU threshold of 0.7 and a 20% decrease at IoU of 0.5 (see CARLA3D→ KITTI in Table I), which suggests a retained object detection and classification ability but much loss in localization precision. To enhance sim-to-real domain adaptation, the paper simply focuses on improving the domain adaptation of the second-stage localization network instead of adopting a complex two-stage UDA design, namely Complex-to-Simple (CTS).

The complete CTS framework is illustrated in Fig. 2. This framework utilizes simulated data from the source domain to initially develop detection capabilities, followed by model refinement through mean teacher-based domain adaptation in real-world traffic scenarios of the target domain. The mean-teacher method [5] mitigates the impact of domain shift

[1]The code of CTS and CARALA3D dataset are available at `https://github.com/tendo518/CTS-UDA`

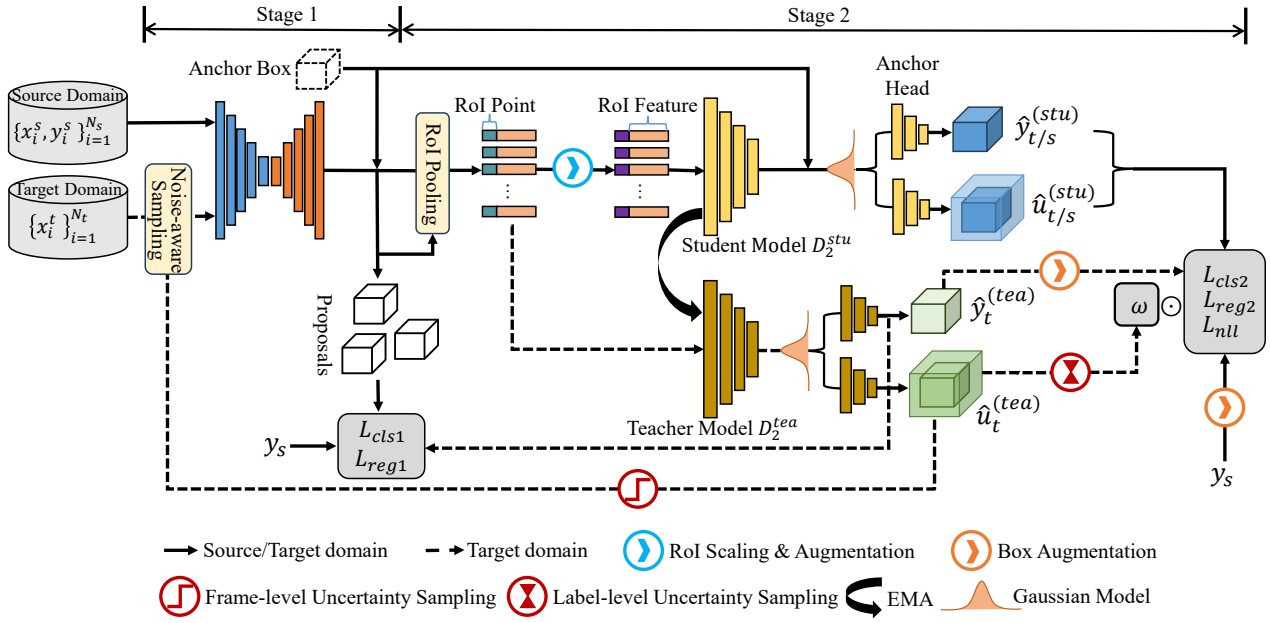

Fig. 2: An illustration of the CTS framework. In the first stage, the model is trained on the source domain with *Anchor Head* (Sec IV-A.1), *RoI Augmentation* (Sec IV-A.2) and *corner-format AU* modeling (Sec IV-B). In the second stage, the *noise-aware* mean teacher approach is applied: the student model is alternatively supervised with pseudo-labels on the target domain and ground-truth labels on the source domain; the teacher model's weights are updated using the EMA. Meanwhile, two *noise-aware* sampling strategies (Sec IV-C) are implemented using the aleatoric uncertainty indicator: frame-level sampling removes noisy frames, while object-level soft-sampling handles noisy labels.

by training the student model with a consistency objective, effectively utilizing unlabeled data from the target domain to improve the model's performance in that domain.

The framework consists of two branches: the student and teacher models. Both models share the same architecture and are initialized with parameters trained on the source domain. However, they are updated through different mechanisms:

1) Student Model: The student model utilizes augmented RoI points and features as input, supervised with pseudo-labels $\hat{y}_t$ in the target domain or ground truth labels y_s in the source domain. It is worth noting that the generated pseudo-labels can serve as supervision for the 1st-stage network as well, thus enabling domain adaptation for the 1st-stage network. Thus, the total loss of this network includes: 1) first-stage RoI regression loss l_{reg1}. 2) first-stage RoI classification loss l_{cls1}. 3) second-stage regression Smooth-L1 loss l_{reg2}. 4) second-stage classification loss l_{cls2}. 5) second-stage AU-NLL loss l_{nll} specified in Sec IV-B.

2) Teacher Model: The teacher model processes raw (non-augmented) data and maintains fixed weights during the backward pass. Instead of employing standard backpropagation with predefined loss functions, the teacher model updates its weights using exponential moving average (EMA) from student model as follows:

$$\theta_t^{tea} = \beta \times \theta_{t-1}^{tea} + (1-\beta) \times \theta_t^{stu} \tag{1}$$

Here, θ_t^{stu} represents the student model's weights at iteration t, β is the EMA decay factor that controls the update rate, and θ_t^{tea} denotes the teacher model's weights. The resulting teacher model's weights provide a smoothed representation of the student model's weights over time.

IV. Proposed Methods

A. Enhancement of Pseudo-Label Quality

CTS starts by training a detector in a labeled source domain and then leverages this knowledge to generate high-quality pseudo-labels in the target domain. However, simulation-reality differences, such as differences in object size and point density, present significant challenges. Specifically, size bias has been shown to significantly reduce localization accuracy [13], especially when in simulation-to-reality scenarios, where expanding the simulation model asset library (e.g., through CAD modeling) to match the target domain is both difficult and expensive. To address these domain shifts and improve the reliability of pseudo-labels in the target domain, we propose the following methodology:

1) Anchor Head (AH): The second-stage model typically predicts size residuals Δ_{whl} between proposals from the first-stage and final bounding boxes, denoted as $\hat{B}$. This approach avoids regressing the size of bounding boxes entirely from scratch. However, a challenge arises when the first-stage model, trained with biased supervision from source domain labels, exhibits inaccuracy in estimating proposal sizes. Unreliable proposal box sizes can lead to size errors accumulating in the second stage, degrading final bounding box refinement accuracy and the effectiveness of pseudo-labels. Inspired by anchor-based detectors [24], we introduce

a fixed-size anchor box w_{an}, h_{an}, l_{an} to replace the proposal, termed the *anchor head (AH)*. The AH replaces the traditional proposal mechanism, allowing the second-stage network to work with a globally fixed-size 3D anchor instead of refining proposals. By employing the AH in both the source and target domains, we ensure consistent behavior of the second-stage network regarding proposal size refinement across domains, thus avoid size error propagation in UDA and enhancing the quality of pseudo-labels.

2) RoI Random Scaling (RRS) and Augmentation: To enhance the diversity in the features of the learning object from the simulated data, we introduce RoI Random Scaling (RRS) and Augmentation. In our setup, the second-stage model utilizes localized points (RoI points) and corresponding RoI features from the first-stage model as inputs. Specifically, only the points undergo augmentation, while their features remain unchanged. Let $\widetilde{X} \in \mathbb{R}^{3\times N}$ denote the decentralized points within a RoI box of dimensions l, w, h, and let q_l, q_w, q_h represent random scaling factors. The scaled RoI sizes are derived by multiplying the original dimensions by the scaling factors, resulting in $q_l l, q_w w, q_h h$. Furthermore, to enhance the second-stage model's robustness, we apply augmentations that involve random rotation, flipping, and translation within specified ranges, as described in [25].

B. 3D Detection with Aleatoric Uncertainty

As noted in [16], Deep Neural Networks (DNNs) are capable of predicting aleatoric uncertainty effectively. Specifically, in the case where the regression y follows a Gaussian distribution with parameters (μ, σ^2), the following loss function $\mathcal{L}_{nll}$ can be employed for optimization:

$$\mathcal{L}_{nll} = \frac{(y - f_\mu(\mathbf{x}, \theta))^2}{2 f_{\sigma^2}(\mathbf{x}, \theta)} + \frac{1}{2}\log(f_{\sigma^2}(\mathbf{x}, \theta)) \tag{2}$$

where θ is the model parameter, f_μ and f_{σ^2} represent subnetworks for predicting the mean and the variance.

When training the regression part of the detector, since the predicted bounding box y is usually encoded with 7 values, i.e., $\mathbf{y}_b = \{\mu_{bx}, \mu_{by}, \mu_{bz}, \mu_{bh}, \mu_{bw}, \mu_{bl}, \mu_{b\alpha}\}$ (called box format, BF), the matched variance values are encoded primarily as $\boldsymbol{\sigma}_b^2 = \{\sigma_{bx}^2, \sigma_{by}^2, \sigma_{bz}^2, \sigma_{bh}^2, \sigma_{bw}^2, \sigma_{bl}^2, \sigma_{b\alpha}^2\}$, of which each element corresponding to the uncertainty of an element in the bounding box representation. Nevertheless, the BF bounding box regression variable, specifically the centroid positions, extents (length, width, height), and orientations exhibits numerical magnitude inconsistencies. These disparities also indicate varying magnitudes of variances across each variable. Applying reduction methods (such as maximum or average) to these variances naively may result in overlooking uncertainties arising from specific components, particularly the orientation, due to its significantly smaller magnitudes.

Inspired by the corner loss methodology [26], we introduce a corner-based uncertainty measurement by encoding the bounding box equally with its 8 corner points, as illustrated in Fig 3. To be specific, during the training process, we firstly perform corner transformation on both model-predicated BF box and corresponding ground truth:

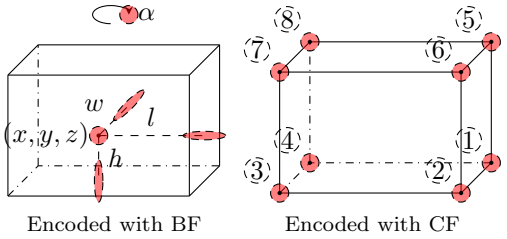

Fig. 3: An illustration of two coding schemes of bounding boxes with uncertainties. (a) BF: box format; (b) CF: corner format, where the red areas stand for the potential ranges, that is, the aleatoric uncertainty.

$$\begin{bmatrix}\mu_{cx}^i \\ \mu_{cy}^i \\ \mu_{cz}^i\end{bmatrix} = R_z(\mu_{b\alpha}) \times \begin{bmatrix}\pm\frac{\mu_{bw}}{2} \\ \pm\frac{\mu_{bh}}{2} \\ \pm\frac{\mu_{bl}}{2}\end{bmatrix} + \begin{bmatrix}\mu_{bx} \\ \mu_{by} \\ \mu_{bz}\end{bmatrix} \tag{3}$$

Where $R_z(\mu_{b\alpha})$ represents the rotation matrix corresponding to the yaw angle $\mu_{b\alpha}$, and $p_c^i = \mu_{cx}^i, \mu_{cy}^i, \mu_{cz}^i{}_{i=1}^{8}$ denotes the positions of the 8 corners of the transformed CF-encoded box. For the sake of regression simplification, we assume that the distribution of each corner's coordinates follows distinct Gaussian all sharing the same variance, denoted as:

$$\mathbf{y}_c^i = \begin{bmatrix}y_{cx}^i \\ y_{cy}^i \\ y_{cz}^i\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\mu_{cx}^i \\ \mu_{cy}^i \\ \mu_{cz}^i\end{bmatrix}, (\sigma_c^i)^2\mathbf{I}\right), i = 1\ldots 8 \tag{4}$$

where $\mathbf{I}$ is the identity matrix. Consequently, we predict 8 (rather than 24) independent variances $(\sigma_c^2)^i$ for a CF encoded box, the overall NLL loss $\mathcal{L}$ and aleatoric uncertainty $\hat{u}$ can be easily reduced with:

$$\mathcal{L}_{nll}^i = \frac{\left(\overline{\mathbf{y}_c^i} - \overline{\hat{\mathbf{y}}_c^i}\right)^2}{2(\sigma_c^i)^2} + \frac{1}{2}\log(\sigma_c^i)^2 \tag{5}$$

Where final NLL loss $\mathcal{L}_{nll} = \frac{\sum_{i=1}^{8}\mathcal{L}_{nll}^i}{8}$ and uncertainty of box $u_{box} = \frac{\sum_{i=1}^{8}(\sigma_c^i)^2}{8}$. And all components contribute equally to the loss and final uncertainty metric.

C. Noise-aware Mean Teacher

Aligning transformations on both student-model inputs and teacher-model output facilitates the acquisition of domain-invariant representations, thereby aiding in adaptation to the target domain using pseudo-labels. However, noisy pseudo-labels can lead to error accumulation. To address this challenge, we leverage aleatoric uncertainties predicted by a model to annotate data in the target domain and mitigate the impact of noisy data during mean teacher domain adaptation with the following sampling strategies:

1) Object-Level Soft Sampling: During each iteration, the final second-stage regression loss L_{reg2} is computed using the supervision provided by pseudo-labels assigned to individual objects. Rather than solely depending on these

pseudo-labels, the loss is weighted by the inverse of their uncertainty u, denoted as:

$$\mathbf{w}_{label} = \{\frac{1}{u} \mid \forall u \in \hat{\mathbf{u}}_{tea}\} \qquad l_2 = \mathbf{w}_{label} \odot \mathbf{l}_2 \quad (6)$$

Where $\mathbf{l}_2$ is the second-stage loss produced per object in the whole point cloud frame, $\odot$ is the element-wise product. Consequently, objects with higher uncertainty associated with their pseudo-labels are softly filtered out, mitigating the adverse effects of noisy objects.

2) Frame-Level Sampling: Instead of using all target data, the sampling process selects a subset based on frame-level uncertainty. Low-noise target frames are sampled to train the model, enhancing its ability to detect objects in the target domain. By integrating curriculum learning strategies [27], the model refines its pseudo-labels and becomes more confident in uncertainty estimates after several training epochs. This iterative process gradually includes more frames until eventually, all target data are sampled. A detailed explanation of the frame-level sampling refers to Algorithm 1.

Algorithm 1 Noise-aware Frame-Level Sampling

Input:
$\mathcal{T}$: Unlabeled Target Domain Dataset
$\mathcal{T}_{sub}$: Target Sub-dataset after Sampling
N_t: Number of samples in $\mathcal{T}$
N_{sub}: Amount of data to be selected
Output: $\mathcal{D}$: Noise-aware Model

while $N_{sub} < N_t$ **do**
  $U_{frame} \leftarrow \{\}$, $\mathcal{T}_{sub} \leftarrow \{\}$
  for each frame x^t in $\mathcal{T}$ **do**
    $\hat{\mathbf{y}}^t, \hat{\mathbf{u}}^t \leftarrow$ inference $\mathcal{D}$ for x^t
    $\hat{u}^t \leftarrow$ mean of $\hat{\mathbf{u}}^t$ for all valid object in x^t
    $U_{frame} \leftarrow$ append $\hat{u}_t$ to U_{frame}
  end for
  for i in $\{1, \ldots, N_{sub}\}$ **do**
    $j \leftarrow$ argmin of U_{frame}
    $\mathcal{T}_{sub} \leftarrow$ append the j_{th} element x_j^t in $\mathcal{T}$ to $\mathcal{T}_{sub}$
    pop the j_{th} element $\hat{u}_j^t$ from U_{frame}
  end for
  $\mathcal{D} \leftarrow$ fine-tune $\mathcal{D}$ with $\mathcal{T}_{sub}$
  $N_{sub} += N_{sub}$
end while
return $\mathcal{D}$

V. Experiments

A. Experimental Setup

1) Datasets: Most existing LiDAR simulation datasets are primarily used for task-specific problems, such as Vehicle-to-Vehicle Communication [28] and Continuous Domain Shift [29], rather than for sim-to-real UDA as addressed in this paper. Thus, we conduct supervised training in a simulated source domain, namely CARLA3D, acquired within the CARLA simulator [9] from scratch. All samples are taken from eight built-in scenarios in CARLA to ensure data diversity. The ego-vehicle is positioned randomly, collecting about 100 samples per scenario, each comprising eight frames at 2Hz. Out of the eight frames per sample, five are randomly chosen for the training set, yielding 3,990 frames with a total of 25,192 objects. Further details of the CARLA3D dataset are outlined in Table II. The target domains chosen include KITTI [10], Lyft [11], and TinySUScape used in [12]. During the testing phase, samples from these datasets along with their corresponding labels will be utilized, whereas only samples will be used during the training phase. A summary of these datasets is presented in Table III.

2) Evaluation Metric: In our 3D object detection evaluation, referring to [13], we utilize the official KITTI evaluation metric from [10] for the *Car* category. We report two average precision (AP) metrics: AP_{BEV} based on bird's-eye view IoUs, and AP_{3D} based on 3D IoUs.

3) Implementation Details: Our proposed method is implemented based on OpenPCDet [25], using PointRCNN [23] as our baseline detector. All experiments were conducted on a Ubuntu Linux server equipped with 12 GiB NVIDIA TITAN V GPUs. The proposed model is first trained in CARLA3D for 50 epochs, in which the learning rate, the weight decay, and the momentum are set as 0.005, 0.0001, and 0.9, respectively. For the anchor head configuration, the anchor dimensions are globally set to $l_{an} = 3.9$, $h_{an} = 1.6$, and $w_{an} = 1.56$. These values are derived from the statistical average of the dimensions of all labeled car objects in the KITTI dataset, deemed a reasonable metric. RoI augmentation is applied, involving random scaled by a factor of range from 0.7 to 1.3, translated by up to ± 0.5 meter, rotated by an angle between $-\frac{\pi}{4}$ and $\frac{\pi}{4}$, and flipped by a chance of 50%. During mean teacher domain adaptation, the model achieving the highest accuracy in the source domain training phase is selected, and both teacher and student models are initialized from it. The Exponential Moving Average (EMA) factor (β) is set to 0.999, and the training lasts for 30 epochs for the Lyft dataset and 50 epochs for the KITTI/TinySUScape datasets. To ensure stability, we train the student model by alternating between source (with ground-truth labels) and target (with pseudo-labels) domain data. Regarding noise-aware training settings, the uncertainty pool is refreshed at the 1st, 6th, 16th, and 21st epochs for the Lyft dataset and at the 1st, 11th, 21st, and 31st epochs for the KITTI and TinySUScape datasets. In each of these epochs, sub-datasets are resampled at percentages of 30%, 50%, 70%, and 100% of the total dataset size for subsequent training iterations.

B. Main Results

Our CTS framework was compared with the following methods: 1) **SN** [13]: A domain adaptation method has been considered effective on various datasets; 2) **MLC-Net** [6]: A domain adaptation method also based on mean teacher, which is similar to ours in the mean teacher part; 3) **ST3D++** [8]: A recent self-training based method that achieved state-of-the-art performance in real-to-real (*e.g.*, Nuscenes [30] $\rightarrow$ KITTI [10]) domain adaptation tasks.

Besides, we provide two possible boundaries of results, they are: 1) **Source Only**: The model is solely trained in

Task	Method	AP_{BEV}@0.7			AP_{3D}@0.7		
		Easy	Moderate	Hard	Easy	Moderate	Hard
CARLA3D→Lyft	Source Only	66.70	54.35	51.76	18.82	13.85	13.64
	SN [13]	66.92	53.31	50.52	23.05	16.79	15.99
	MLC-Net [6]	77.95	64.46	62.13	53.97	40.04	37.47
	ST3D++ [8]	75.57	61.68	57.49	51.02	37.24	35.41
	Ours	**81.66**	**67.86**	**65.17**	**61.93**	**45.87**	**43.87**
	Oracle	90.92	83.97	81.70	80.06	66.05	64.01
CARLA3D→KITTI	Source Only	27.45	20.55	17.51	5.67	4.06	3.23
	SN [13]	31.21	30.23	28.18	9.37	9.15	7.63
	MLC-Net [6]	70.45	56.66	49.41	43.02	32.68	27.39
	ST3D++ [8]	64.50	54.91	49.75	34.34	27.22	23.99
	Ours	**78.92**	**64.17**	**57.37**	**58.41**	**45.28**	**39.61**
	Oracle	93.18	83.26	80.20	86.02	71.70	66.86
CARLA3D→TinySUScape	Source Only	18.02	16.69	N/A	4.59	3.83	N/A
	SN [13]	27.45	14.96	N/A	1.42	1.36	N/A
	MLC-Net [6]	19.64	18.81	N/A	8.27	7.59	N/A
	ST3D++ [8]	40.86	38.17	N/A	26.09	23.86	N/A
	Ours	**42.45**	**38.62**	N/A	**31.47**	**28.02**	N/A

TABLE I: Comparison results of three different sim-to-real domain adaptation tasks. We report AP_{BEV} and AP_{3D} of the *car* category at IoU = 0.7 for different difficulty levels. As TinySUSCape [12] does not provide labels with the occlusion level, *Hard* is marked as Not Available (N/A).

Scenario	Frames	Easy	Moderate	Hard	Times
Town01	800	309	798	1572	100
Town02	800	577	898	1983	100
Town03	800	581	1574	3471	100
Town04	792	555	3167	5978	99
Town05	800	695	1727	3855	100
Town06	800	229	445	2495	100
Town07	800	251	758	1967	100
Town10	792	823	1648	2998	99
Total	6384	4020	11015	24319	798

TABLE II: Overview of CARLA3D dataset. *Frames* represents the number of point cloud frames sampled in the scenario; *Easy*, *Moderate*, and *Hard* represent the quantities of objects with different difficult levels in the scenario, respectively. *Times* refers to the number of sampling.

Dataset	Size(Train/Test)	LiDAR Beams	Points Per Frame
CARLA3D	3990 / 2394	1×64	$286.2K$
KITTI [10]	3712 / 3769	1×64	$118.7K$
Lyft [11]	12017 / 2891	1×40 *or* 64	$72.3K$
TinySUScape [12]	2579 / 965	1×128	$230.4K$

TABLE III: A summary of datasets. The *Size(Train/Test)* refers to the number of samples used in training and testing.

AH	Aug2	MT	NLL	FL-NA	OL-NA	mAP_{3D}
						15.44
✓						34.63
✓	✓					43.51
✓	✓		CF			43.83
✓	✓	✓				45.67
✓	✓	✓	CF			45.91
✓	✓	✓	CF	✓		46.47
✓	✓	✓	CF		✓	48.67
✓	✓	✓	BF	✓	✓	49.37
✓	✓	✓	CF	✓	✓	**50.56**

TABLE IV: Ablation study results on CARLA3D → Lyft. *AH*: anchor head scheme proposed in Sec IV-A.1; *Aug2*: second-stage augmentation in Sec IV-A.2; *MT*: mean teacher based domain adaptation; *NLL*: usage of NLL loss for aleatoric uncertainty; *CF* and *BF* refer to corner-format and box-format encoding respectively in Sec IV-B; *FL-NA* and *OL-NA*: frame-level and object-level noise-aware sampling strategies respectively in Sec IV-C. The mAP_{3D} metric is obtained by averaging over the three difficulty levels.

a supervised manner on the source domain and is directly applied to the target domain without employing any domain adaptation methods, which serve as a lower bound; 2) **Oracle**: A fully supervised model trained on the target/reality domain with actual labels, considered as an upper bound.

The results obtained using different UDA methods are summarized in Table I. Our CTS method surpasses all others in sim-to-real detection tasks. Specifically, compared to the *source only* method, our approach improves AP_{BEV} by approximately $15\%-35\%$ and AP_{3D} by around $25\%-50\%$. However, due to the significant domain shift between the simulator and reality, our CTS method still exhibits a noticeable gap compared to the supervised *Oracle*. In contrast, the SN method, which generally performs well in various real-world domains, struggles in sim-to-real cross-domain tasks, experiencing performance degradation, such as in the CARLA3D → TinySUSCape scenario.

C. Ablation Study

To further demonstrate the effectiveness of the individual components in our proposed method, we conducted extensive ablation experiments on the CARLA3D → Lyft task.

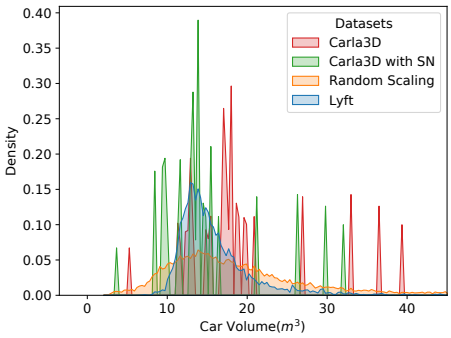

Fig. 4: An illustration of the car sizes distribution of Lyft [11], and CARLA3D datasets with different processing methods, *i.e.*, SN [13] and Random Scaling.

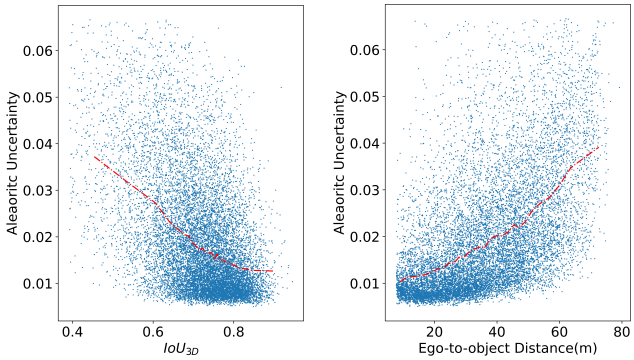

Fig. 5: An illustration of the correlation between AU value and IoU/ego-to-object distance for the target dataset. *Blue* points denote the AU values of detected objects; the *red* line represents the means of the AU values.

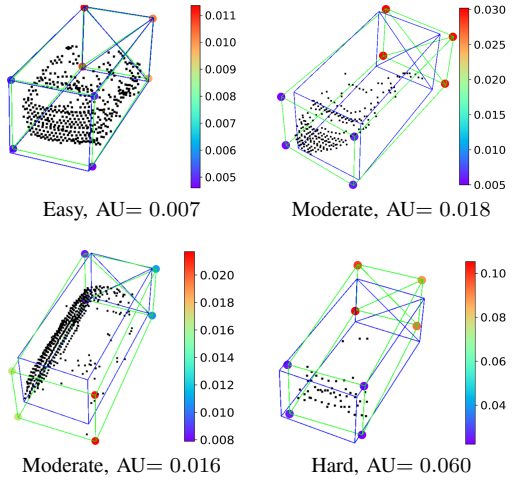

Fig. 6: Examples of different levels of difficulties in 3D boxes. The *blue* boxes represent the ground truth; the *green* boxes represent the predicted results. The points in different colors at the box corners represent the 8 AU value components, whose mean is the final AU value of the entire object.

1) Benefits of Anchor Head: Incorporating the anchor head (AH) into the second-stage detector effectively reduces regression complexity while enhancing cross-domain robustness. As described in Table IV, compared to the original setup, the AH scheme yields over 19% improvement, highlighting its effectiveness in cross-domain tasks even with a simple anchor size replacement.

2) Benefits of RRS and Second-stage Augmentation: Compared to SN's approach [13], our RoI Random Scaling (RRS) method effectively encourages the sizes of processed objects to resemble an unimodal distribution similar to real-world data, rather than solely aligning with statistical volumes that still exhibit multi-modal, as illustrated in Figure 4. Furthermore, integrating RRS into our second-stage augmentation (Aug2) resulted in a performance improvement of approximately 9%, as demonstrated in Table IV. These augmentation techniques enhance data diversity at the object level, enabling the model to learn diverse information.

3) Benefits of Corner-Format AU: In contrast to BF, CF encoding uniformly distributes the localization uncertainty of the object across each corner component without requiring additional operations. Table IV demonstrates that using BF and CF representations for noise-aware sampling improves performance by 3.7% and 4.9%, respectively. This suggests that CF is more effective in identifying reliable pseudo-labels. Employing the CF encoding scheme, we investigate the aleatoric uncertainties (AUs) associated with predicted objects, considering their Intersection over Union (IoU) with ground truths and their ego-to-object distance, as depicted in Figure 5. Our observations reveal a decrease in AU values with increasing IoU, while they increase with greater ego-to-object distance. Furthermore, Figure 6 showcases examples where sparse and corrupted point clouds lead to elevated AU. These findings underscore the efficacy of predicted AUs in evaluating pseudo-label noise and their utility as a reliability metric for pseudo-labels.

4) Benefits of Noise Awareness in Mean Teacher: As mentioned in Sec IV-C, two diverse noise-aware sampling strategies are used to minimize the adverse impacts of noisy pseudo-labels generated during mean teacher domain adaptation. with both the *frame-level noise-aware* (FL-NA) and *object-level noise-aware* (OL-NA) strategies, performance improves by 4.65%.

Additionally, utilization of NLL loss function solely has been shown to bring improvement [22]. Table IV also indicates a minor increase from 43.51% to 43.81% in source-only training with NLL. However, While adding NLL loss and extra uncertainty layers yields only a 0.3% improvement, employing both FL-NA and OL-NA results in an extra significant improvement of 4.3%. This demonstrates that the main performance gain arises from noise-aware sampling strategies rather than just loss function replacement.

D. Limitations

Although our proposed model shows enhanced adaptation performance within the target domain via multiple schemes, sim-to-real UDA still lags behind real-to-real methods due to limitations inherent in simulators. The restricted vehicle

assets in simulators like CARLA fail to represent the diverse range of real-world vehicles. Additionally, simulators struggle to replicate complex real-world scenarios, including dynamic traffic patterns and diverse urban landscapes (e.g., different weather conditions), thus limiting their effectiveness in providing realistic training data for domain adaptation.

VI. Conclusion

This paper has introduced a CTS framework for unsupervised domain adaptation (UDA) in 3D object detection, bridging the gap between simulation and real-world domains. The proposed techniques, including RoI random scaling and augmentation, along with the fixed-size anchor head, enhance the diversity of simulation data and address object size discrepancies across domains, thereby improving the quality of pseudo-labels. Additionally, the proposed aleatoric uncertainty (AU) estimation, based on a uniform corner-format representation of bounding boxes, facilitates the integration of pseudo-label noise awareness into the mean teacher domain adaptation process, leading to high-quality pseudo-label sampling. Experimental results on the CARLA, KITTI, Lyft, and TinySUScape datasets demonstrate substantial improvements over existing methods in various sim-to-real UDA tasks, with 5%-17% gains in AP_{3D} and 2%-10% gains in AP_{BEV}. Future work will focus on extending this approach to cover both sim-to-real and real-to-real UDA scenarios, as well as incorporating additional categories (e.g., bicycles, pedestrians) in domain adaptation.

References

[1] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International Conference on Machine Learning*. PMLR, 2015, pp. 1180–1189.

[2] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[3] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International Conference on Machine Learning*. PMLR, 2015, pp. 97–105.

[4] M. Chen, S. Zhao, H. Liu, and D. Cai, "Adversarial-learned loss for domain adaptation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 04, 2020, pp. 3521–3528.

[5] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," *Advances in neural information processing systems*, vol. 30, 2017.

[6] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 8866–8875.

[7] J. Yang, S. Shi, Z. Wang, H. Li, and X. Qi, "St3d: Self-training for unsupervised domain adaptation on 3d object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 10 368–10 378.

[8] J. Yang, S. Shi, Z. Wang, H. Li, and X. Qi, "St3d++: Denoised self-training for unsupervised domain adaptation on 3d object detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 5, pp. 6354–6371, 2022.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Conference on Robot Learning*. PMLR, 2017, pp. 1–16.

[10] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[11] R. Kesten, M. Usman, T. P. J. Houston, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. O. a. S. Shah, A. Kulkarni, A. Kazakova, L. P. C. Tao, W. Jiang, and a. V. Shet, "Lyft level 5 perception dataset 2020," 2019.

[12] G. Ding, M. Zhang, E. Li, and Q. Hao, "Jst: Joint self-training for unsupervised domain adaptation on 2d&3d object detection," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 477–483.

[13] Y. Wang, X. Chen, Y. You, L. E. Li, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao, "Train in germany, test in the usa: Making 3d object detectors generalize," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 713–11 723.

[14] D. Hegde, V. Sindagi, V. Kilic, A. B. Cooper, M. Foster, and V. Patel, "Uncertainty-aware mean teacher for source-free unsupervised domain adaptive 3d object detection," *arXiv preprint arXiv:2109.14651*, 2021.

[15] J. Deng, W. Li, Y. Chen, and L. Duan, "Unbiased Mean Teacher for Cross-Domain Object Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4091–4101.

[16] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *Advances in neural information processing systems*, vol. 30, 2017.

[17] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *International Conference on Machine Learning*. PMLR, 2016, pp. 1050–1059.

[18] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7482–7491.

[19] N. Tagasovska and D. Lopez-Paz, "Single-model uncertainties for deep learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[20] D. J. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.

[21] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, "Lasernet: An efficient probabilistic 3d object detector for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 677–12 686.

[22] D. Feng, L. Rosenbaum, F. Timm, and K. Dietmayer, "Leveraging heteroscedastic aleatoric uncertainties for robust real-time lidar 3d object detection," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 1280–1287.

[23] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 770–779.

[24] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.

[25] O. D. Team, "OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds," 2020. [Online]. Available: https://github.com/open-mmlab/OpenPCDet

[26] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[27] X. Wang, Y. Chen, and W. Zhu, "A survey on curriculum learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, pp. 4555–4576, 2021.

[28] R. Xu, H. Xiang, X. Xia, X. Han, J. Li, and J. Ma, "Opv2v: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2583–2589.

[29] T. Sun, M. Segu, J. Postels, Y. Wang, L. Van Gool, B. Schiele, F. Tombari, and F. Yu, "Shift: a synthetic driving dataset for continuous multi-task domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 21 371–21 382.

[30] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 621–11 631.