FAST3D: Flow-Aware Self-Training for 3D Object Detectors

Christian Fruhwirth-Reisinger[1,2]
christian.reisinger@icg.tugraz.at

Michael Opitz[1,2,3]
micopitz@amazon.de

Horst Possegger[2]
possegger@icg.tugraz.at

Horst Bischof[1,2]
bischof@icg.tugraz.at

[1] Christian Doppler Laboratory for Embedded Machine Learning

[2] Institute of Computer Graphics and Vision
Graz University of Technology

[3] Amazon

Abstract

In the field of autonomous driving, self-training is widely applied to mitigate distribution shifts in LiDAR-based 3D object detectors. This eliminates the need for expensive, high-quality labels whenever the environment changes (*e.g.* geographic location, sensor setup, weather condition). State-of-the-art self-training approaches, however, mostly ignore the temporal nature of autonomous driving data.

To address this issue, we propose a flow-aware self-training method that enables unsupervised domain adaptation for 3D object detectors on continuous LiDAR point clouds. In order to get reliable pseudo-labels, we leverage scene flow to propagate detections through time. In particular, we introduce a flow-based multi-target tracker that exploits flow consistency to filter and refine resulting tracks. The emerged precise pseudo-labels then serve as a basis for model re-training. Starting with a pre-trained KITTI model, we conduct experiments on the challenging Waymo Open Dataset to demonstrate the effectiveness of our approach. Without any prior target domain knowledge, our results show a significant improvement over the state-of-the-art.

1 Introduction

In order to safely navigate through traffic, self-driving vehicles need to robustly detect surrounding objects (*e.g.* vehicles, pedestrians, cyclists). State-of-the-art approaches leverage deep neural networks operating on LiDAR point clouds [41, 42, 43]. However, training such 3D object detection models usually requires a huge amount of manually annotated high-quality data [2, 9, 45]. Unfortunately, the labelling effort for 3D LiDAR point clouds is very time-consuming and consequently also expensive – a major drawback for real-world applications. Most datasets are recorded in specific geographic locations (*e.g.* Germany [9, 10]), with a fixed sensor configuration and under good weather conditions. Applied to new data collected in other locations (*e.g.* USA [45]), with a different sensor (*e.g.* sparser resolution [2]) or under adverse weather conditions (*i.e.* fog, rain, snow), 3D detectors suffer from
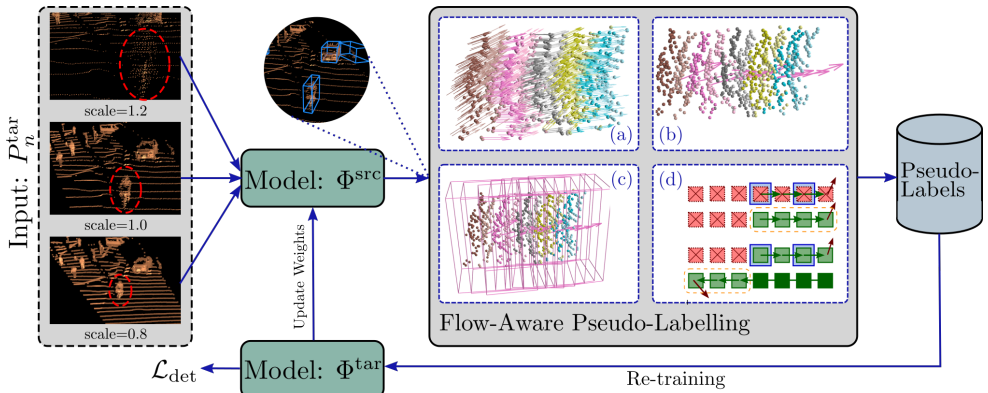
Figure 1: FAST3D scheme. The initial model Φ^{src}, pre-trained on source data, generates non-confident detections on the target data for different scales. We leverage scene flow information (a) to calculate box flow (b), *i.e.* the mean flow of an object. This allows us to increase the recall and propagate labels through time (c). A final refinement step (d) reduces false positives and false negatives to obtain high-quality pseudo-labels for re-training. Points with the same colour originate from the same point cloud P_n^{tar}.

distribution shifts (domain gap). This mostly causes serious performance drops [50], which in turn leads to unreliable recognition systems.

This can be mitigated by either manual or semi-supervised annotation [30, 43] of representative data, each time the sensor setup or area of operation changes. However, this is infeasible for most real-world scenarios given the expensive labelling effort. A more general solution to avoid the annotation overhead is unsupervised domain adaptation (UDA), which adapts a model pre-trained on a label-rich source domain to a label-scarce target domain. Hence, no or just a small number of labelled frames from the target domain are required.

For 3D object detection, UDA via self-training has gained a lot of attention [40, 58, 53]. Similar to 2D approaches [5, 17, 58], the idea is to use a 3D detector pre-trained on a labelled source dataset and apply it on the target dataset to obtain pseudo-labels. These labels are leveraged to re-train the model. Both steps, label generation and re-training, are repeated until convergence. However, generating reliable pseudo-labels is a non-trivial task.

Although most real-world data is continuous in nature, this property is rarely exploited for UDA of 3D object detectors. As a notable exception, [53] leverages a probabilistic offline tracker. Though simple and effective, a major weakness of probabilistic trackers is that they heavily depend on the detection quality. Because of the domain gap, however, the detection quality usually degrades significantly. Additionally, these trackers require a hand-crafted motion model which must be adjusted manually. These limitations result in unreliable pseudo-labels driven by missing and falsely classified objects.

To overcome these issues, we present our **F**low-**A**ware **S**elf-**T**raining approach for **3D** object detection (*FAST3D*), leveraging scene flow [27, 28, 53] for robust pseudo-labels, as illustrated in Fig. 1. Scene flow estimates per-point correspondences between consecutive point clouds and calculates the motion vector for each point. Furthermore, although trained on synthetic data [29] only, scene flow estimators already achieve a favourable accuracy on real-world road data [53]. Thus, we investigate scene flow for UDA. In particular, we will show that scene flow allows us to propagate pseudo-labels reliably, recover missed objects

and discard unreliable detections. This results in significantly improved pseudo-label quality which boosts the performance of self-training 3D detectors drastically.

We conduct experiments on the challenging Waymo Open Dataset (WOD) [45] considering two state-of-the-art 3D detectors, PointRCNN [41] and PV-RCNN [42], pre-trained on the much smaller KITTI dataset [9, 10]. Without any prior target domain knowledge (*e.g.* [50]), nor the need for source domain data (*e.g.* [58]), we surpass the state-of-the-art by a significant margin.

2 Related Work

3D Object Detection   3D object detectors localize and classify an unknown number of objects within a 3D environment. Commonly, the identified objects are represented as tightly fitting oriented bounding boxes. Most recent 3D detectors, trained and evaluated on autonomous driving datasets, operate on LiDAR point clouds.

One way to categorize them is by their input representation. Voxel-based approaches [7, 22, 23, 51, 56, 57, 61, 64] rasterize the input space and assign points of irregular and sparse nature to grid cells of a fixed size. Afterwards, they either project these voxels directly to the bird's eye view (BEV) or first learn feature representations by leveraging 3D convolutions and project them to BEV afterwards. Finally, a conventional 2D detection head predicts bounding boxes and class labels. Another line of work are point-based detectors [32, 41, 44, 60]. In order to generate proposals directly from points, they leverage PointNet [54, 55] to extract point-wise features. Hybrid approaches [5, 12, 42, 43, 59, 65], on the other hand, seek to leverage the advantages of both of the aforementioned strategies. In contrast to point cloud-only approaches, multi-modal detectors [4, 20, 25, 26, 33, 55] utilize 2D images complementary to LiDAR point clouds. The additional image information can be beneficial to recognise small objects.

Since most state-of-the-art approaches operate on LiDAR point clouds only, we also focus on this type of detectors. In particular, we demonstrate our self-training approach using PointRCNN [41] and PV-RCNN [42]. Both detectors have already been used in UDA settings for self-driving vehicles [50, 58] and achieve state-of-the-art robustness and accuracy.

Scene Flow   Scene flow represents the 3D motion field in the scene [47]. Recently, a relatively new area of research aims to estimate point-wise motion predictions in an end-to-end manner directly from raw point clouds [1, 27, 53]. With few exceptions [28], most approaches process two consecutive point clouds as input. A huge benefit of data-driven scene flow models, especially regarding UDA, is their ability to learn in an unsupervised manner [24, 31, 53]. The biggest drawback of these networks is their huge memory consumption which limits the number of input points. To address this, [16] proposes a light-weight model applicable to the dense Waymo Open Dataset [45] point clouds.

In our work, we use the 3D motion field to obtain robust and reliable pseudo-labels for UDA by leveraging the motion consistency of sequential detections. To this end, we utilize PointPWC-Net [53] which achieves state-of-the-art scene flow estimation performance.

Unsupervised Domain Adaptation (UDA)   The common pipeline for UDA is to start with a model trained in a fully supervised manner on a label-rich source domain and adapt it on data from a label-scarce target domain in an unsupervised manner. Hence, the goal is to

close the gap between both domains. There is already a large body of literature on UDA for 2D object detection in driving scenes [6, 13, 14, 18, 37, 39, 49, 54, 56]. Due to the growing number of publicly available large-scale autonomous driving datasets, UDA on 3D point cloud data has gained more interest recently [1, 15, 36, 62].

For LiDAR-based 3D object detection, [50] demonstrate serious domain gaps between various datasets, mostly caused by different sensor setups (*e.g.* resolution or mounting position) or geographic locations (*e.g.* Germany $\rightarrow$ USA). A popular solution to UDA is self-training, as for the 2D case [3, 17, 58]. For example, [58] initially train the detector with random object scaling and an additional score prediction branch. Afterwards, pseudo-labels are updated in a cyclic manner considering previous examples. In order to overcome the need for source data, [40] performs test-time augmentation with multiple scales. The best matching labels are selected by checking motion coherency.

In contrast to [53], where temporal information is exploited by probabilistic tracking, we propose to leverage scene flow. This enables us to reliably extract pseudo-labels despite initially low detection quality by exploiting motion consistency. As in [40], we also utilize test-time augmentation to overcome scaling issues, but we only need two additional scales.

3   Flow-Aware Self-Training

We now introduce our Flow-Aware Self-Training approach *FAST3D*, which consists of four steps as illustrated in Fig. 1. First, starting with a model trained on the source domain, we obtain initial 3D object detections for sequences of the target domain (Sec. 3.1). Second, we leverage scene flow to propagate these detections throughout the sequences and obtain tracks, robust to the initial detection quality (Sec. 3.2). Third, we recover potentially missed tracks and correct false positives in a refinement step (Sec. 3.3). Finally, we extract pseudo-labels for self-training to improve the initial model (Sec. 3.4).

Problem Statement   Given a 3D object detection model Φ^{src} pre-trained on N^{src} source domain frames $\{(P_n^{\text{src}}, L_n^{\text{src}})\}_{n=1}^{N^{\text{src}}}$, our task is to adapt to unseen target data with N^{tar} unlabelled sequences $\{S_i^{\text{tar}}\}_{i=1}^{N^{\text{tar}}}$, where $S_i^{\text{tar}} = \{P_n^{\text{tar}}\}_{n=1}^{N_i^{\text{tar}}}$ with varying length N_i^{tar}. Here, P_n and L_n denote the point cloud and corresponding labels of the nth frame. To obtain the target detection model Φ^{tar}, we apply self-training which assumes that both domains contain the same set of classes (*i.e.* vehicle/car, pedestrian, cyclist) but these are drawn from different distributions. In the following, we show how to self-train a detection model Φ^{tar} without access to source data or target label statistics (as *e.g.* [50]) and without modifying the source model in any way (as *e.g.* [58]) to achieve performance beyond the state-of-the-art. Because we only work with target domain data, we omit the domain superscripts to improve readability.

3.1   Pseudo-Label Generation

The first step, as in all self-training approaches (*e.g.* [40, 58, 53]), is to run the initial source model Φ^{src} on the target data and keep high-confident detections. More formally, the jth detection at frame n is represented by its bounding box $\mathbf{b}_{j,n} = (\mathbf{p}, \mathbf{d}, \theta)$ and confidence score $c_{j,n}^{\mathbf{b}}$. Each box is defined by its centre position $\mathbf{p} = (c_x, c_y, c_z)$, dimension $\mathbf{d} = (l, w, h)$ and heading angle θ. Note that we can use any vanilla 3D object detector, since we do not change the initial model at all, not even by adapting the mean object size of anchors (as

in [58]). In order to deal with different object sizes between various datasets without pre-training Φ^{src} on scaled source data [50, 58], we leverage test-time augmentation instead. In particular, we feed Φ^{src} with three scales (*i.e.* 0.8, 1.0, 1.2) of the same point cloud, similar to [40]. However, [40] requires to run the detector on **125** different scale levels (due to different scaling factors along each axis). By leveraging scene flow, we only need **three** scales to robustly handle both larger and smaller objects. We combine the detection outputs at all three scales via non-maximum suppression (NMS) and threshold the detections at a confidence of $c_{j,n}^{\mathbf{b}} \geq 0.8$ to obtain the initial high-confident detections.

3.2 Flow-Aware Pseudo-Label Propagation

Keeping only high-confident detections of the source model gets rid of false positives (FP) but also results in a lot of false negatives (FN). While this can partially be addressed via multi-target tracking (MTT), *e.g.* [63], standard MTT approaches (such as [52]) are not suitable for low quality detections because of their hand-crafted motion prediction. Due to the domain gap, however, the detection quality of the source model will inevitably be low.

To overcome this issue, we introduce our scene flow-aware tracker. We define a set of tracks $T = \{t_1, t_2, \ldots, t_k\}$ where each track t_k contains a list of bounding boxes and is represented by its current state $\mathbf{x}_{k,n}$, *i.e.* its bounding box, and track confidence score $c_{k,n}^{\mathbf{t}}$. Following the tracking-by-detection paradigm, we use the high-confident detections and match them in subsequent frames. Instead of hand-crafted motion models, however, we utilize scene flow to propagate detections. More formally, given two consecutive point clouds P_{n-1} and P_n, PointPWC-Net [53] estimates the motion vector $\mathbf{v}_{i,n-1} = (v_x, v_y, v_z)_{i,n-1}$ for each point $\mathbf{p}_{i,n-1} \in P_{n-1}$. We then average all motion vectors within a track's bounding box $\mathbf{x}_{k,n-1}$ to compute its mean flow $\mathbf{v}_{k,n-1}$. To get the predicted state $\mathbf{x}_{k,n|n-1}$ for the current frame n, we estimate the position of each track as $\mathbf{p}_{k,n|n-1} = \mathbf{p}_{k,n-1} + \mathbf{v}_{k,n-1}$. We then assign detections to tracks via the Hungarian algorithm [21], where we use the intersection over union between the detections $\mathbf{b}_{j,n}$ and the predicted tracks $\mathbf{x}_{k,n|n-1}$.

We initialize a new track for each unassigned detection. This naive initialisation ensures that we include all object tracks, whereas potentially wrong tracks can easily be filtered in our refinement step (Sec. 3.3). Initially, we set a track's confidence $c_{k,n}^{\mathbf{t}}$ to the detection confidence $c_{j,n}^{\mathbf{b}}$. For each assigned pair, we perform a weighted update based on the confidence scores as $\mathbf{x}_{k,n|n} = \frac{c_{k,n-1}^{\mathbf{t}} \mathbf{x}_{k,n|n-1} + c_{j,n}^{\mathbf{b}} \mathbf{b}_{j,n}}{c_{k,n-1}^{\mathbf{t}} + c_{j,n}^{\mathbf{b}}}$ and update the track confidence $c_{k,n}^{\mathbf{t}} = \frac{(c_{k,n-1}^{\mathbf{t}})^2 + (c_{j,n}^{\mathbf{b}})^2}{c_{k,n-1}^{\mathbf{t}} + c_{j,n}^{\mathbf{b}}}$. Note that we update the track's heading angle only if the orientation change is less than $30°$, otherwise we keep its previous orientation. For boxes with only very few point correspondences, the flow estimates may be noisy. To suppress such flow outliers, we allow a maximum change in velocity of 1.5 m/s, along with the maximum orientation change of $30°$. If the estimated flow exceeds these limits, we keep the previous estimate.

For track termination, we distinguish moving (*i.e.* $\mathbf{v}_{k,n-1} > 0.8$ m/s) and static objects. For moving objects, we terminate their tracks if there are no more points within their bounding box. Static object tracks are kept alive as long as they are within the field-of-view. This ensures that even long term occlusions can be handled, as such objects will eventually contain points or be removed during refinement.

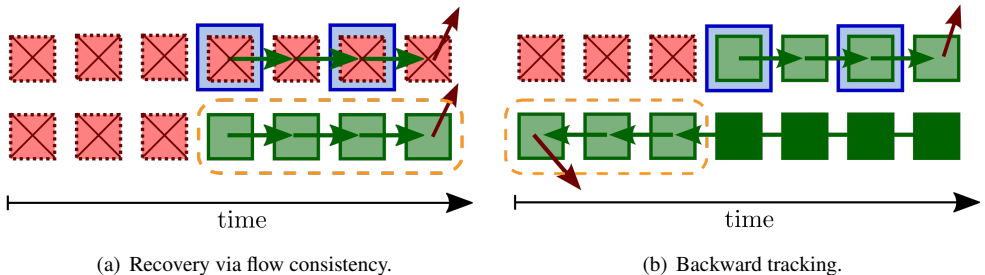(a) Recovery via flow consistency.   (b) Backward tracking.

Figure 2: Track refinement to address the initially low detection quality on the target domain. The valid track, false negative (missed) and true positive detections are illustrated in green, red and blue, respectively. Green and red arrows show valid and inconsistent flow (high change in orientation/velocity), respectively. (a) Flow consistency recovers object tracks with low detection hit ratio. (b) Starting from the first detection, backward tracking propagates the bounding box back in time to recover missing tails. Refined tracks are marked by the dashed orange regions.

3.3   Pseudo-Label Refinement

Due to our simplified track management (initialisation and termination), we now have tracks covering most of the visible objects but suffer from several false positive tracks. However, these can easily be corrected in the following refinement step.

Track Filtering and Correction   We first compute the *hit ratio* for each track, *i.e.* the number of assigned detections divided by the track's length. Tracks are removed if their hit ratio is less than 0.3. Additionally, we discard tracks shorter than 0.5 s (*i.e.* 5 frames at the typical LiDAR frequency of 10 Hz) and tracks which detections never exceed 15 points (to suppress spurious detections).

In our experiments, we observed that detections are most accurate on objects with more points (as opposed to the confidence score which is usually less reliable due to the domain gap). Thus, we sort a track's assigned detections by the number of contained points and compute the average dimension over the top three. We then apply this average dimension to all boxes of the track. Additionally, for static cars we also apply the average position and heading angle to all boxes of the track since these properties cannot change for rigid static objects.

Track Recovery   This conservative filtering of unreliable tracks ensures a lower number of false positives. However, true positive tracks with only very few detections might be removed prematurely. In order to recover these, we leverage *flow consistency*. As illustrated in Fig. 2(a), we consider removed tracks with at least two non-overlapping detections (*i.e.* moving objects). For these tracks, we propagate the first detection by solely utilizing the scene flow estimates. If this flow-extrapolated box overlaps with the other detections (minimum IoU of 0.3), we recover this track.

Backward Completion   A common drawback of the tracking-by-detection paradigm is late initialization caused by missing detections at the beginning of the track. To overcome this issue, we look backwards for each track as illustrated in Fig. 2(b) to extend missing tails.

Hence, we leverage scene flow in the opposite direction and propagate the bounding box back in time. We propagate backwards as long as the flow is consistent, meaning no unreasonable changes in velocity or direction, and the predicted box location contains points. To avoid including the ground plane in the latter condition, we count only points within the upper 70% of the bounding box volume.

3.4 Self-Training

We use the individual bounding boxes of the refined tracks as high-quality pseudo-labels to re-train the detection model. To this end, we use standard augmentation strategies, *i.e.* randomized world scaling, flipping, rotation, as well as ground truth sampling [42, 56]. As we don't need to modify the initial model Φ^{src}, we also use its detection loss $\mathcal{L}_{\mathrm{det}}$ without modifications for re-training.

4 Experiments

In the following, we present the results of our flow-aware self-training approach using two vanilla 3D object detectors. Additional evaluations demonstrating the improved pseudo-label quality are included in the supplemental material.

4.1 Datasets and Evaluation Details

Datasets   We conduct experiments on the challenging Waymo Open Dataset (WOD) [45] with a source model Φ^s pre-trained on the KITTI dataset [9, 10]. We sample 200 sequences ($\sim$25%) from the WOD training set for pseudo-label generation and 20 sequences ($\sim$10%) from the WOD validation set for testing. With this evaluation, we cover various sources of domain shifts: geographic location (Germany $\rightarrow$ USA), sensor setup (Velodyne $\rightarrow$ Waymo LiDAR) and weather (broad daylight $\rightarrow$ sunny, rain, night). In contrast to recent work, we do not only consider the car/vehicle class, but also pedestrians and cyclists. Because the initial model Φ^s is trained on front view scans only (available KITTI annotations), we stick to this setup for evaluation.

Evaluation Metrics   We follow the official WOD evaluation protocol [45] and report the average precision (AP) for the intersection over union (IoU) thresholds 0.7 and 0.5 for vehicles, as well as 0.5 and 0.25 for pedestrians and cyclists. AP is reported for both bird's eye view (BEV) and 3D – denoted as AP_{BEV} and AP_{3D} – for different ranges: $0\,\mathrm{m} - 30\,\mathrm{m}$, $30\,\mathrm{m} - 50\,\mathrm{m}$ and $50\,\mathrm{m} - 75\,\mathrm{m}$. Additionally, we calculate the *closed gap* as in [58].

Comparisons   The lower performance bound is to directly apply the source model Φ^s on the target data, denoted source only (SO). The fully supervised model (FS) trained on the whole target data ($\sim$ 25% of WOD) on the other hand defines the upper performance bound. Statistical normalization (SN) [50] is the state-of-the-art on UDA for 3D object detection and leverages target data statistics. The recent DREAMING [53] approach also exploits temporal information for UDA. Finally, we also compare against the few shot [50] approach which adapts the model using a few fully labelled sequences of the target domain.

Implementation details   We demonstrate FAST3D with two detectors, PointRCNN [41] and PV-RCNN [42] from OpenPCDet [46], configured as in their official implementations. To compensate for the ego-vehicle motion, we project all frames of a sequence into the world coordinate system by computing the ego-vehicle poses [45]. We slightly increase the field-of-view (FOV) from approximately $45°$ to $60°$, which simplifies handling pseudo-label tracks at the edge of the narrower FOV. In each self-training cycle, we re-train the detector for 4 epochs with Adam [19] and a learning rate of 3×10^{-4} until we reach convergence (*i.e.* PointRCNN 2 cycles, PV-RCNN 3 cycles).

We estimate the scene flow using PointPWC-Net [53]. Although PointPWC-Net, similar to other flow estimators [24, 51], could be fine-tuned in a self-supervised fashion, we use the off-the-shelf model pre-trained only on synthetic data [29]. This allows us to demonstrate the benefits and simplicity of leveraging 3D motion without additional fine-tuning. To prepare the point clouds for scene flow estimation, we use RANSAC [8] to remove ground points (as in [53]) and randomly subsample the larger input point cloud to match the smaller one.

4.2   Empirical Results

In Table 1 we compare FAST3D for two detectors with the source only (SO) and fully supervised (FS) strategies, which define the lower and upper performance bound, respectively. Across all classes and different IoU thresholds, we manage to close the domain gap significantly (36% to 87%), achieving almost the fully supervised oracle performance although our approach is unsupervised and does not rely on any prior knowledge about the target domain. To the best of our knowledge, we are the first to additionally report adaptation performance for both pedestrians and cyclists as these vulnerable road users should not be neglected in evaluations. We focus this evaluation on AP_{3D} because (in contrast to AP_{BEV}) this metric also penalizes estimation errors along the z-axis (*i.e.* vertical centre and height). Consequently, AP_{BEV} scores are usually much higher than AP_{3D}.

Since reporting the closed domain gap has only been proposed very recently [58], no other approaches reported this closure for KITTI→WOD yet. We can, however, relate the results for the most similar setting, *i.e.* adaptation of PV-RCNN for the car/vehicle class, where we achieve a closed gap of 76.5% (KITTI→WOD, *vehicles* within all sensing ranges) in contrast to 70.7% of [58] (WOD→KITTI, *cars* within KITTI's *moderate* difficulty). Despite the different settings, we achieve a favourable domain gap closure while evaluating on the more challenging KITTI→WOD. In particular, according to [50], starting from KITTI as the source domain is the most difficult setting for UDA, as it has orders of magnitude fewer samples than other datasets. Additionally, KITTI contains only sunny daylight data and its *car* class does not include trucks, vans or buses which are, however, contained in the WOD *vehicle* class.

Table 2 reports the results in AP_{3D} split by their detection range for different intersection over union (IoU) thresholds. We can observe that our self-training pipeline significantly improves the initial model on all ranges. The only notable outlier are cyclists within the far range of 50–75 meters, where even after re-training both detectors achieve rather low scores. This is due to the low number of far range cyclists within the WOD validation set, *i.e.* only a few detection failures already drastically degrade the AP_{3D} for the *cyclist* class.

	Class	PointRCNN				PV-RCNN			
		SO	FAST3D	FS	Closed Gap	SO	FAST3D	FS	Closed Gap
AP_{3D} 0.5 0.5 0.7	Vehicle	8.0	58.1	65.6	87.0 %	10.3	63.6	80.0	76.5 %
	Pedestrian	18.9	36.8	47.8	62.0 %	13.7	44.7	74.4	51.1 %
	Cyclist	24.2	48.0	60.0	66.5 %	14.6	35.6	72.9	36.0 %
AP_{3D} 0.25 0.25 0.5	Vehicle	60.0	77.3	80.9	82.8 %	50.3	85.1	95.2	77.5 %
	Pedestrian	30.8	41.7	52.2	50.9 %	21.6	57.2	85.4	55.8 %
	Cyclist	31.8	66.2	69.0	92.5 %	32.5	49.4	75.5	39.3 %

Table 1: Our FAST3D compared to lower (source-only, SO) and upper (fully-supervised, FS) bound, on KITTI → Waymo Open Dataset (WOD) for different IoU thresholds.

	Class	PointRCNN			PV-RCNN		
		0m - 30m	30m - 50m	50m - 75m	0m - 30m	30m - 50m	50m - 75m
AP_{3D} 0.5 0.5 0.7	Vehicle	70.3	58.7	37.0	74.9	64.6	44.0
	Pedestrian	66.2	45.7	14.6	66.5	55.0	22.7
	Cyclist	76.7	65.7	4.9	64.4	46.0	1.4
AP_{3D} 0.25 0.25 0.5	Vehicle	86.9	79.3	63.5	91.2	85.9	70.4
	Pedestrian	74.1	50.3	17.4	78.0	67.3	36.4
	Cyclist	93.9	76.2	33.6	90.6	54.4	5.0

Table 2: Detailed results at different sensing ranges (KITTI → WOD) for two detectors at different IoU thresholds.

4.3   Comparison with the State-of-the-Art

To fairly compare with the state-of-the-art in Table 3, we follow the common protocol and report both AP_{3D} and AP_{BEV} on the *vehicle* class. We clearly outperform the best approach (statistical normalization), even though this method utilizes target data statistics and is thus considered weakly supervised, whereas our approach is unsupervised. We also outperform the only temporal 3D pseudo-label approach [53] by a huge margin for all ranges, especially at the high-quality IoU ≥ 0.7. Moreover, we perform on par with the few shot approach from [50] on close-range data and even outperform it on medium and far ranges.

Method	IoU 0.7			IoU 0.5		
	0m - 30m	30m - 50m	50m - 75m	0m - 30m	30m - 50m	50m - 75m
SO	29.2 / 10.0	27.2 / 8.0	24.7 / 4.2	67.8 / 66.8	70.2 / 63.9	48.0 / 38.5
SN [50]	**87.1** / 60.1	**78.1** / 54.9	46.8 / 25.1	-	-	-
DREAMING [53]	51.4 / 13.8	44.5 / 16.7	25.6 / 7.8	81.1 / **78.5**	69.9 / 61.8	50.2 / 41.0
FAST3D (ours)	81.7 / **70.3**	75.4 / **58.7**	**52.4 / 37.0**	**87.2** / 74.9	**81.2 / 64.6**	**68.6 / 44.0**
Few Shot [50]	88.7 / 74.1	78.1 / 57.2	45.2 / 24.3	-	-	-
FS	86.2 / 78.5	77.8 / 63.7	60.8 / 48.1	91.7 / 92.1	83.7 / 78.3	73.3 / 69.7

Table 3: Comparison to the state-of-the-art for KITTI → WOD. All approaches adapt PointRCNN. Following the common protocol, results are listed for the *vehicle* class at different sensing ranges reported in AP_{BEV} / AP_{3D}.

5   Conclusion

We presented a flow-aware self-training pipeline for unsupervised domain adaptation of 3D object detectors on sequential LiDAR point clouds. Leveraging motion consistency via scene flow, we obtain reliable and precise pseudo-labels at high recall levels. We do not exploit any prior target domain knowledge, nor do we need to modify the 3D detector in any way. As demonstrated in our evaluations, we surpass the current state-of-the-art in self-training for UDA by a large margin.

Acknowledgements

References

[1] Idan Achituve, Haggai Maron, and Gal Chechik. Self-Supervised Learning for Domain Adaptation on Point Clouds. In *Proc. WACV*, 2021.

[2] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A Multimodal Dataset for Autonomous Driving. In *Proc. CVPR*, 2020.

[3] Qi Cai, Yingwei Pan, Chong-Wah Ngo, Xinmei Tian, Lingyu Duan, and Ting Yao. Exploring Object Relation in Mean Teacher for Cross-Domain Detection. In *Proc. CVPR*, 2019.

[4] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3D Object Detection Network for Autonomous Driving. In *Proc. CVPR*, 2017.

[5] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast Point R-CNN. In *Proc. ICCV*, 2019.

[6] Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Domain Adaptive Faster R-CNN for Object Detection in the Wild. In *Proc. CVPR*, 2018.

[7] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection. In *Proc. AAAI*, 2021.

[8] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395, 1981.

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

[10] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets Robotics: The KITTI Dataset. *IJRR*, 2013.

[11] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. HPLFlowNet: Hierarchical Permutohedral Lattice FlowNet for Scene Flow Estimation on Large-Scale Point Clouds. In *Proc. CVPR*, 2019.

[12] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure Aware Single-stage 3D Object Detection from Point Cloud. In *Proc. CVPR*, 2020.

[13] Zhenwei He and Lei Zhang. Multi-Adversarial Faster-RCNN for Unrestricted Object Detection. In *Proc. ICCV*, 2019.

[14] Han-Kai Hsu, Chun-Han Yao, Yi-Hsuan Tsai, Wei-Chih Hung, Hung-Yu Tseng, Maneesh Singh, and Ming-Hsuan Yang. Progressive Domain Adaptation for Object Detection. In *Proc. WACV*, 2020.

[15] Maximilian Jaritz, Tuan-Hung Vu, Raoul de Charette, Emilie Wirbel, and Patrick Pérez. xMUDA: Cross-modal unsupervised domain adaptation for 3D semantic segmentation. In *Proc. CVPR*, 2020.

[16] Philipp Jund, Chris Sweeney, Nichola Abdo, Zhifeng Chen, and Jonathon Shlens. Scalable Scene Flow from Point Clouds in the Real World. *arXiv CoRR*, abs/2103.01306, 2021.

[17] Mehran Khodabandeh, Arash Vahdat, Mani Ranjbar, and William G. Macready. A Robust Learning Approach to Domain Adaptive Object Detection. In *Proc. ICCV*, 2019.

[18] Taekyung Kim, Minki Jeong, Seunghyeon Kim, Seokeon Choi, and Changick Kim. Diversify and Match: A Domain Adaptive Representation Learning Paradigm for Object Detection. In *Proc. CVPR*, 2019.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proc. ICLR*, 2015.

[20] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. Joint 3D Proposal Generation and Object Detection from View Aggregation. In *Proc. IROS*, 2018.

[21] H. W. Kuhn and Bryn Yaw. The Hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–97, 1955.

[22] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proc. CVPR*, 2019.

[23] Bo Li. 3D fully convolutional network for vehicle detection in point cloud. In *Proc. IROS*, 2017.

[24] Ruibo Li, Guosheng Lin, and Lihua Xie. Self-Point-Flow: Self-Supervised Scene Flow Estimation from Point Clouds with Optimal Transport and Random Walk. In *Proc. CVPR*, 2021.

[25] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep Continuous Fusion for Multi-Sensor 3D Object Detection. In *Proc. ECCV*, 2018.

[26] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-Task Multi-Sensor Fusion for 3D Object Detection. In *Proc. CVPR*, 2019.

[27] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. FlowNet3D: Learning Scene Flow in 3D Point Clouds. In *Proc. CVPR*, 2019.

[28] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. MeteorNet: Deep Learning on Dynamic 3D Point Cloud Sequences. In *Proc. ICCV*, 2019.

[29] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In *Proc. CVPR*, 2016.

[30] Qinghao Meng, Wenguan Wang, Tianfei Zhou, Jianbing Shen, Luc Van Gool, and Dengxin Dai. Weakly Supervised 3D Object Detection from Lidar Point Cloud. In *Proc. ECCV*, 2020.

[31] Himangi Mittal, Brian Okorn, and David Held. Just Go With the Flow: Self-Supervised Scene Flow Estimation. In *Proc. CVPR*, 2020.

[32] Jiquan Ngiam, Benjamin Caine, Wei Han, Brandon Yang, Yuning Chai, Pei Sun, Yin Zhou, Xi Yi, Ouais Alsharif, Patrick Nguyen, Zhifeng Chen, Jonathon Shlens, and Vijay Vasudevan. StarNet: Targeted Computation for Object Detection in Point Clouds. *arXiv CoRR*, abs/1908.11069, 2019.

[33] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum Point-Nets for 3D Object Detection from RGB-D Data. In *Proc. CVPR*, 2017.

[34] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. CVPR*, 2017.

[35] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proc. NeurIPS*, 2017.

[36] Can Qin, Haoxuan You, L. Wang, C.-C. Jay Kuo, and Yun Fu. PointDAN: A Multi-Scale 3D Domain Adaption Network for Point Cloud Representation. In *Proc. NeurIPS*, 2019.

[37] A. L. Rodriguez and K. Mikolajczyk. Domain Adaptation for Object Detection via Style Consistency. In *Proc. BMVC*, 2019.

[38] Aruni RoyChowdhury, Prithvijit Chakrabarty, Ashish Singh, SouYoung Jin, Huaizu Jiang, Liangliang Cao, and Erik Learned-Miller. Automatic Adaptation of Object Detectors to New Domains Using Self-Training. In *Proc. CVPR*, 2019.

[39] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Strong-Weak Distribution Alignment for Adaptive Object Detection. In *Proc. CVPR*, 2019.

[40] Cristiano Saltori, Stéphane Lathuiliére, Nicu Sebe, Elisa Ricci, and Fabio Galasso. SF-UDA3D: Source-Free Unsupervised Domain Adaptation for LiDAR-Based 3D Object Detection. In *Proc. 3DV*, 2020.

[41] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud. In *Proc. CVPR*, 2019.

[42] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. In *Proc. CVPR*, 2020.

[43] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE TPAMI*, 2020.

[44] Weijing Shi and Raj Rajkumar. Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud. In *Proc. CVPR*, 2020.

[45] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *Proc. CVPR*, 2020.

[46] OpenPCDet Development Team. OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds. https://github.com/open-mmlab/OpenPCDet, 2020.

[47] S. Vedula, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. *IEEE TPAMI*, 27(3):475–480, 2005.

[48] Sean Walsh, Jason Ku, Alex D. Pon, and Steven. L. Waslander. Leveraging Temporal Data for Automatic Labelling of Static Vehicles. In *Proc. CRV*, 2020.

[49] Xudong Wang, Zhaowei Cai, Dashan Gao, and Nuno Vasconcelos. Towards Universal Object Detection by Domain Attention. In *Proc. CVPR*, 2019.

[50] Yan Wang, Xiangyu Chen, Yurong You, Li Erran, Bharath Hariharan, Mark Campbell, Kilian Q. Weinberger, and Wei-Lun Chao. Train in germany, test in the usa: Making 3d object detectors generalize. In *Proc. CVPR*, 2020.

[51] Yue Wang, Alireza Fathi, Abhijit Kundu, David A. Ross, Caroline Pantofaru, Tom Funkhouser, and Justin Solomon. Pillar-Based Object Detection for Autonomous Driving. In *Proc. ECCV*, 2020.

[52] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3D Multi-Object Tracking: A Baseline and New Evaluation Metrics. In *Proc. IROS*, 2020.

[53] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. PointPWC-Net: Cost Volume on Point Clouds for (Self-) Supervised Scene Flow Estimation. In *Proc. ECCV*, 2020.

[54] C. Xu, Xingran Zhao, Xin Jin, and Xiu-Shen Wei. Exploring Categorical Regularization for Domain Adaptive Object Detection. In *Proc. CVPR*, 2020.

[55] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation. In *Proc. CVPR*, 2018.

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

[57] Bin Yang, Wenjie Luo, and Raquel Urtasun. PIXOR: Real-time 3D Object Detection from Point Clouds. In *Proc. CVPR*, 2018.

[58] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. ST3D: Self-training for Unsupervised Domain Adaptation on 3D Object Detection. In *Proc. CVPR*, 2021.

[59] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. STD: Sparse-to-Dense 3D Object Detector for Point Cloud. In *Proc. ICCV*, 2019.

[60] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3DSSD: Point-Based 3D Single Stage Object Detector. In *Proc. CVPR*, 2020.

[61] Maosheng Ye, Shuangjie Xu, and Tongyi Cao. HVNet: Hybrid Voxel Network for LiDAR Based 3D Object Detection. In *Proc. CVPR*, 2020.

[62] Li Yi, Boqing Gong, and T. Funkhouser. Complete & Label: A Domain Adaptation Approach to Semantic Segmentation of LiDAR Point Clouds. *arXiv CoRR*, abs/2007.08488, 2020.

[63] Yurong You, Carlos Andres Diaz-Ruiz, Yan Wang, Wei-Lun Chao, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Exploiting Playbacks in Unsupervised Domain Adaptation for 3D Object Detection. *arXiv CoRR*, abs/2103.14198, 2021.

[64] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *Proc. CVPR*, 2018.

[65] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-End Multi-View Fusion for 3D Object Detection in LiDAR Point Clouds. In *Proc. CRL*, 2020.

[66] Chenfan Zhuang, Xintong Han, Weilin Huang, and M. Scott. iFAN: Image-Instance Full Alignment Networks for Adaptive Object Detection. In *Proc. AAAI*, 2020.