Exploiting Playbacks in Unsupervised Domain Adaptation for 3D Object Detection in Self-Driving Cars

Yurong You*†, Carlos Andres Diaz-Ruiz*‡, Yan Wang†, Wei-Lun Chao§,
Bharath Hariharan†, Mark Campbell‡, Kilian Q Weinberger†

***Abstract*— Self-driving cars must detect other traffic participants like vehicles and pedestrians in 3D in order to plan safe routes and avoid collisions. State-of-the-art 3D object detectors, based on deep learning, have shown promising accuracy but are prone to over-fit domain idiosyncrasies, making them fail in new environments—a serious problem for the robustness of self-driving cars. In this paper, we propose a novel learning approach that reduces this gap by fine-tuning the detector on high-quality pseudo-labels in the target domain – pseudo-labels that are automatically generated *after driving* based on replays of previously recorded driving sequences. In these replays, object tracks are smoothed forward and backward in time, and detections are interpolated and extrapolated—crucially, leveraging future information to catch hard cases such as missed detections due to occlusions or far ranges. We show, across five autonomous driving datasets, that fine-tuning the object detector on these pseudo-labels substantially reduces the domain gap to new driving environments, yielding strong improvements detection reliability and accuracy.**

***Index Terms*— Object Detection, Segmentation and Categorization; Computer Vision for Automation; Transfer Learning; Deep Learning for Visual Perception**

I. Introduction

Detecting traffic participants such as cars, cyclists, and pedestrians in 3D is a fundamental learning problem for self-driving cars. Typically, inputs consists of LiDAR point clouds and/or images; outputs are sets of tight 3D bounding boxes that envelop detected objects. The problem is challenging, because the detection must be highly accurate and reliable. The current state of the art in 3D object detection is based on deep learning approaches [1], [2], [3], [4], trained on short driving segments with labeled bounding boxes [5], [6], which yield up to 80% average precision on held-out segments [4].

However, as with all machine learning approaches, these techniques succeed when the training and test data distributions match. One way to ensure this is to constrain self-driving cars to a small geo-fenced area, such as with a fleet of similar self-driving taxis collecting and sharing training data about the same area. This approach, however, is not generalizable to consumer self-driving cars which should be able to drive freely anywhere, similar to a human-driven car. This unconstrained scenario introduces an inherent adaptation problem: the car producer cannot foresee where the owner will ultimately operate the car. For example, the perception system might be trained on urban roads in Germany [5], [6], but the car may be driven in the mountain roads in the USA, where cars are larger and fewer, roads may be snowier, and the environment (trees, roads, etc.) may look different. Past work has shown that such differences can cause $>35\%$ drop in detection accuracy [7]. Closing this adaptation gap is one of the biggest challenges for consumer self-driving vehicles.

Formally, this challenge is a problem in *unsupervised domain adaptation (UDA)* [8]: the detector, having been previously trained on labeled data from a *source* domain, must now adapt to a *target* domain where only unlabeled data are available. While the UDA problem is easily cast with training and testing datasets in different locations (Germany vs USA, urban vs rural, etc.), similar problems exist in many self-driving car scenarios. For example, consider the common case of car owners who drive many of the same routes (e.g., commuting), and leave their cars parked (e.g., at night) for extended periods of time. This raises an intriguing possibility: the car can collect sensor data on these trips, and then retrain itself autonomously *offline* to adapt to this new environment, to improve subsequent *online* driving.

In this paper, we present a novel approach for UDA of 3D detectors for self-driving cars. Our approach uniquely uses two key insights. First, data collected over time via a *video* is not simply a bag of independent frames. Second, the *dynamics* of our objects of interest (i.e., cars) can be modeled effectively. We propose to use time correlations between frames and object physics to enable more accurate and efficient solutions to the UDA problem *offline*. More specifically, our approach takes the 'confident' detections of nearby objects, estimates their states (e.g., locations, sizes and speeds), and then *extrapolates* the tracks forward and backward in time, discovering challenging cases when the detector made mistakes (e.g. missed detections due to occlusions, detections at far ranges, etc.) The *playback* of the data allows us to go back in time and annotate labels in frames which were previously missed. Although this process cannot be performed in real time (since it uses future information), we can use it *offline* to generate a new training set with pseudo-labels for the target environment. We can then adapt the detector to the target domain using this newly created dataset, thus allowing the detector to generalize to more scenarios. We call our approach *dreaming*, as the car learns by replaying past driving sequences backwards and forwards, potentially while it is parked overnight.

We evaluate our approach on the most challenging of

∗ Equal contributions
† Computer Science Department, Cornell University {yy785, yw763, bh497, kqw4}@cornell.edu
‡ Mechanical and Aerospace Engineering Department, Cornell University {cad297, mc288}@cornell.edu
§ Department of Computer Science and Engineering, the Ohio State University chao.209@osu.edu

UDA problems, that of 3D object detection across multiple autonomous driving datasets, including KITTI [5], [6], Argoverse [9], Lyft [10], Waymo [11], and nuScenes [12]. We show *across all dataset combinations* that discovering detector mistakes and retraining the detector with our dreaming car procedure strongly reduces the source/target domain gap with high consistency. In fact, the resulting detector after "dreaming" *substantially exceeds* the accuracy of the offline system used to generate the pseudo-labels, which—although able to look into the future—is limited to the extrapolation of confident detections before adaption. Our dreaming procedure can easily be implemented on-device and we believe that it constitutes a significant step towards safely operating autonomous vehicles without geo-restrictions.

II. Related Work

3D object detection can be categorized based on the input: using 3D sensors like LiDAR or 2D images from cameras [13], [14], [15], [16]. We focus on the former due to its higher accuracy. Examples of LiDAR-based detectors include F-PointNet [1], PointRCNN [2], PIXOR [3] and VoxelNet [17]. While these methods have consistently improved the detection accuracy, it has been recently revealed in [7] that they cannot generalize well when trained and tested on different datasets, especially on distant objects with sparse LiDAR points.

Unsupervised domain adaptation (UDA) has been widely studied in machine learning and computer vision, especially on image classification [8], [18], [19]. The common setup is to adapt a model trained from one labeled source domain (e.g., synthetic images) to another unlabeled target domain (e.g., real images). Recent work has extended UDA to driving scenes, but mainly for 2D object detection [20], [21], [22], [23], [24] and semantic segmentation [25], [26], [27], [28], [29]. The mainstream approach is to match the feature distributions or image appearances between domains, e.g., via adversarial learning [18], [25] or image translation [30]. The approaches more similar to ours are [31], [32], [33], [34], [29], [35], [36], [37], [38], which iteratively assign pseudo-labels to (some of) the unlabeled data in the target domain and retrain the models. This procedure, usually named self-training, is proven to be effective in learning with unlabeled data, such as semi-supervised and weakly-supervised learning [39], [40], [41], [42]. For UDA, self-training enables the model to adapt its features to the target domain in a supervised fashion.

UDA in 3D tackles the domain discrepancy in point clouds. Qin et al. [43] were the first to match point cloud distributions between domains, via adversarial learning. However, they considered point clouds of isolated objects, which are very different from the ones captured in driving scenes. Other approaches project 3D points to the frontal or bird's-eye view and apply UDA methods in the resulting 2D images [44], [45], [46], which can be sub-optimal in models' accuracy.

Instead, *we follow the self-training paradigm and show that UDA in 3D object detection can be drastically improved by our high-quality pseudo-labels.* Concurrent work ST3D [47] also tackles UDA in 3D via self-training, but it focuses on addressing the object size discrepancy across domains [7]. It pre-trains a 3D detector with random object scaling and uses pseudo-labels to train student detectors in a curriculum-learning way. In contrast, our work makes use of consecutive LiDAR scans in the target domain to improve pseudo-label quality, especially for faraway or previously missed objects. Our work is thus orthogonal and complementary to theirs.

Leveraging videos for object detection has been explored in [48], [49], [50], [51] to ease the labeling efforts by mining extra 2D bounding boxes from videos. The main idea is to leverage the temporal information to extend weakly-labeled instances or potential object proposals across frames, which are then used as pseudo-labels to retrain the detectors. In the context of UDA, [52], [31], [53] also incorporate object tracks to discover high quality 2D pseudo-labels for self-training.

Our approach is different in two aspects. First, we not only interpolate but also *extrapolate* tracks to infer object locations when they are too far away to be detected accurately. Second, we operate in 3D. We apply a physical-based motion model and exploit the fact that objects in 3D are scale-invariant to correct the detection along tracks. In contrast, the methods above operate in 2D and may disregard faraway objects (that appear too small in images) due to unreliable 2D tracks [53].

Auto-labeling. Our work is also related to concurrent work in 3D auto-labeling [54], [55], which improves the initial detections of a 3D object detector in an offline manner by aggregating them across the whole sequence with a standard tracker [56]. Our work is different from theirs in two aspects. First, our approach not only aggregates detections in a track, but *extrapolates* them with a motion model, which is crucial to recover the missing detections faraway (Table V). Second, our end-goal is to improve the 3D object detector by fine-tuning it with the improved pseudo-labels, while auto-labeling solely focuses on improving pseudo-labels.

III. Exploiting Playbacks for UDA

Similar to most published work on 3D object detection for autonomous driving [1], [2], [3], [4], [57], we focus on *frame-wise* 3D detectors. A detector is first trained on a source domain and then applied to a target domain (e.g., a new city). [7] revealed a drastic accuracy drop in such a scenario: many of the target objects are either misdetected or mislocalized, especially if they are far away. To aid adaptation, we assume access to an unlabeled dataset of *video sequences* in the target domain, which could simply be recordings of the vehicle's sensors while it was in operation. Our approach is to generate pseudo-labels for these recordings that can be used to adapt the detector to the new environment during periods when the car is not in use. We assume no access to the source data when performing adaptation—it is unlikely that car producers share data with the customers after the detector is deployed.

A. Tracking for improved detection

One way to improve the test accuracy based on the frame-wise detection outputs is *online* tracking by detection [58], [59], [60]. Here, detected objects are associated across current and *past* frames to derive trajectories, which are used to filter

out false positives, correct false negatives, and adjust the initial detection boxes in the current frame.

Online 3D object tracking. We investigate this idea with a Kalman filter based tracker [61], [62], [56], which has shown promising results in benchmark tracking leader boards [12]. *We opt to not use a learning-based tracker [63] since it would also require adaptation before it can be applied in the target domain.* Specifically, we apply the tracker in [61]. The algorithm estimates the joint probability $p(\mathbf{a}_k, \mathbf{x}_k|\mathbf{z}_k)$ at time k, where $\mathbf{x}_k$ is the set of tracked object states (e.g., cars speeds and locations), $\mathbf{z}_k$ is the set of observed sensor measurements (here each measurement is a frame-wise detection), and $\mathbf{a}_k$ is the assignment of measurements to tracks. The joint probability can be decomposed into continuous estimation $p(\mathbf{x}_k|\mathbf{a}_k, \mathbf{z}_k)$ and discrete data assignment $p(\mathbf{a}_k|\mathbf{z}_k)$. The former is solved recursively via an Extended Kalman Filter (EKF); the latter is solved via Global Nearest-Neighbor (GNN), where the cost to be minimized is the negative BEV IoU between the predicted box and the measurement. The EKF parameterizes the state $\mathbf{x}$ of a single (ith) object as a vehicle (position, velocity, shape) *relative* to the ego-vehicle

$$\mathbf{x}_k^i = \begin{bmatrix} x & y & \theta & s & l & w \end{bmatrix}^T, \tag{1}$$

where x, y are the location of the tracked vehicle's back axle relative to a fixed point on the ego-vehicle, θ is the vehicle orientation relative to the ego-vehicle, s is the absolute ground speed, and l, w are the length and width. The EKF uses a dynamics model of the evolution of the state over time. Here we assume that the tracked vehicle is moving at a *constant* speed and heading in the global coordinate frame, with added noise to represent the uncertainty associated with vehicle maneuvers. This tracker has been shown to work well on tracking moving objects from a self-driving car [61], [64]. We initialize a new track when $c_{\min-\text{hits}}$ measurements of the same tracked object are realized. We end a track when it does not obtain any measurement updates over $c_{\max-\text{age}}$ frames, or the tracked object exits the field of view (FOV). As will be seen in subsection IV-B, applying this tracker can indeed improve the detection accuracy online, via inputting missed detections, correcting mislocalized detections, and rejecting wrong detections in $\mathbf{z}_k$ at current time k by $\mathbf{x}_k$.

Offline 3D object tracking. Online trackers only use past information to improve current detections. By relaxing this for offline tracking (e.g., to be able to look into the future and come back to the current time), we can obtain more accurate estimates of vehicle states. While such a relaxation is not applicable during test time, higher accuracy tracking on unlabeled driving sequences will be very valuable for adapting the source detector to the target domain in a self-supervised fashion, as we will explain in the following section.

B. Self-training for UDA

Self-training is a simple yet fairly effective way to improve a model with unlabeled data [39], [40], [41], [65]. The basic idea is to apply an existing model to an unlabeled dataset and use the high confidence predictions (here detections), which are likely to be correct, as "pseudo-labels" for fine-tuning. One key to success for self-training is the quality of the pseudo-labels. In particular, we desire two qualities out of the detections we use as pseudo-labels. First, they should be *correct*, i.e., they should not include false positives. Second, they should have *high coverage*, i.e., they should cover all cases of objects. Choosing high confidence detections as pseudo-labels satisfies the first criterion but not the second. For 3D object detection, we find that most of the high confidence examples are easy cases: unoccluded objects near the self-driving car. This is where offline tracking becomes a crucial component to include the more challenging cases (far away, or partially occluded objects) in the pseudo-label pool.

C. High quality pseudo-labels via 3D offline tracking

How do we obtain pseudo-labels for far-away, hard-to-detect objects that the detector cannot reliably detect? We propose to exploit tracking by leveraging two facts in the autonomous driving scenario. First, the available unlabeled data is in the form of *sequences* (akin to videos) of point clouds over time. Second, the objects of interest and the self-driving car move in fairly constrained ways. We will run the object detector on *logged* data, so that we can easily analyze both forwards and backwards in time. The object detector will detect objects accurately only when they are close to the self-driving car. Once detected over a few frames, we can estimate the object's motion either towards the self-driving car or away from it, and then both *interpolate* the object's positions in frames where it was missed, or *extrapolate* the object into frames where it is too far away for accurate detection. We show an example of this procedure in Figure 1. Through dynamic modeling, tracking, and smoothing over time we can correct noisy detections. Further with extrapolation and interpolation, we can recover far away and missed detections.

Concretely, we proposed to augment the following functionalities that utilize the future information into the online tracker introduced in subsection III-A, turning it into an offline tracker specifically designed to improve detection, i.e., generating higher quality pseudo-labels for self-training.

State smoothing. Frame-wise 3D object detectors can generate inconsistent, noisy detection across time (e.g., frame 4, 7 and 9 in Figure 1). The model-based tracking approach in subsection III-A reduces this noise, but we can go further by smoothing tracks back and forth over time, since our data is offline. In this work, we use a fixed-point Rauch-Tung-Striebel (RTS) smoother [66] to smooth the tracked state estimates. Smoothing requires a backward iteration ($k = N, N-1, ...1$) that is performed after the forward filtering, where the a-posteriori state and state error covariance estimates ($\bar{\mathbf{x}}_{k|k}$ and $P_{k|k}$) and the a-priori state and state error covariance estimates ($\bar{\mathbf{x}}_{k+1|k}$ and $P_{k+1|k}$) have been calculated. The smoothed gain, C_k, is obtained from

$$C_k = P_{k|k} F_k^T P_{k+1|k}^{-1}, \tag{2}$$

where F_k is the Jacobian of the dynamics model evaluated at $\bar{\mathbf{x}}_{k|k}$. The smoothed state is then evaluated as

$$\bar{\mathbf{x}}_{k|N} = \bar{\mathbf{x}}_{k|k} + C_k[\bar{\mathbf{x}}_{k+1|N} - \bar{\mathbf{x}}_{k|k}] \tag{3}$$

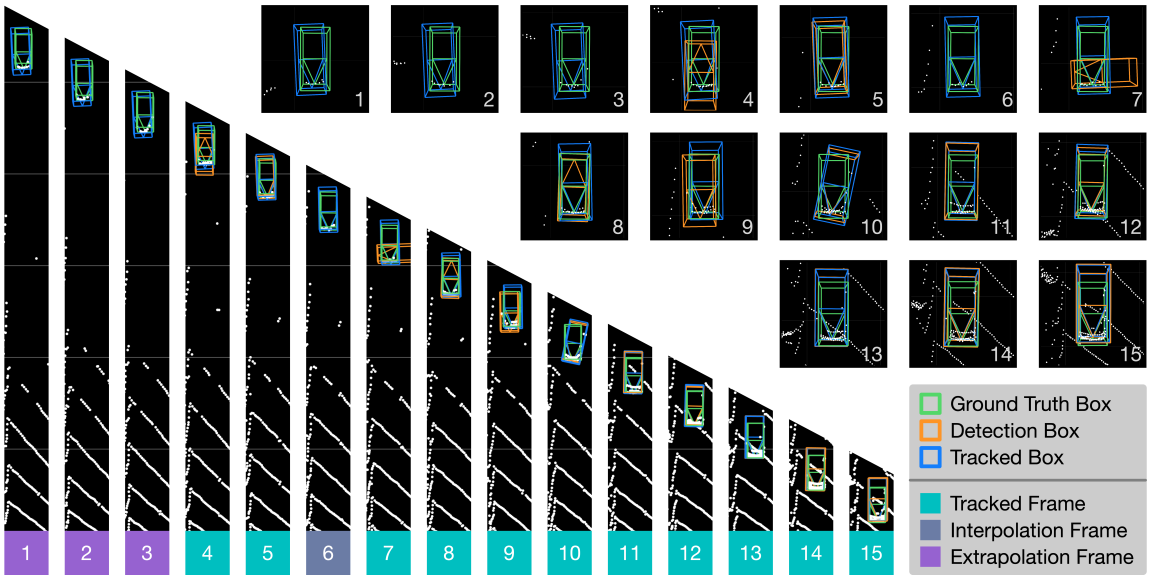

Fig. 1: An example tracked vehicle moves towards ego-vehicle (at the bottom), where pseudo-labels are recovered through extrapolation, interpolation, and smoothing. The improvements of pseudo-labels via offline tracking are instances where estimated bounding boxes (blue) are observed, while the frame-wise detections (orange) are missing or poorly aligned with ground truths (green). Better viewed in color.

while the covariance of the smoothed state is evaluated as

$$P_{k|N} = P_{k|k} + C_k[P_{k+1|N} - P_{k+1|k}]C_k^T. \quad (4)$$

Adjusting object sizes. As shown in [7], the distribution of car sizes in different domains (e.g. different cities) can be different. As such, when tested on a novel domain, detectors often predict incorrect object sizes. This is especially true when the LiDAR signal is too sparse for correct size information. We can also use our tracking to correct such systematic error. Assuming that the most confident detections are more likely to be accurate, we estimate the size of the object by averaging the size of the three highest confidence detections. We use this size for all objects in this track.

Interpolation and extrapolation. We use estimation (forward in time) and smoothing (backward in time) to recover missed detections, and in turn, to increase the recall rate of pseudo-labels (e.g., frame 1–3 and 6 in Figure 1). If a detection is missed in the middle of a track, we restore it by taking the estimated state from smoothing. We also extrapolate the tracks both backward and forward in time, so that tracks that were prematurely terminated due to missing detections, can be recovered. More concretely, we are able to recover detections of vehicles that were lost as they moved away from the ego-vehicle because the sensor signals became sparser (or in turn, the vehicles started far away and were then only detected when they got close enough). Extrapolations are performed by first using dynamics model predictions of the EKF to predict potential bounding boxes; measurements are obtained by performing a search and detection in the vicinity of the prediction. We apply the detector in a 3 m^2 area around the extrapolated prediction, yielding several 3D bounding box candidates. After filtering out candidates with confidences lower than some threshold, we select the candidate with the highest BEV IoU with the prediction as the measurement. If a track loses such a measurement for three consecutive frames, extrapolations are stopped. With this targeted search, we are able to recover objects that were missed due to low confidence. After extrapolating and interpolating detections for all tracks, we perform Non Maximum Suppression (NMS) over bounding boxes in BEV, where more recent extrapolations/interpolations are prioritized.

Discussion. The tracker we apply is standard and simple. We opt it to show the power of our dreaming approach for UDA—exploiting offline, forward and backward information to derive high-quality pseudo-labels for adapting detectors. More sophisticated trackers will likely improve the results further. While we focus on frame-wise 3D detectors, our algorithm can be applied to adapt video-based 3D object detectors [67] as well. One particular advantage of fine-tuning on the pseudo-labeled target data is to allow the detector adapting not only its predictions (e.g., the box regression) but also its features (e.g., early layers in the neural networks) to the target domain. The resulting detector thus can usually lead to more accurate detections than the pseudo-labels it has been trained on.

IV. Experiments

Datasets. We experiment with five autonomous driving data sets: KITTI [5], [6], Argoverse [9], nuScenes [12], Lyft [10] and Waymo [11]. All datasets provide LiDAR data in sequences and ground-truth bounding box labels for either all or part of the data. We briefly summarize these five datasets in the supplementary. We follow a setup similar to [7], but with different splits on Lyft, nuScenes, and Waymo in order to keep the training and test sets non-overlapping with sequences. For nuScenes and Waymo, we only use data from a single location (Boston for nuScenes and San Francisco for Waymo).

UDA settings. We train models in the source domain using labeled frames. We split each dataset into two non-overlapping parts, a training set and a test set. *We use the **train** set (and*

*its pseudo-labels) of the target domain to adapt the source detector, and evaluate the adapted detector on the **test** set.*

Metric. We follow KITTI to evaluate object detection in 3D and BEV metrics. We focus on the *Car* category as it is the main focus of existing work. We report average precision (AP) with the intersection over union (IoU) thresholds at 0.5 or 0.7, i.e., a car is correctly detected if the IoU between it and the detected box is larger than 0.5 or 0.7. We denote AP in 3D and BEV by AP_{3D} and AP_{BEV}, respectively. Because on the other datasets there is no official separation on the difficulty levels like in KITTI, we split AP by depth ranges.

3D object detection models. We use two LiDAR-based models POINTRCNN [2] and PIXOR [3] to detect objects in 3D. They represent two different but popular ways of processing point cloud data. POINTRCNN uses PointNet++ [68] to extract point-wise features, while PIXOR applies 2D convolutions in BEV of voxelized point clouds. Neither relies on images. We mainly report and discuss results of POINTRCNN except for the last study in subsection IV-B.

Hyper-parameters. To train detectors on source domain, we use the hyper-parameters provided by [2] for POINTRCNN. For PIXOR, we follow [7] to train it using RMSProp with momentum 0.9 and learning rate 5×10^{-5} (decreased by a factor of 10 after 50 and 80 epochs) for 90 epochs. For self-training on the target domain, we initialize from the pre-trained model on the source domain. For POINTRCNN, we fine-tune it with learning rate 2×10^{-4} and 40 epochs in RPN and 10 epochs in RCNN. For PIXOR, we use RMSProp with momentum 0.9 and learning rate 5×10^{6} (decreased by a factor of 10 after 10 and 20 epochs) for 30 epochs.

We developed and tuned our dreaming method with Argoverse as the source domain and KITTI as the target domain (in the target domain, we only use the training set). We then fixed all hyper-parameters for all subsequent experiments.

A. Baselines

We compare against two baselines under the UDA setting.

Self-Training (ST). We apply a self-training scheme similar to that typically used in the 2D problems [40]. When adapting the model from the source to the target, we apply the source model to the target training set. We then keep the detected cars of confidence scores > 0.8 (label-sharpening) and use them as pseudo-labels to fine-tune the model. We select the threshold following our hyper-parameter selection procedure and apply it to all the experiments.

Statistical Normalization (SN). [7] showed that car sizes vary between domains: popular cars at different areas can be different. When the mean bounding box size in the target domain is accessible, either from limited amount of labeled data or statistical data, we can apply *statistical normalization (SN)* [7] to mitigate such a systematic difference in car sizes. SN adjusts the bounding box sizes and corresponding point clouds in the source domain to match those in the target domain, and fine-tunes the model on such "normalized" source data, with no need to access target sensor data. We follow the exact setting in [7] to apply SN.

TABLE I: **Pseudo-label quality.** We compare the quality of the pseudo-labels on KITTI **train** set generated by a detector trained on Argoverse (PL) and those after smoothing, resizing, interpolation, and extrapolation (PL (AFTER)). We show the AP_{BEV}/ AP_{3D} of the **car** category at IoU $= 0.7$ and 0.5 across different depth range.

Method	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
PL	**80.5** / 80.0	63.7 / 60.7	23.9 / 18.6	63.9 / 36.6	33.0 / 12.1	10.0 / **0.9**
PL (AFTER)	**80.5 / 80.2**	**68.0 / 61.1**	**29.5 / 22.6**	**64.9 / 38.0**	**38.4 / 15.1**	**12.1 / 0.9**

B. Empirical Results

Pseudo-label quality. In Table I we evaluate the quality of pure pseudo-labels and pseudo-labels after smoothing, resizing, interpolation, and extrapolation under the Argoverse to KITTI setting. It can be seen that the dreaming process significantly improves the pseudo-label quality across all ranges, which leads to further improvement after self-training.

Adaptation from Argoverse to KITTI. We compare the UDA methods under Argoverse to KITTI in Table III and observe several trends: 1) models experience a smaller domain gap when objects are closer (0-30 m vs 30-80 m); 2) though directly applying online tracking can improve the detection performance, models improve more after just self-training; 3) the offline tracking is used to provide extra pseudo-labels for fine-tuning, and interestingly, models fine-tuned from pseudo-labels can outperform pseudo-labels themselves; 4) DREAMING improves over ST and SN by a large margin, especially on IoU at 0.5; 5) DREAMING has a large gain in AP for faraway objects, e.g., on range 50-80 m, compared to ST, it boosts the AP_{BEV} on IoU at 0.5 from 28.2 to 30.1.

Adaptation among five datasets. We further applied our methods to adaptation tasks among the five datasets. Due to limited space, we show the results of AP_{BEV} and AP_{3D} on range 50-80 m and 0-80 m at IoU $= 0.5$ in Table II. Our method consistently improves the adaptation performance on faraway ranges, while having mostly equal or better performance over the full range. We include detailed evaluation results across all ranges, at IoU $= 0.7$, and with SN in the supplementary material. We observe a consistent and clear trend as in Table II.

Adaptation between different locations inside the same dataset. Different datasets not only come from different locations but also use different sensor configurations. To isolate the effects of the former (which is our motivating application), in Table IV we evaluate our method's performance for domain adaptation within the KITTI dataset. In this case, the source and target domain are all scenes from KITTI, while the source is composed of city and campus scenes (38 sequences, 9,556 frames, 3,420 labeled frames) and the target consists of residential and road scenes (23 sequences, 10,448 frames, 3619 labeled frames). Our method consistently outperforms no fine-tuning and ST, especially on 30-80 m range.

Ablation Study. We show ablation results in Table V. Here we fine-tune models using ST and adding smoothing (S), resizing (R), interpolation (I) and extrapolation (E) to the pseudo-label generation. It can be observed that ST alone already boosts performance considerably. Through selecting high confidence detections, smoothing and adjusting the object size we ensure that the pseudo-labels provided are mostly

TABLE II: **Dreaming results on UDA among five auto-driving datasets.** We report AP_{BEV} and AP_{3D} of the Car category on far-away range (50-80m) and full range (0-80m) at IoU= 0.5. On each entry (*row*, *column*), we report AP of UDA from *row* to *column* in the order of No Re-training / Self-Training / Dreaming. At the diagonal entries is the AP of in-domain model. Our method is marked in blue.

(a) 50 – 80 m

AP_{BEV}	KITTI	Argoverse	Lyft	nuScenes	Waymo
KITTI	40.4	20.1/26.9/**28.4**	49.6/56.3/**56.4**	1.4/ **9.1**/ 4.5	42.8/48.5/**50.2**
Argoverse	18.0/28.2/**30.1**	37.8	46.3/48.8/**54.5**	0.6/ **9.1**/ 3.0	50.4/50.9/**56.0**
Lyft	26.1/30.8/**33.9**	29.3/30.7/**35.2**	67.2	4.5/ **9.1**/ 6.0	51.6/51.9/**56.9**
nuScenes	9.6/16.4/**21.7**	3.0/12.4/**17.7**	22.9/39.1/**46.4**	3.5	24.9/42.5/**50.1**
Waymo	14.3/25.6/**27.8**	24.7/23.4/**28.3**	45.3/54.0/**55.5**	0.2/ 3.0/ **9.1**	58.2
AP_{3D}	KITTI	Argoverse	Lyft	nuScenes	Waymo
KITTI	36.3	15.4/19.5/**20.8**	40.0/46.3/**47.7**	**0.9**/ 0.4/ 0.4	33.4/39.3/**41.0**
Argoverse	13.9/22.5/**25.6**	30.0	42.9/46.0/**47.6**	0.1/ 0.2/ **3.0**	47.8/48.2/**49.1**
Lyft	20.4/24.0/**26.4**	25.3/26.7/**27.3**	65.5	0.4/ **9.1**/ 1.1	49.6/50.6/**50.8**
nuScenes	5.5/ 8.7/**13.4**	3.0/ 9.1/**13.0**	15.1/30.2/**37.8**	2.8	22.5/39.6/**45.5**
Waymo	8.5/18.1/**19.7**	19.6/21.3/**22.3**	43.0/48.1/**49.6**	0.0/ 0.3/ **9.1**	50.8

(b) 0 – 80 m

AP_{BEV}	KITTI	Argoverse	Lyft	nuScenes	Waymo
KITTI	87.1	56.8/58.8/**59.4**	68.2/68.3/**68.6**	27.7/27.8/**28.9**	62.5/68.8/**69.0**
Argoverse	82.3/83.2/**83.3**	68.5	66.7/67.0/**67.6**	**26.9**/25.2/25.4	69.8/69.6/**70.0**
Lyft	82.6/84.9/**85.3**	60.2/65.7/**66.0**	79.2	28.8/28.2/**29.1**	70.6/70.9/**71.1**
nuScenes	61.7/76.5/**79.5**	22.4/38.0/**46.6**	41.1/59.0/**65.5**	37.7	51.5/62.2/**68.7**
Waymo	81.2/82.2/**83.1**	56.9/58.3/**59.2**	67.3/68.9/**69.4**	23.1/27.2/**29.1**	71.9
AP_{3D}	KITTI	Argoverse	Lyft	nuScenes	Waymo
KITTI	86.7	49.0/55.0/**55.9**	60.7/65.2/**66.4**	20.9/22.3/**23.1**	59.4/60.5/**61.4**
Argoverse	77.4/**81.1**/**81.1**	65.9	64.9/65.5/**66.4**	**22.8**/22.1/21.7	62.1/62.4/**67.7**
Lyft	77.5/82.2/**82.3**	56.7/58.6/**58.9**	78.3	24.1/23.6/**26.3**	63.0/**69.1**/68.7
nuScenes	45.5/67.5/**70.3**	19.5/36.1/**42.7**	32.4/56.7/**58.6**	36.8	42.6/60.2/**61.1**
Waymo	74.1/76.6/**77.3**	54.1/55.9/**56.2**	66.0/68.1/**68.7**	21.6/23.5/**24.3**	71.3

TABLE III: **UDA from Argoverse to KITTI.** We report AP_{BEV}/ AP_{3D} of the **car** category at IoU = 0.7 and IoU = 0.5 across different depth range on the test set. NR stands for No-Retrain baseline, ST stands for Self-Training [40], SN stands for Statistical Normalization [7]. Our method *Dream* is marked in blue. We show the performance of in-domain model, i.e., the model trained and evaluated on KITTI, at the first row in gray. We also show results by directly applying online and offline (not feasible in real-time) tracking. Best viewed in color.

Method	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	90.0 / 89.9	81.0 / 79.9	40.4 / 36.3	89.0 / 78.0	70.3 / 51.5	26.6 / 9.8
NR	89.4 / 88.8	71.0 / 65.2	18.0 / 13.9	72.6 / 47.8	35.8 / 14.6	4.9 / 3.0
NR + online	89.3 / 88.5	71.9 / 66.1	18.3 / 14.0	72.3 / 47.5	38.2 / 14.9	5.5 / 1.2
NR + offline	89.3 / 88.8	72.7 / 67.7	18.9 / 15.0	74.5 / 49.6	43.5 / 18.1	7.3 / 1.8
ST	**89.5 / 89.2**	73.2 / 68.5	28.2 / 22.5	76.8 / 52.9	44.2 / 20.6	13.1 / 2.1
Dream	89.3 / **89.2**	**74.6 / 72.4**	**30.1 / 25.6**	**77.6 / 54.7**	**49.9 / 24.3**	**14.5 / 3.4**
SN	89.3 / 88.2	69.6 / 65.4	14.6 / 13.3	83.8 / 59.1	53.5 / 27.2	9.3 / 3.6
SN + online	89.4 / 88.2	68.8 / 65.4	19.4 / 16.2	83.5 / 60.1	50.7 / 27.0	13.4 / 9.5
SN + offline	89.4 / 88.2	68.9 / 66.1	21.8 / 19.3	83.3 / 59.2	50.9 / 26.9	13.8 / 9.8
SN + ST	**89.8** / 89.3	74.4 / 72.8	22.3 / 21.3	**87.0** / 70.4	62.2 / 37.7	16.4 / 7.1
SN + Dream	**89.8 / 89.4**	**75.4 / 73.8**	**29.4 / 25.4**	**87.0 / 73.5**	**62.8 / 41.9**	**17.2 / 10.3**

TABLE IV: **UDA from KITTI (city, campus) to KITTI (road, residential)**. Naming is as in Table III.

Method	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
NR	89.4 / 89.4	79.7 / 77.9	36.2 / 30.9	88.3 / 77.9	66.0 / 47.5	19.2 / 6.2
ST	89.4 / 89.4	78.0 / 76.7	35.9 / 31.2	88.2 / 77.8	66.3 / 48.3	19.0 / 7.7
Dream	**89.6 / 89.6**	**80.1 / 78.9**	**41.4 / 35.3**	**88.5 / 78.2**	**68.0 / 49.6**	**23.2 / 12.7**

TABLE V: **Ablation study of UDA from Argoverse to KITTI.** We report AP_{BEV}/ AP_{3D} of the **car** category at IoU = 0.5 and IoU = 0.7 across different depth range, using POINTRCNN model. Naming is as in Table III. S stands for smoothing, R for resizing, I for interpolation and E for extrapolation.

Method	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
NR	89.4 / 88.8	71.0 / 65.2	18.0 / 13.9	72.6 / 47.8	35.8 / 14.6	4.9 / 3.0
ST	89.5 / 89.2	73.2 / 68.5	28.2 / 22.5	76.8 / 52.9	44.2 / 20.6	13.1 / 2.1
ST + S	89.5 / 89.3	73.7 / 68.7	28.3 / 22.3	76.8 / 53.4	45.2 / 21.5	12.9 / **4.7**
ST + S + R	**89.5 / 89.3**	74.0 / 71.6	28.3 / 23.9	**78.0 / 55.2**	**50.8** / 23.9	10.3 / 2.7
ST + S + R + I	89.3 / 89.1	73.9 / 71.6	28.1 / 23.3	77.6 / 54.5	50.4 / 24.0	11.2 / 3.4
ST + S + R + I + E	89.4 / 89.2	**74.9 / 72.5**	**31.0 / 25.7**	77.8 / 55.1	50.4 / **24.1**	**14.3** / 3.3

correct. But just these do not address the second criteria for desired pseudo-labels: *high coverage*. We observe noticeable boosts when interpolation and extrapolations are added, specially for far away objects. This is due to extrapolations and interpolations recovering pseudo-labels for low confidence or missed detections for distant vehicles.

Adaptation results using PIXOR. To show the generality of our approach, we further apply it to another detector PIXOR [3] from Argoverse to KITTI in Table VI. Dreaming improves the accuracy at farther ranges (30-80 m) while maintaining the accuracy at close range (0-30 m). Interestingly,

TABLE VI: **UDA from Argoverse to KITTI using PIXOR.** We report AP_{BEV} of the **car** category at IoU = 0.5 and 0.7. Naming is as in Table III.

Method	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	88.7	62.6	21.4	79.6	49.9	10.0
NR	85.7	57.2	12.9	54.2	23.6	4.7
ST	86.0	56.3	12.2	55.3	24.6	2.5
Dream	**87.1**	**61.1**	**20.2**	**58.0**	**28.1**	**4.8**
SN	86.7	58.7	15.1	76.2	38.7	5.1
SN + ST	**87.4**	58.9	12.4	**78.0**	42.4	3.8
SN + Dream	**87.4**	**64.2**	**22.2**	77.9	**42.5**	**4.5**

at IoU 0.5 in the 30-80 m ranges, we are able to surpass the in-domain performance, which uses models trained only in the target domain with the ground-truth labels. This results showcases the power of unsupervised domain adaptation (UDA): with a suitably designed algorithm, UDA that leverages both the source and target domain could outperform models trained only in a single domain.

Others. We show more results and qualitative visualizations in the supplementary material.

V. Conclusion and Discussion

In this paper, we have introduced a novel method towards closing the gap between source and target in unsupervised domain adaptation for LiDAR-based 3D object detection. Our approach is based on self-training, while leveraging vehicle dynamics and offline analysis to generate pseudo-labels. Importantly, we can generate high quality pseudo-labels even for difficult cases (i.e. far-away objects), which the detector tends to miss before adaptation. Fine-tuning on these pseudo-labels improves detection performance drastically in the target domain. It is hard to conceive an autonomous vehicle manufacturer that could collect, label, and update data for every consumer environment, meeting the requirements to allow self-driving cars to operate everywhere freely and safely. By significantly reducing the adaptation gap between domains, our approach takes a significant step towards making this vision a reality nevertheless.

Acknowledgment

This research is supported by grants from the National Science Foundation NSF (III-1618134, III-1526012, IIS-1149882, IIS-1724282, TRIPODS-1740822, IIS-2107077, OAC-2118240, OAC-2112606), the Office of Naval Research DOD (N00014-17-1-2175), the Bill and Melinda Gates Foundation, and the Cornell Center for Materials Research with funding from the NSF MRSEC program (DMR-1719875).

REFERENCES

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

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

[3] B. Yang, W. Luo, and R. Urtasun, "Pixor: Real-time 3d object detection from point clouds," in *CVPR*, 2018.

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

[5] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *CVPR*, 2012.

[6] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[7] Y. Wang, C. Xiangyu, Y. Yurong, E. L. Li, B. Hariharan, M. Campbell, K. Q. Weinberger, and C. Wei-Lun, "Train in germany, test in the usa: Making 3d object detectors generalize," in *CVPR*, 2020.

[8] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *CVPR*, 2012.

[9] M.-F. Chang, J. W. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," in *CVPR*, 2019.

[10] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet, "Lyft level 5 av dataset 2019," urlhttps://level5.lyft.com/dataset/, 2019.

[11] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," 2019.

[12] 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," *preprint, arXiv:1903.11027*, 2019.

[13] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," in *CVPR*, 2019.

[14] Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving," in *ICLR*, 2020.

[15] R. Qian, D. Garg, Y. Wang, Y. You, S. Belongie, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao, "End-to-end pseudo-lidar for image-based 3d object detection," in *CVPR*, 2020.

[16] P. Li, X. Chen, and S. Shen, "Stereo r-cnn based 3d object detection for autonomous driving," in *CVPR*, 2019.

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

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

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

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

[21] Z. He and L. Zhang, "Multi-adversarial faster-rcnn for unrestricted object detection," in *ICCV*, 2019.

[22] T. Kim, M. Jeong, S. Kim, S. Choi, and C. Kim, "Diversify and match: A domain adaptive representation learning paradigm for object detection," in *CVPR*, 2019.

[23] K. Saito, Y. Ushiku, T. Harada, and K. Saenko, "Strong-weak distribution alignment for adaptive object detection," in *CVPR*, 2019.

[24] X. Zhu, J. Pang, C. Yang, J. Shi, and D. Lin, "Adapting object detectors via selective cross-domain alignment," in *CVPR*, 2019.

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

[26] F. S. Saleh, M. S. Aliakbarian, M. Salzmann, L. Petersson, and J. M. Alvarez, "Effective use of synthetic data for urban scene semantic segmentation," in *ECCV*. Springer, 2018.

[27] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker, "Learning to adapt structured output space for semantic segmentation," in *CVPR*, 2018.

[28] Y. Zhang, P. David, H. Foroosh, and B. Gong, "A curriculum domain adaptation approach to the semantic segmentation of urban scenes," *TPAMI*, vol. 42, no. 8, pp. 1823–1841, 2019.

[29] Y. Zou, Z. Yu, B. Vijaya Kumar, and J. Wang, "Unsupervised domain adaptation for semantic segmentation via class-balanced self-training," in *ECCV*, 2018.

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

[31] A. RoyChowdhury, P. Chakrabarty, A. Singh, S. Jin, H. Jiang, L. Cao, and E. Learned-Miller, "Automatic adaptation of object detectors to new domains using self-training," in *CVPR*, 2019.

[32] Q. Tao, H. Yang, and J. Cai, "Zero-annotation object detection with web knowledge transfer," in *ECCV*, 2018.

[33] J. Liang, R. He, Z. Sun, and T. Tan, "Distant supervised centroid shift: A simple and efficient approach to visual domain adaptation," in *CVPR*, 2019.

[34] W. Zhang, W. Ouyang, W. Li, and D. Xu, "Collaborative and adversarial network for unsupervised domain adaptation," in *CVPR*, 2018.

[35] S. Kim, J. Choi, T. Kim, and C. Kim, "Self-training and adversarial background regularization for unsupervised domain adaptive one-stage object detection," in *ICCV*, 2019.

[36] G. French, M. Mackiewicz, and M. Fisher, "Self-ensembling for visual domain adaptation," in *ICLR*, 2018.

[37] N. Inoue, R. Furuta, T. Yamasaki, and K. Aizawa, "Cross-domain weakly-supervised object detection through progressive domain adaptation," in *CVPR*, 2018.

[38] M. Khodabandeh, A. Vahdat, M. Ranjbar, and W. G. Macready, "A robust learning approach to domain adaptive object detection," in *ICCV*, 2019.

[39] D. McClosky, E. Charniak, and M. Johnson, "Effective self-training for parsing," in *ACL*, 2006.

[40] A. Kumar, T. Ma, and P. Liang, "Understanding self-training for gradual domain adaptation," *preprint, arXiv:2002.11361*, 2020.

[41] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, 2013.

[42] R. G. Cinbis, J. Verbeek, and C. Schmid, "Weakly supervised object localization with multi-fold multiple instance learning," *TPAMI*, vol. 39, no. 1, pp. 189–203, 2016.

[43] C. Qin, H. You, L. Wang, C.-C. J. Kuo, and Y. Fu, "Pointdan: A multi-scale 3d domain adaption network for point cloud representation," in *NeurIPS*, 2019.

[44] K. Saleh, A. Abobakr, M. Attia, J. Iskander, D. Nahavandi, M. Hossny, and S. Nahvandi, "Domain adaptation for vehicle detection from bird's eye view lidar point cloud data," in *ICCVW*, 2019.

[45] Z. Wang, S. Ding, Y. Li, M. Zhao, S. Roychowdhury, A. Wallin, G. Sapiro, and Q. Qiu, "Range adaptation for 3d object detection in lidar," in *ICCVW*, 2019.

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

[47] 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.

[48] X. Liang, S. Liu, Y. Wei, L. Liu, L. Lin, and S. Yan, "Towards computational baby learning: A weakly-supervised approach for object detection," in *ICCV*, 2015.

[49] A. Ošep, P. Voigtlaender, J. Luiten, S. Breuers, and B. Leibe, "Large-scale object mining for object discovery from unlabeled video," in *ICRA*, 2019.

[50] I. Misra, A. Shrivastava, and M. Hebert, "Watch and learn: Semi-supervised learning for object detectors from video," in *CVPR*, 2015.

[51] K. Kumar Singh, F. Xiao, and Y. Jae Lee, "Track and transfer: Watching videos to simulate strong human supervision for weakly-supervised object detection," in *CVPR*, 2016.

[52] K. Tang, V. Ramanathan, L. Fei-Fei, and D. Koller, "Shifting weights: Adapting object detectors from image to video," in *NeurIPS*, 2012.

[53] A. Ošep, P. Voigtlaender, J. Luiten, S. Breuers, and B. Leibe, "Large-scale object discovery and detector adaptation from unlabeled video," *preprint, arXiv:1712.08832*, 2017.

[54] C. R. Qi, Y. Zhou, M. Najibi, P. Sun, K. Vo, B. Deng, and D. Anguelov, "Offboard 3d object detection from point cloud sequences," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6134–6144.

[55] B. Yang, M. Bai, M. Liang, W. Zeng, and R. Urtasun, "Auto4d: Learning to label 4d objects from sequential point clouds," *arXiv preprint arXiv:2101.06586*, 2021.
[56] X. Weng, J. Wang, D. Held, and K. Kitani, "3d multi-object tracking: A baseline and new evaluation metrics," *preprint, arXiv:1907.03961*, 2020.
[57] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *ICCV*, 2019, pp. 12 697–12 705.
[58] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Online multiperson tracking-by-detection from a single, uncalibrated camera," *TPAMI*, vol. 33, no. 9, pp. 1820–1833, 2010.
[59] ——, "Robust tracking-by-detection using a detector confidence particle filter," in *ICCV*. IEEE, 2009, pp. 1515–1522.
[60] Y. Hua, K. Alahari, and C. Schmid, "Online object tracking with proposal selection," in *ICCV*, 2015, pp. 3092–3100.
[61] C. Diaz-Ruiz, Y. Wang, W. Chao, K. Weinberger, and M. Campbell, "Vision-only 3d tracking for self-driving cars," in *CASE*, 2019, pp. 1029–1034.
[62] H.-k. Chiu, A. Prioletti, J. Li, and J. Bohg, "Probabilistic 3d multi-object tracking for autonomous driving," *preprint, arXiv:2001.05673*, 2020.
[63] T. Yin, X. Zhou, and P. Krähenbühl, "Center-based 3d object detection and tracking," *preprint, arXiv:2006.11275*, 2020.
[64] I. Miller, M. Campbell, and D. Huttenlocher, "Efficient unbiased tracking of multiple dynamic obstacles under large viewpoint changes," *IEEE Transactions on Robotics*, vol. 27, no. 1, pp. 29–46, 2011.
[65] M. Chen, K. Q. Weinberger, and J. Blitzer, "Co-training for domain adaptation," in *NeurIPS*, 2011.
[66] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2001.
[67] J. Yin, J. Shen, C. Guan, D. Zhou, and R. Yang, "Lidar-based online 3d video object detection with graph-based message passing and spatiotemporal transformer attention," in *CVPR*, 2020.
[68] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *NeurIPS*, 2017.
[69] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun, "3d object proposals for accurate object class detection," in *NeurIPS*, 2015.

APPENDIX

A. Hyper-parameters

In performing the forward pass of the Extended Kalman Filter (EKF) to calculate the a-posteriori state and state error covariance estimates (i.e., $\bar{\mathbf{x}}_{k|k}$ $P_{k|k}$) and the a-priori state and state error covariance (i.e., $\bar{\mathbf{x}}_{k+1|k}$ $P_{k+1|k}$) in subsection III-C, a process noise covariance matrix (Q), measurement noise covariance matrix (R), initial state estimate ($\bar{\mathbf{x}}_0$), and initial state error covariance matrix (P_0) are necessary. The measurement noise matrix is established through measurement variance, and was obtained by observing the variance of position (x, y), orientation (θ), length (l) and width (w) errors from testing detector in Argoverse to KITTI setting. A larger measurement noise matrix is used in the EKF for extrapolation as larger variance was observed with far-away detections. The process noise matrix should represent the magnitudes of dynamical noise that the system might experience. In the model in [61], which is constant velocity and heading, there are seven noise parameters: $e_\theta, e_s, e_{v_x}, e_{v_y}, e_{\omega_z}, e_l, e_w$ which correspond to the diagonal of the Q matrix. The variables e_θ, e_s, e_l, e_w are modeled as zero mean, mutually uncorrelated, Gaussian, and white process noise. Intuitively, e_θ, e_s represent the uncertainty associated with the orientation and speed of vehicles, especially given that the model assumes constant speed straight line motion. The noise associated with the orientation and speed of the target vehicles are the largest and of most importance. The time derivative of the objects length and width are zero, where e_l, e_w are small tuning parameters which control the response of the filter. While e_{v_x}, e_{v_y}, e_{ω_z} are noises associated with the pose of ego-vehicle and small values were chosen given that a high accuracy in ego-vehicle pose is expected across the datasets. Finally, the EKF necessitates an initialization for the state and state error covariance estimates. The initial detection was used for state initialization, and relatively large conservative uncertainties were used for state error covariance.

- Extended Kalman Filter (EKF) and Global Nearest Neighbor (GNN) parameters for tracking and data association (cf. subsection III-C):
  1) Measurement noise covariance matrix: $R = \text{diag}(0.1\text{m}^2, 0.1\text{m}^2, 0.015\text{rad}^2, 0.07\text{m}^2, 0.04\text{m}^2)$
  2) Process noise covariance matrix: $Q = \text{diag}(0.1218\frac{\text{rad}}{\text{s}}^2, 1\frac{\text{m}}{\text{s}^2}^2, 0.00545\frac{\text{rad}}{\text{s}}^2, 0.00545\frac{\text{m}}{\text{s}}^2, 0.00307\frac{\text{rad}}{\text{s}}^2, 0.01\frac{\text{m}}{\text{s}}^2, 0.01\frac{\text{m}}{\text{s}}^2)$
  3) State error covariance matrix initialization: $P_0 = \text{diag}(2\text{m}^2, 2\text{m}^2, 0.1\text{rad}^2, 5\frac{\text{m}}{\text{s}}^2, 0.5\text{m}^2, 0.32\text{m}^2)$
  4) The initial state estimate, $\bar{\mathbf{x}}_0$, is set to be the first detection values.
  5) The data association threshold (in **BEV IoU**) in GNN is set to be 0.3.
  6) The fraction of distance between the vehicle center and back-axle is $\frac{1}{4}l$, as in [61].
- EKF parameters for extrapolation:
  1) Measurement noise covariance matrix: $R = \text{diag}(0.5\text{m}^2, 0.5\text{m}^2, 0.06\text{rad}^2, 0.07\text{m}^2, 0.04\text{m}^2)$

We set $c_{\text{min-hits}} = 3$ and $c_{\text{max-age}} = 3$. When doing extrapolation, we use -25 and -3 as thresholds for POINTR-CNN and PIXOR models respectively.

B. Details on Datasets

We split each dataset into two parts, a training set and a test set. When a dataset is used as the source, we train the detector using its (ground truth) training labeled frames. When a dataset is used as the target, we adapt the detector using its training sequences without revealing the ground truth labels. We evaluate the adapted models on the test set. We provide detailed properties of the five autonomous driving datasets and the way we split the data as follows.

KITTI. The KITTI object detection benchmark [6], [5] contains 7,481 scenes for training and 7,518 scenes for testing. All the scenes are pictured around Karlsruhe, Germany in clear weather and day time. For each scene, KITTI provides 64-beam Velodyne LiDAR point cloud and stereo images. The training set is further separated into 3,712 training and 3,769 validation scenes as suggested by [69]. The training scenes are sampled from 96 data sequences, which have no overlap with the sequences where validation scenes are sampled. These training sequences are collected in 10 Hz, resulting in 13,596 frames. We extract the sequences from the raw KITTI data as our adaptation data. We use such data splits in all experiments related to KITTI, except for **adaptation between different**

locations inside the same dataset (cf. Table IV of the main paper). For adaptation between different locations inside the KITTI dataset, we split the sequences in training set by their categories: *city/campus* as the source (38 sequences, 9,556 frames, 3,420 labeled frames) and *residential/road* as the target (23 sequences, 10,448 frames, 3619 labeled frames). In Table IV, all models are pre-trained in city/campus data. Due to limited data, different from what we do UDA among five autonomous driving datasets, *we perform adaptation and final evaluation on the full target data.* Note that in this case, the detectors still do not have access to ground truth labels during UDA.

Argoverse. The Argoverse dataset [9] is collected around Miami and Pittsburgh, USA in multiple weathers and during different times of the day. For each scene (timestamp), Argoverse provides a 64-beam LiDAR point cloud captured by stacking two 32-beam Velodyne LiDAR vertically. We extracted synchronized images (from front camera) and corresponding point clouds from the original Argoverse dataset. We follow the official split on training and validation sets, which contain 13,122 scenes and 5,014 scenes respectively. We use sequences in the training set (without using the ground truth labels) as our adaptation data.

Lyft. The Lyft Level 5 dataset collects 18,634 scenes around Palo Auto, USA in clear weather and during day time. For each scene, Lyft provides the ground-truth bounding box labels and point cloud captured by a 40 (or 64)-beam roof LiDAR and two 40-beam bumper LiDAR sensors. We follow [7] and separate the dataset by sequences, resulting in 12,599 frames for training (100 sequences), 3,024 validation frames (24 sequences) and 3,011 frames for testing (24 sequences). The sequences in 5 Hz and we use training sequences without labels as adaptation data.

nuScenes. The nuScenes dataset [12] collects scenes around Boston, USA and Singapore in multiple weather conditions and during different times of the day. For each scene, nuScenes provides a point cloud captured by a 32-beam roof LiDAR. We use the data collected in Boston, and sampled 312 sequences for training and 78 sequences for validation. We use 10 Hz sensor data without labels as our adaptation data (61,121 frames in total), and evaluate the model on 2 Hz labeled data in validation set (3,133 frames). Note that after generating pseudo-labels, we sub-sample adaptation data using 2 Hz into 12,562 frames.

Waymo. The Waymo open dataset [11] is mostly collected around San Francisco, Phoenix, and Mountain View in multiple weather conditions and at multiple times of the day. It provides point clouds captured in 10 Hz by five LiDAR sensors (one on roof, four on side) and images from five cameras. We randomly sample 60 tracks (11,886 frames) captured in San Francisco as our adaptation data, and another 100 tracks (19,828 frames) in San Francisco as our validation data.

C. Qualitative Results

In Figure 2, Figure 3 and Figure 4, we compare qualitatively the detection results from models trained with different adaptation strategies. We select (Argoverse, KITTI), (nuScenes, KITTI) and (nuScenes, Argoverse) as (*source*, *target*) example pairs in qualitative visualization. It can be seen that models without retraining tend to miss faraway objects. Models with self-training are able to detect some of these objects. Models with dreaming can detect more faraway objects. Self-training and dreaming both exhibit some more false positive detection.

D. Adaptation among Five Datasets

In Table VII, we present UDA results on all possible (source, target) pairs among the five autonomous driving datasets (20 pairs in total). On each pair, we show results with and without statistical normalization [7]. As in Table III, we report AP_{BEV}/ AP_{3D} of the **car** category at IoU $= 0.7$ and IoU $= 0.5$ across different depth range, using the PointRCNN detector. It can be seen that under these 20 UDA scenarios, our method consistently improves the adaptation performance on faraway ranges, while having mostly equal or better performance on close-by ranges.

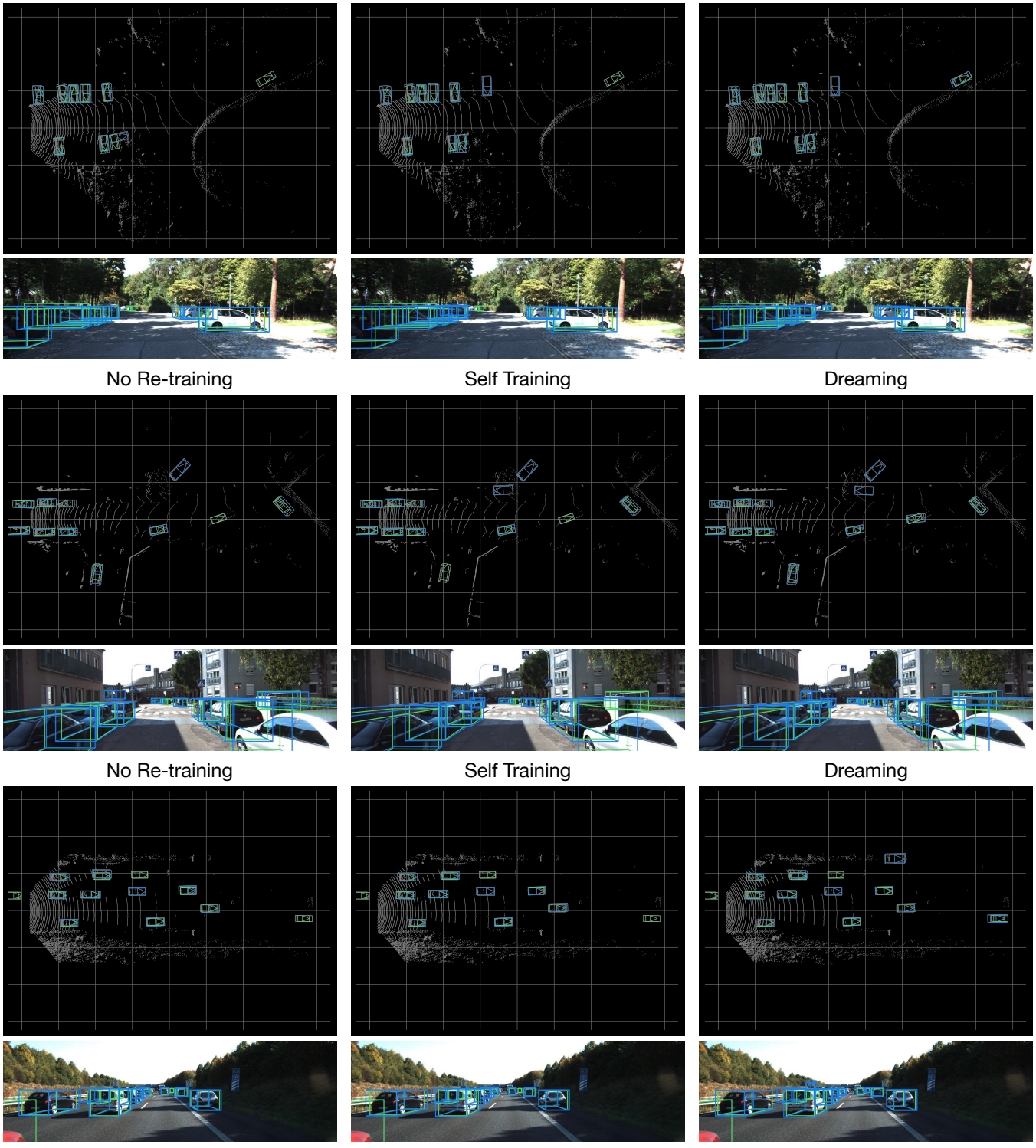

Fig. 2: **Qualitative Results.** We compare the detection results on several scenes from the KITTI validation set by the POINTRCNN detectors that are trained on 1) Argoverse dataset (No Re-training), 2) Argoverse dataset and fine-tuned using self-training on KITTI (Self Training), and 3) Argoverse dataset and fine-tuned using dreaming on KITTI (Dreaming). We visualize them from both frontal-view images and bird's-eye view point maps. Ground-truth boxes are in green and detected bounding boxes are in blue. The ego vehicle is on the left side of the BEV map and looking to the right. One floor square is 10m$\times$10m. Best viewed in color. Zoom in for details.

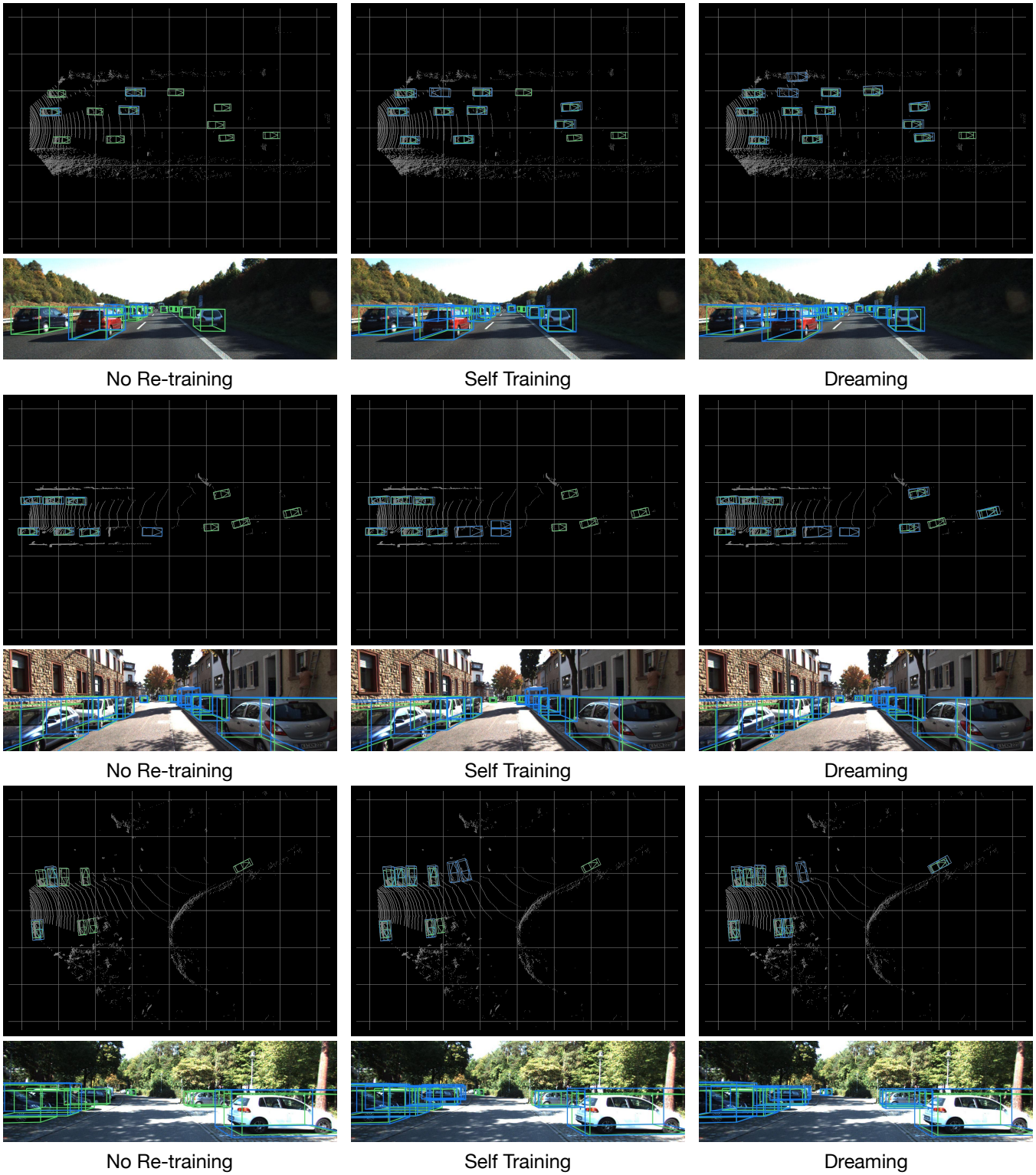

Fig. 3: **Qualitative Results.** The setups are the same as those in Figure 2, but the models are pre-trained in nuScenes dataset and tested on KITTI dataset.

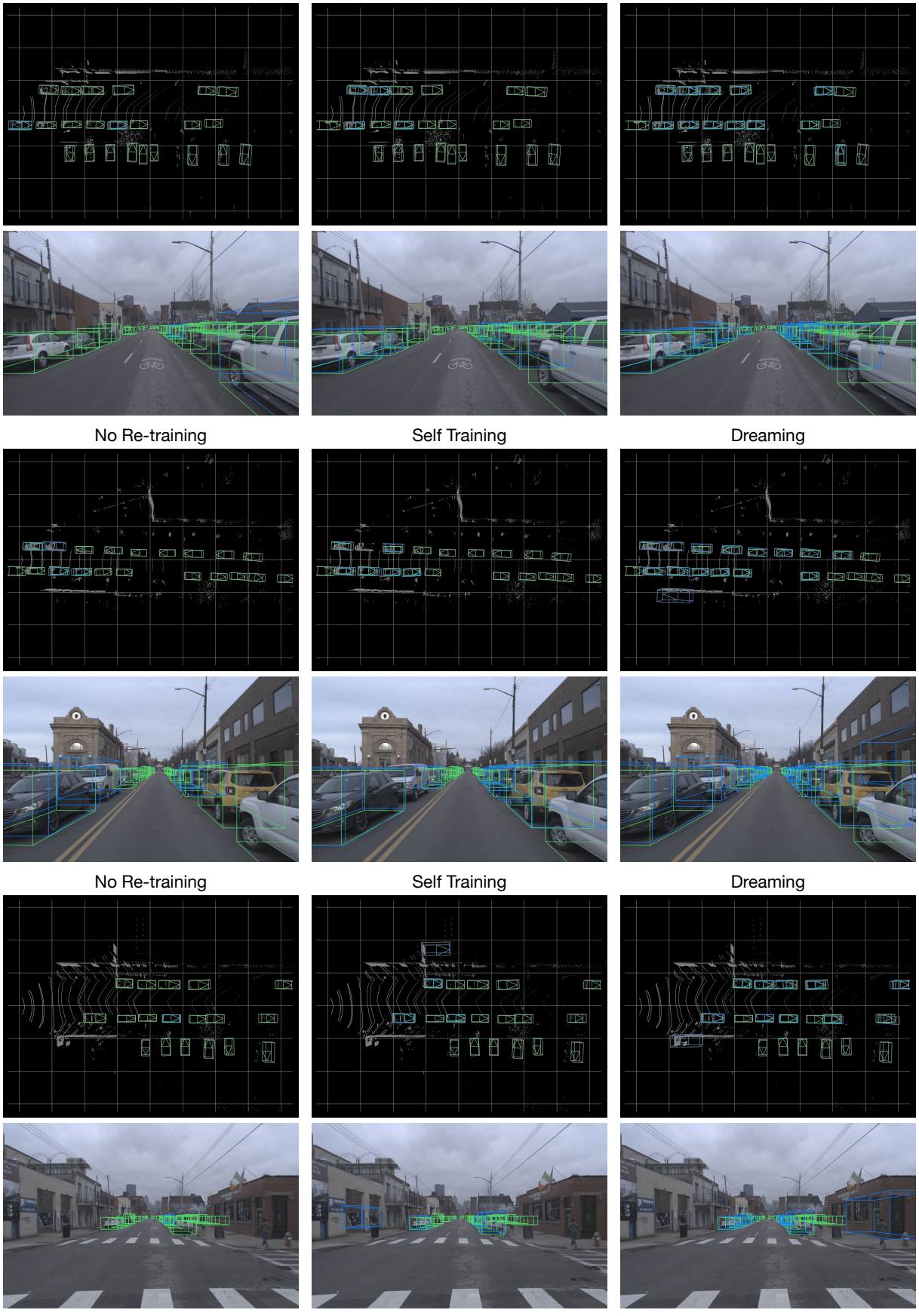

Fig. 4: **Qualitative Results.** The setups are the same as those in Figure 2, but the models are pre-trained in nuScenes dataset and tested on Argoverse dataset.

TABLE VII: **Unsupervised domain adaptation among five autonomous driving datasets.** Naming is as that in Table III of the main paper.

(a) KITTI to Argoverse

Method \ Range(m)	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	88.6 / 85.7	68.5 / 66.4	37.8 / 30.0	76.5 / 53.2	56.6 / 30.4	20.2 / 10.1
no retraining	79.4 / 76.5	51.9 / 44.6	20.1 / 15.4	53.1 / 32.9	29.3 / 12.3	5.9 / 3.0
ST	85.2 / 77.6	56.2 / 52.2	26.9 / 19.5	61.3 / **38.8**	34.9 / 17.4	10.9 / **9.1**
Dreaming	**85.6 / 77.8**	**61.3 / 54.1**	**28.4 / 20.8**	**63.3** / 38.6	**41.6 / 20.2**	**12.4** / 4.5
SN only	79.3 / 77.6	54.7 / 52.0	27.5 / 21.2	66.2 / 48.5	43.8 / 21.2	16.7 / 9.1
SN + ST	84.9 / 78.0	56.7 / 54.7	29.0 / 22.5	73.0 / **50.8**	46.4 / **22.3**	17.6 / 9.1
SN + Dreaming	**85.1 / 78.2**	**62.3 / 55.7**	**33.6 / 26.5**	**73.1** / 50.3	**50.6** / 20.6	**18.5 / 9.9**

(b) KITTI to Lyft

Method \ Range(m)	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	89.2 / 88.9	78.4 / 77.9	67.2 / 65.5	88.4 / 79.0	74.8 / 54.6	54.7 / 27.3
no retraining	81.0 / **80.8**	73.6 / 66.8	49.6 / 40.0	69.7 / 50.4	48.0 / 24.5	25.4 / 5.7
ST	**85.9** / 80.7	74.1 / 67.9	56.3 / 46.3	75.2 / 53.7	53.8 / 25.3	28.9 / 7.6
Dreaming	**85.9** / 80.7	**74.8 / 68.6**	**56.4 / 47.7**	**76.6 / 54.2**	**56.3 / 31.3**	**34.4 / 10.2**
SN only	**81.0 / 80.9**	72.6 / 67.2	55.0 / 47.5	**78.7 / 67.3**	64.9 / 45.1	43.4 / 18.0
SN + ST	80.6 / 80.5	73.5 / 72.5	56.0 / 48.8	78.1 / 65.9	66.6 / 45.5	45.9 / 18.7
SN + Dreaming	80.5 / 80.3	**74.3 / 72.9**	**56.6 / 49.7**	78.1 / 65.9	**67.0 / 48.5**	**46.7 / 22.0**

(c) KITTI to nuScenes

Method \ Range(m)	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	65.6 / 64.5	18.4 / 17.2	3.5 / 2.8	57.8 / 37.0	15.8 / 5.3	2.6 / 0.8
no retraining	48.6 / 40.7	11.0 / 4.5	1.4 / **0.9**	33.3 / 13.4	4.5 / 0.7	0.3 / **0.0**
ST	52.2 / 43.9	11.8 / 9.1	**9.1** / 0.4	33.2 / 8.6	9.1 / **4.5**	3.0 / **0.0**
Dreaming	**53.6 / 45.8**	**13.8 / 9.7**	4.5 / 0.4	**39.1 / 14.2**	**9.8** / 3.0	**4.5 / 0.0**
SN only	52.9 / 47.0	11.1 / 10.0	1.0 / 0.4	44.7 / 22.0	10.2 / 4.5	0.6 / **0.1**
SN + ST	51.9 / 47.3	11.5 / 10.1	1.4 / 0.6	45.6 / **26.3**	10.6 / **9.1**	0.9 / **0.1**
SN + Dreaming	**53.4 / 51.4**	**13.7 / 10.5**	**10.1 / 9.1**	**50.0** / 24.3	**11.3 / 9.1**	**9.1 / 0.1**

(d) KITTI to Waymo

Method \ Range(m)	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	81.6 / 81.5	71.7 / 71.2	58.2 / 50.8	80.8 / 68.8	69.1 / 54.9	47.0 / 27.2
no retraining	80.5 / 71.5	69.0 / 60.3	42.8 / 33.4	43.5 / 16.6	34.6 / 11.0	20.6 / **10.8**
ST	81.0 / 78.4	69.5 / 61.4	48.5 / 39.3	48.2 / **18.6**	36.8 / 16.0	23.6 / 7.0
Dreaming	**81.1 / 78.5**	**69.9 / 61.8**	**50.2 / 41.0**	**51.4** / 13.8	**44.5 / 16.7**	**25.6** / 7.8
SN only	81.0 / 80.4	70.0 / 62.4	42.7 / 40.7	71.4 / **53.4**	60.9 / 39.8	38.0 / 19.4
SN + ST	81.1 / 80.6	70.5 / **68.5**	49.7 / 42.5	**78.1** / 53.1	61.7 / 45.1	40.2 / 20.6
SN + Dreaming	**81.2 / 80.8**	**70.9** / 68.3	**51.1 / 48.3**	**78.1** / 51.6	**67.4 / 45.5**	**41.1 / 20.8**

(e) Argoverse to KITTI

Method \ Range(m)	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	90.0 / 89.9	81.0 / 79.9	40.4 / 36.3	89.0 / 78.0	70.3 / 51.5	26.6 / 9.8
no retraining	89.4 / 88.8	71.0 / 65.2	18.0 / 13.9	72.6 / 47.8	35.8 / 14.6	4.9 / 3.0
ST	**89.5** / **89.2**	73.2 / 68.5	28.2 / 22.5	76.8 / 52.9	44.2 / 20.6	13.1 / 2.1
Dreaming	89.3 / **89.2**	**74.6** / **72.4**	**30.1** / **25.6**	**77.6** / **54.7**	**49.9** / **24.3**	**14.5** / **3.4**
SN only	89.3 / 88.2	69.6 / 65.4	14.6 / 13.3	83.8 / 59.1	53.5 / 27.2	9.3 / 3.6
SN + ST	**89.8** / 89.3	74.4 / 72.8	22.3 / 21.3	**87.0** / 70.4	62.2 / 37.7	16.4 / 7.1
SN + Dreaming	**89.8** / **89.4**	**75.4** / **73.8**	**29.4** / **25.4**	**87.0** / **73.5**	**62.8** / **41.9**	**17.2** / **10.3**

(f) Argoverse to Lyft

Method \ Range(m)	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	89.2 / 88.9	78.4 / 77.9	67.2 / 65.5	88.4 / 79.0	74.8 / 54.6	54.7 / 27.3
no retraining	79.9 / 79.8	68.5 / 67.4	46.3 / 42.9	77.2 / 54.8	57.6 / 32.0	31.8 / 12.4
ST	80.2 / **80.1**	72.7 / 67.9	48.8 / 46.0	78.3 / 55.4	63.3 / 35.8	35.0 / 11.3
Dreaming	**80.3** / **80.1**	**73.8** / **72.4**	**54.5** / **47.6**	**78.9** / **57.3**	**64.2** / **37.1**	**35.6** / **12.8**
SN only	79.6 / 79.2	67.1 / 66.0	44.8 / 41.8	75.7 / 54.7	55.3 / 30.4	32.4 / **14.2**
SN + ST	80.1 / 79.9	72.2 / 67.4	47.4 / 45.3	78.3 / 56.0	63.6 / 38.0	36.4 / 13.5
SN + Dreaming	**80.2** / **80.0**	**73.5** / **68.2**	**54.0** / **47.5**	**78.5** / **56.8**	**64.1** / **39.3**	**41.9** / 14.1

(g) Argoverse to nuScenes

Method \ Range(m)	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	65.6 / 64.5	18.4 / 17.2	3.5 / 2.8	57.8 / 37.0	15.8 / 5.3	2.6 / 0.8
no retraining	**51.6** / **45.3**	**10.3** / **9.1**	0.6 / 0.1	**43.3** / 17.9	**9.1** / 0.3	0.2 / **0.1**
ST	47.7 / 43.4	6.3 / 5.4	**9.1** / 0.2	41.5 / **18.4**	5.6 / **4.5**	0.3 / **0.1**
Dreaming	48.1 / 43.5	8.0 / 4.5	3.0 / **3.0**	41.4 / **18.4**	5.3 / 2.3	**3.0** / 0.0
SN only	47.6 / 45.6	5.7 / 4.5	0.5 / 0.2	43.1 / 18.2	4.5 / 0.7	0.1 / 0.0
SN + ST	49.1 / 44.5	11.0 / 4.2	**9.1** / **9.1**	42.7 / 22.4	10.4 / 1.8	**9.1** / **9.1**
SN + Dreaming	**50.7** / **45.9**	**12.8** / **10.1**	**9.1** / **9.1**	**44.1** / **24.7**	**10.7** / **9.1**	**9.1** / 4.5

(h) Argoverse to Waymo

Method \ Range(m)	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	81.6 / 81.5	71.7 / 71.2	58.2 / 50.8	80.8 / 68.8	69.1 / 54.9	47.0 / 27.2
no retraining	81.2 / 80.4	69.3 / 61.7	50.4 / 47.8	70.8 / 43.7	59.6 / 35.0	39.1 / 18.8
ST	**81.3** / **80.7**	69.9 / **67.1**	50.9 / 48.2	**78.0** / **50.0**	**60.8** / 36.0	40.7 / **20.2**
Dreaming	**81.3** / 80.6	**70.0** / 67.0	**56.0** / **49.1**	77.3 / 43.4	60.5 / **36.2**	**40.9** / **20.2**
SN only	81.3 / 80.7	69.3 / 61.6	49.7 / 47.4	71.0 / 51.8	59.1 / 34.3	38.3 / 15.0
SN + ST	**81.4** / **81.0**	70.1 / **67.7**	50.7 / 48.0	**78.4** / **55.5**	**61.1** / 41.1	40.4 / **20.0**
SN + Dreaming	**81.4** / **81.0**	**70.4** / 67.5	**56.1** / **49.5**	77.9 / 53.6	**61.1** / **41.8**	**44.9** / **20.0**

(i) Lyft to KITTI

Range(m) / Method	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	90.0 / 89.9	81.0 / 79.9	40.4 / 36.3	89.0 / 78.0	70.3 / 51.5	26.6 / 9.8
no retraining	88.9 / 88.6	67.7 / 64.9	26.1 / 20.4	65.2 / 38.7	36.1 / 12.9	8.5 / 2.2
ST	89.9 / 89.6	73.5 / 68.8	30.8 / 24.0	71.7 / 40.8	42.2 / **20.5**	10.8 / 2.9
Dreaming	**90.0 / 89.7**	**74.7 / 72.8**	**33.9 / 26.4**	**74.2 / 42.8**	**46.2** / 18.7	**12.1 / 3.6**
SN only	89.5 / 89.5	67.4 / 66.5	28.7 / 24.9	88.0 / 76.0	57.2 / 33.8	20.1 / 6.4
SN + ST	90.0 / 90.0	73.2 / 72.3	31.9 / 28.4	88.4 / 77.2	63.1 / 42.3	22.1 / 10.0
SN + Dreaming	**90.1 / 90.1**	**76.2 / 74.8**	**36.1 / 32.1**	**88.6 / 78.0**	**64.6 / 42.5**	**23.6 / 10.7**

(j) Lyft to Argoverse

Range(m) / Method	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	88.6 / 85.7	68.5 / 66.4	37.8 / 30.0	76.5 / 53.2	56.6 / 30.4	20.2 / 10.1
no retraining	86.4 / 78.6	58.2 / 55.3	29.3 / 25.3	72.5 / 40.1	44.6 / 17.3	17.0 / **9.1**
ST	86.7 / **84.2**	63.6 / 57.0	30.7 / 26.7	74.8 / **42.3**	**50.4 / 19.2**	18.1 / 4.5
Dreaming	**87.3** / 84.1	**64.9 / 61.3**	**35.2 / 27.3**	**75.1** / 41.9	49.7 / 18.7	**18.6 / 9.1**
SN only	86.5 / 78.4	58.1 / 55.0	29.2 / 22.2	72.3 / 46.5	44.3 / 13.0	17.7 / 9.1
SN + ST	86.4 / 78.6	63.8 / 56.9	30.6 / 26.9	74.8 / **48.6**	**50.8 / 21.4**	18.9 / **10.1**
SN + Dreaming	**87.4 / 84.1**	**64.8 / 61.5**	**34.9 / 27.9**	**74.9** / 45.6	50.4 / 20.6	**19.0** / 9.9

(k) Lyft to nuScenes

Range(m) / Method	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	65.6 / 64.5	18.4 / 17.2	3.5 / 2.8	57.8 / 37.0	15.8 / 5.3	2.6 / 0.8
no retraining	**53.2** / 47.1	9.4 / 3.8	4.5 / 0.4	45.9 / 24.0	7.3 / 0.9	4.5 / **0.3**
ST	51.6 / 46.2	13.0 / **10.4**	**9.1 / 9.1**	45.2 / 25.8	11.6 / **9.1**	**9.1** / 0.1
Dreaming	52.8 / **50.4**	**14.9** / 10.2	6.0 / 1.1	**49.6 / 26.7**	**11.8 / 9.1**	4.5 / 0.1
SN only	**53.5** / 47.0	9.0 / 3.6	3.0 / 0.4	45.7 / 24.1	6.4 / 2.3	1.0 / 0.0
SN + ST	52.0 / 46.6	12.9 / **10.5**	4.5 / **4.5**	45.6 / **25.4**	11.4 / **9.1**	**4.5 / 0.1**
SN + Dreaming	52.3 / **49.8**	**15.1** / 10.2	**5.0** / 1.5	**48.8** / 23.3	**12.1 / 9.1**	3.0 / 0.0

(l) Lyft to Waymo

Range(m) / Method	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	81.6 / 81.5	71.7 / 71.2	58.2 / 50.8	80.8 / 68.8	69.1 / 54.9	47.0 / 27.2
no retraining	**81.5** / 81.2	70.9 / 69.8	51.6 / 49.6	**79.5** / 55.8	61.6 / 44.5	40.5 / 18.6
ST	**81.5 / 81.3**	71.1 / 69.8	51.9 / 50.6	79.3 / **56.1**	**67.5 / 46.0**	**46.3 / 23.8**
Dreaming	**81.5** / 81.2	**71.2 / 70.2**	**56.9 / 50.8**	79.1 / 55.7	67.2 / 45.5	46.0 / 21.2
SN only	**81.4 / 81.3**	71.0 / 70.1	51.1 / 49.5	79.8 / **65.5**	61.1 / 46.1	39.0 / 21.3
SN + ST	**81.4 / 81.3**	71.2 / 70.2	51.9 / 50.5	**80.3** / 65.0	**67.5 / 47.5**	**45.5 / 25.1**
SN + Dreaming	**81.4 / 81.3**	**71.3 / 70.4**	**56.9 / 50.9**	80.1 / 64.9	67.1 / **47.5**	44.6 / 22.8

(m) nuScenes to KITTI

Method \ Range(m)	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	90.0 / 89.9	81.0 / 79.9	40.4 / 36.3	89.0 / 78.0	70.3 / 51.5	26.6 / 9.8
no retraining	68.5 / 57.2	46.1 / 33.5	9.6 / 5.5	35.7 / 5.3	12.7 / 1.4	2.8 / **1.8**
ST	**87.8** / **78.9**	64.8 / 51.0	16.4 / 8.7	45.9 / 10.9	21.4 / 2.5	3.4 / 0.6
Dreaming	87.6 / 78.8	**69.3** / **56.3**	**21.7** / **13.4**	**46.4** / **14.5**	**25.7** / **10.2**	**5.4** / 0.5
SN only	78.8 / 78.3	48.9 / 46.6	6.0 / 5.5	74.9 / 40.9	37.7 / 11.4	4.7 / 1.1
SN + ST	**88.7** / **88.5**	65.6 / 63.5	16.6 / 13.0	**84.3** / **61.2**	53.3 / 22.3	**10.6** / **1.8**
SN + Dreaming	88.5 / 88.2	**69.9** / **65.8**	**23.9** / **20.9**	84.0 / 60.8	**53.9** / **24.5**	8.9 / 1.5

(n) nuScenes to Argoverse

Method \ Range(m)	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	88.6 / 85.7	68.5 / 66.4	37.8 / 30.0	76.5 / 53.2	56.6 / 30.4	20.2 / 10.1
no retraining	39.5 / 37.0	21.6 / 17.4	3.0 / 3.0	28.6 / 5.1	17.6 / **9.1**	3.0 / 0.1
ST	65.2 / 59.5	32.6 / 25.9	12.4 / 9.1	**51.9** / **15.7**	24.3 / **9.1**	9.1 / **0.6**
Dreaming	**65.9** / **63.2**	**50.0** / **40.3**	**17.7** / **13.0**	50.7 / 10.6	**32.3** / 2.9	**9.6** / **0.6**
SN only	49.6 / 47.8	20.8 / 16.4	4.5 / 4.5	38.2 / 8.2	16.6 / 2.3	4.5 / **1.8**
SN + ST	56.7 / 55.5	23.3 / 18.0	4.7 / 3.4	50.3 / **13.3**	17.4 / **9.1**	3.3 / 0.5
SN + Dreaming	**64.3** / **61.8**	**45.7** / **40.9**	**19.4** / **14.1**	**50.8** / 11.0	**33.4** / 3.0	**12.8** / 0.8

(o) nuScenes to Lyft

Method \ Range(m)	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	89.2 / 88.9	78.4 / 77.9	67.2 / 65.5	88.4 / 79.0	74.8 / 54.6	54.7 / 27.3
no retraining	52.3 / 51.0	41.1 / 37.9	22.9 / 15.1	49.5 / 17.9	36.8 / 10.5	15.1 / 9.1
ST	**79.7** / 78.0	65.9 / 58.0	39.1 / 30.2	**77.0** / 20.2	57.0 / **13.0**	30.0 / **10.4**
Dreaming	79.6 / **78.1**	**66.8** / **63.2**	**46.4** / **37.8**	**77.0** / **20.7**	**62.1** / 10.4	**36.4** / 5.2
SN only	62.3 / 61.7	41.6 / 39.9	16.7 / 15.4	60.1 / 26.8	32.1 / 5.7	15.6 / 9.1
SN + ST	78.6 / 77.9	56.1 / 53.7	23.1 / 13.9	**77.3** / **32.4**	53.0 / 13.8	20.4 / 3.0
SN + Dreaming	**78.7** / **78.0**	**64.7** / **57.5**	**39.7** / **36.1**	**77.3** / 31.6	**56.0** / **16.0**	**29.6** / **11.0**

(p) nuScenes to Waymo

Method \ Range(m)	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	81.6 / 81.5	71.7 / 71.2	58.2 / 50.8	80.8 / 68.8	69.1 / 54.9	47.0 / 27.2
no retraining	62.2 / 61.2	50.7 / 42.1	24.9 / 22.5	60.5 / 25.3	41.7 / 13.8	21.3 / 4.5
ST	**80.7** / **71.9**	68.7 / 60.9	42.5 / 39.6	71.6 / **34.5**	60.7 / **24.6**	37.0 / 8.8
Dreaming	**80.7** / **71.9**	**69.4** / **61.3**	**50.1** / **45.5**	**71.7** / 33.5	**61.2** / 24.3	**39.3** / **10.7**
SN only	71.1 / 70.3	50.6 / 41.8	25.1 / 16.6	61.1 / 34.8	40.5 / 14.4	16.1 / 2.8
SN + ST	80.7 / 79.1	67.4 / 59.2	41.3 / 32.1	71.4 / **44.0**	58.2 / 27.3	29.7 / 7.4
SN + Dreaming	**81.1** / **79.2**	**69.5** / **61.5**	**50.1** / **41.3**	**71.7** / 42.2	**60.7** / **27.6**	**38.5** / **14.4**

(q) Waymo to KITTI

Range(m) / Method	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	90.0 / 89.9	81.0 / 79.9	40.4 / 36.3	89.0 / 78.0	70.3 / 51.5	26.6 / 9.8
no retraining	88.4 / 87.0	64.7 / 55.2	14.3 / 8.5	45.8 / 10.4	22.9 / 3.1	2.5 / 0.8
ST	89.3 / 88.3	70.5 / 62.8	25.6 / 18.1	**50.7** / 10.1	28.6 / 11.9	4.3 / 2.3
Dreaming	**89.5 / 88.7**	**72.6 / 65.8**	**27.8 / 19.7**	49.8 / **16.3**	**31.9 / 12.3**	**5.0 / 3.0**
SN only	88.6 / 88.5	63.0 / 61.9	11.6 / 10.7	84.1 / 53.8	51.7 / 28.3	8.4 / 5.1
SN + ST	**89.7 / 89.7**	72.5 / 69.0	19.8 / 18.6	**87.1** / 60.6	**60.8 / 38.6**	13.8 / **6.2**
SN + Dreaming	89.6 / 89.6	**73.5 / 72.3**	**22.1 / 19.6**	86.3 / **63.1**	58.0 / 36.4	**14.0** / 4.0

(r) Waymo to Argoverse

Range(m) / Method	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	88.6 / 85.7	68.5 / 66.4	37.8 / 30.0	76.5 / 53.2	56.6 / 30.4	20.2 / 10.1
no retraining	84.0 / 76.3	54.4 / 51.3	24.7 / 19.6	69.6 / 28.4	40.6 / 13.8	15.9 / 1.6
ST	85.7 / **78.6**	55.5 / 52.9	23.4 / 21.3	**70.7 / 29.8**	**44.3 / 14.0**	14.9 / 2.1
Dreaming	**85.8** / 78.3	**56.7 / 54.6**	**28.3 / 22.3**	64.8 / 28.8	44.1 / 13.5	**17.3 / 3.0**
SN only	83.3 / 74.9	54.3 / 47.2	19.2 / 16.3	69.3 / 29.2	43.2 / 16.9	10.3 / 3.0
SN + ST	85.2 / 78.0	55.4 / 52.8	22.8 / 20.4	**73.9 / 37.6**	**45.8 / 19.8**	14.8 / **9.1**
SN + Dreaming	**85.4 / 78.1**	**56.0 / 53.6**	**27.7 / 22.1**	71.8 / 35.2	45.5 / 16.8	**17.6 / 9.1**

(s) Waymo to Lyft

Range(m) / Method	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	89.2 / 88.9	78.4 / 77.9	67.2 / 65.5	88.4 / 79.0	74.8 / 54.6	54.7 / 27.3
no retraining	80.8 / **80.8**	67.8 / 66.8	45.3 / 43.0	78.3 / 55.3	56.7 / 31.2	34.4 / 10.5
ST	80.7 / 80.6	74.6 / 73.4	54.0 / 48.1	78.9 / 54.9	**65.4 / 32.8**	42.9 / 17.2
Dreaming	**86.8 / 80.8**	**75.4 / 74.0**	**55.5 / 49.6**	**79.2 / 55.5**	**65.4** / 32.6	**44.6 / 17.3**
SN only	80.5 / 80.2	66.4 / 65.1	38.4 / 37.1	78.0 / 55.9	55.8 / 31.4	33.5 / 15.4
SN + ST	86.8 / 80.7	74.2 / 73.3	45.5 / 44.7	79.8 / 63.9	65.4 / **38.1**	36.1 / 14.5
SN + Dreaming	**87.3 / 81.1**	**75.3 / 73.8**	**48.0 / 47.0**	**80.3 / 64.6**	**66.3 / 38.1**	**38.4 / 18.4**

(t) Waymo to nuScenes

Range(m) / Method	IoU 0.5			IoU 0.7		
	0-30	30-50	50-80	0-30	30-50	50-80
in-domain	65.6 / 64.5	18.4 / 17.2	3.5 / 2.8	57.8 / 37.0	15.8 / 5.3	2.6 / 0.8
no retraining	46.4 / 43.2	3.0 / 3.0	0.2 / 0.0	42.6 / 23.9	3.0 / 0.2	0.1 / 0.0
ST	51.4 / 46.5	10.7 / 9.1	3.0 / 0.3	45.8 / **26.4**	9.1 / **3.0**	3.0 / 0.0
Dreaming	**53.8 / 48.0**	**12.4 / 10.0**	**9.1 / 9.1**	**47.4** / 23.7	**11.1 / 3.0**	**9.1 / 0.8**
SN only	45.8 / 43.5	9.1 / 9.1	0.1 / 0.0	42.9 / 23.5	9.1 / 0.2	0.0 / **0.0**
SN + ST	51.0 / 45.8	9.1 / 9.1	1.0 / 0.3	44.7 / 25.6	9.1 / **9.1**	1.0 / **0.0**
SN + Dreaming	**53.7 / 47.6**	**12.4 / 10.2**	**2.8 / 0.8**	**46.8 / 26.5**	**11.2** / 2.3	**2.3 / 0.0**