Label-Efficient 3D Object Detection For Road-Side Units

Minh-Quan Dao[1], Holger Caesar[2], Julie Stephany Berrio[3], Mao Shan[3], Stewart Worrall[3], Vincent Frémont[4], Ezio Malis[1]

2024 IEEE Intelligent Vehicle Symposium (IV) | 979-8-3503-4881-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/IV55156.2024.10588666

Abstract— **Occlusion presents a significant challenge for safety-critical applications such as autonomous driving. Collaborative perception has recently attracted a large research interest thanks to the ability to enhance the perception of autonomous vehicles via deep information fusion with intelligent roadside units (RSU), thus minimizing the impact of occlusion. While significant advancement has been made, the data-hungry nature of these methods creates a major hurdle for their real-world deployment, particularly due to the need for annotated RSU data. Manually annotating the vast amount of RSU data required for training is prohibitively expensive, given the sheer number of intersections and the effort involved in annotating point clouds. We address this challenge by devising a label-efficient object detection method for RSU based on unsupervised object discovery. Our paper introduces two new modules: one for object discovery based on a spatial temporal aggregation of point clouds, and another for refinement. Furthermore, we demonstrate that fine-tuning on a small portion of annotated data allows our object discovery models to narrow the performance gap with, or even surpass, fully supervised models. Extensive experiments are carried out in simulated and real-world datasets to evaluate our method [†].**

I. INTRODUCTION

Autonomous vehicles (AVs) have the potential to transform transportation. To safely navigate their environment, AVs rely on LiDARs to detect other road users. While being able to produce accurate and dense range measurements, LiDARs are highly vulnerable to occlusion and sparsity, which result in low-density or absent measurements in certain areas. Such unobservable areas can be safety critical as other road users can emerge from these areas, thus risking collisions. The severity of this issue is quantified by the fact that 80% of collisions involving AVs in California, USA, occur at intersections where occlusion is the most severe [1].

One solution to such a challenge involves enhancing AVs perception through collaboration with intelligent roadside units (RSUs), advanced sensing systems positioned at elevated positions around intersections such that they have a minimally occluded field of view. Various collaborative perception methods [2], [3] share the common requirement for annotated data to train deep learning models. The multitude of intersections results in a large amount of data to be annotated. While human annotation provides the most accurate labels, manual labelling is laborious and costly. A more scalable method in terms of labelling effort is, therefore, needed.

Recent advances in unsupervised object discovery in outdoor point clouds [4]–[8] present a potential solution to this challenge. These methods first discover objects with a parametric hand-crafted model made of density-based clustering (e.g., [9], [10]) and tightest-box fitting [11]. The discovered objects are then utilized as labels for training a deep learning-based detection model according to the self-training process of [12]. The main drawback of these methods is their performance gap compared to fully supervised models primarily due to the low precision and recall of the hand-crafted model.

The low recall issue arises from the failure of density-based cluster detection, attributed to low point density and the disjointed nature of point clouds from different parts of the same objects. Our solution is to use multi-frame multi-scale object discovery is to (1) increase points density by point clouds aggregation using scene flow and, (2) apply clustering algorithms to point clouds at different scales, with smaller point cloud scaling better able to detect larger vehicles (e.g. trucks, busses).

The low precision is the result of poor estimation of the dimension and pose of bounding boxes produced by the tightest-box fitting approach [11] when clusters form incomplete L-shapes. Such incompleteness is caused by objects' partial visibility or the clustering algorithm's missing of objects' points. We resolve clusters' incompleteness by devising a new refinement method based on the aggregation of points on objects' trajectories.

Furthermore, our unsupervised object discovery is complemented with fine-tuning. We demonstrate that fine-tuning the model pre-trained using discovered objects and self-training on a small amount of manually labeled data can bridge the performance gap with respect to the fully supervised model.

In summary, our paper makes the following contributions:

- We present the first autolabeling framework for RSUs' point clouds to address the challenge of label-efficient object detection models for smart infrastructures.
- Our framework introduces two novel modules: (1) object discovery based on spatial and temporal aggregation of point clouds and at multiple scales, and (2) refinement of discovered objects.
- We demonstrate that with 100 manually-labeled point clouds, our method achieves performance compara-

[1]INRIA Centre at Université Côte d'Azur, Sophia Antipolis, France, minh-quan.dao@inria.fr
[2]Intelligent Vehicles Group, Delft University of Technology, the Netherlands
[3]Australian Centre for Robotics, University of Sydney, Australia,
[4]LS2N, École Centrale de Nantes,

This work has been carried out and funded in the framework of the ANNAPOLIS project managed by the National Agency for Research (ANR-21-CE22-0014)

[†]The code is released in `https://gitlab.inria.fr/mdao/label-efficient-rsu`

ble to models trained fully supervised on 8900 and 1920 manually-labeled point clouds from the synthetic V2X-Sim dataset [13] and the real-world dataset A9-Intersection [14], respectively, reaching 99% and 96% performance, respectively.

II. RELATED WORK

A. *Unsupervised Object Discovery*

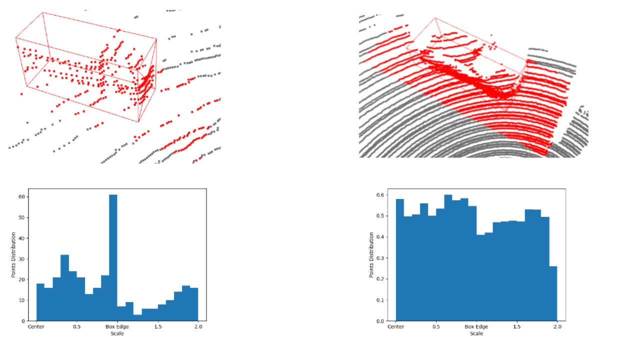
Unsupervised Object Discovery is a recent advance of the unsupervised object detection. Methods of this framework consists of a hand-crafted model for discovering (or localizing) objects in point clouds based on geometrical and statistical cues. The discovered objects then play the role of the initial labels set in the training of learning-based detection models following the self-training method of [12].

MODEST [4] is the first to demonstrate object discovery in automotive point clouds. Its object discovery model is based on the ephemeral score which is a statistic describing the consistency of the neighbourhood of a LiDAR point across different traversals. A concerntration of points with high ephemeral scores, detected by DBSCAN [9], is regarded as a dynamic object, which is then localized by the bounding box fitting algorithm of [11]. MODEST has three drawbacks including (1) its need for multiple traversals, (2) its restriction to discover only dynamic objects, and (3) its limited performance compared to fully supervised models.

MI-UP [6] resolves the multi-traversal requirement by detecting dynamic points using the scene flow. Furthermore, it improves the final performance by tracking discovered objects through time to obtain a better estimation of their dimension. While achieving strong performance MI-UP's reliance on dynamic points, like MODEST, confines its scope exclusively to discovering dynamic objects.

OYSTER [7] overcomes such restrictions on dynamic points by directly applying DBSCAN to non-ground points, although this approach carries the risk of introducing spurious clusters due to static points, primarily from the background. Its solution to spurious clusters is to restrict the discovery range (40 meters for 64-beam LiDARs) so that foreground objects (e.g., cars, pedestrians) have a high density of LiDAR points and complete appearances, thus facilitating density-based clustering. Furthermore, OYSTER employs tracking to filter objects belonging to short trajectories. The detection range is subsequently extended during the self-training stage, thanks to a data augmentation method that drops points of each label to simulate their appearance at a long range.

The rate dropping procedure of OYSTER is tailored to LiDAR mounted on the roof of vehicles, making it unsuitable to apply to RSUs. In contrast to prior works, we simultaneously resolve the restriction to dynamic objects and discovery range thanks to the aggregation of point clouds from multiple RSUs and multiple time-steps, offering dense and complete point clouds of the scene. We further develop a refinement module based on multi-object tracking and point clouds registration to obtain better estimation of objects' dimension and poses. Compared to the refinement of OYSTER which is also based



(a) Points' distribution resembles a Gaussian in the absence of the top face

(b) Points' distribution is uniform in the presence of the top face

Fig. 1. Distribution of points in the proximity of two detections with and without the top face, measured by *scale*, calculated according to [15].

on tracking, ours goes one step further to correct their poses by solving a least-square optimization. DRIFT [15] extends MODEST by improving its performance through a reward ranked fine-tuning. Particularly, DRIFT uses a reward function based on several heuristics for ranking an unsupervised model's detections and keeps only high ranked ones to use as labels during self-training. Its most important heuristic - *alignment reward*, which gives high scores to detections having points scattering near their side faces, is not suitable for RSUs' point clouds where objects' top faces are visible. As illustrated in Fig.1, points on objects' top face render the assumption behind the alignment reward that good detections have points distributed according to the Gaussian distribution invalid. Our method, taking a different approach, performs fine-tuning directly on manually-generated labels.

B. *Semi-Supervised Object Detection*

In parallel to our work, semi-supervised learning (SSL) [16]–[19] also addresses label-efficient 3D object detection. Assuming a part of the training set is labeled, these methods train a teacher model in a fully supervised manner, then use it to generate labels, referred to as *pseudo labels*, for the unlabelled data which are used for training the student model. These methods focus on improving the quality of pseudo labels Wang et al [17] address this by filtering low confidence and poorly localized pseudo labels using detection confidence scores and estimated Intersection-over-Union, respectively. [18] targets the recall rate of the teacher which they improve using test-time augmentation and an ensemble of models made of the teacher's weights obtained at difficulty epochs. [19] uses a dual-threshold strategy to filter low quality pseudo labels of difficulty classes. As our method is based on unsupervised object discovery, the availability of labeled data is optional.

MS3D [20] and its follow-up work [21] take a different setting where supervision comes as manually-generated labels of different domains (i.e., datasets). They use an ensemble of models trained on manually-labeled data, referred to as experts, to generate pseudo labels. Pseudo labels are then refined by tracking to exclude dynamic objects and obtain a better dimension estimation of static objects. While being similar in using tracking for discovered objects refinement, the role of tracking in our method is not to exclude dynamic objects but to jointly improve dimension and pose estimation.

III. Method

A. Problem Definition

We target the scenario where an intersection is covered by at least one RSU, each of which has at least one LiDAR. These RSUs are infrastructure; therefore, we assume they are well localized in a common frame of reference and obtain point clouds in synchronization. The input to our object discovery is the aggregation of point clouds from each of the RSUs, which is possible because the object discovery takes place offline using recorded data. A small part of the training set is manually annotated for hyperparameters tunning and model fine-tuning. The validation set, on the other hand, is manually annotated entirely.

B. Notations

A point cloud is a set of 3D points, denoted by $\mathcal{P} = \{\mathbf{p}_j = [x, y, z]\}$. A cluster, $\mathcal{C}$, is a subset of the point cloud $\mathcal{P}$ containing points from a single object. An object is localized by its bounding box $\mathbf{b} = [c_x, c_y, c_z, w, l, h, \theta, v_x, v_y]$ which is parameterized by the 3D coordinate of its center $[c_x, c_y, c_z]$, its dimension $[w, l, h]$, its yaw angle θ, and its velocity on the horizontal plane $[v_x, v_y]$. In the following, we use *object* to refer to both the cluster $\mathcal{C}$ of the foreground object and the bounding box $\mathbf{b}$ that encapsulates this cluster.

Our method, illustrated in Fig. 2, comprises four stages: multi-frame, multi-scale object discovery, refinement, self-training, and fine-tuning. We will present the details of each module in the following sub-sections.

C. Object Discovery

The typical object discovery pipeline consists of four sequential steps: ground removal, cluster detection, bounding box fitting, and filtering based on cluster dimensions [4]. The second step using density-based clustering algorithms such as DBSCAN result in sensitivity of this pipeline to point density, which is illustrated in Fig. 3.

We ensure a high point density by aggregating point clouds spatially from multiple RSUs and temporally from multiple timesteps. While the spatial aggregation is rather straightforward thanks to the static positions of RSUs, the temporal aggregation requires the treatment of noise in terms of tails from dynamic objects, which is caused by the change of objects' locations from one timestep to another. We resolve this issue by aligning point clouds using scene flow.

1) Scene Flow Estimation: We use the unsupervised method ICP-flow [22] for scene flow estimation. The method first detects clusters using HDBSCAN [23] in two ego-motion-compensated point clouds, $\mathcal{P}^t$ and $\mathcal{P}^{t'}$, then matches clusters found in $\mathcal{P}^t$ to those in $\mathcal{P}^{t'}$ by solving a linear assignment problem representing a cost matrix $\mathbf{M}$. An element at row i and column j of the cost matrix $\mathbf{M}$ represents the inlier ratio of registering cluster $\mathcal{C}_i^t$ found in $\mathcal{P}^t$ to cluster $\mathcal{C}_j^{t'}$ found in $\mathcal{P}^{t'}$ using ICP [24]. The transformation resulting from the registration of a pair of matched clusters is used to displace points in cluster $\mathcal{C}_i^t$ so that their scene flows are calculated as their displacement.

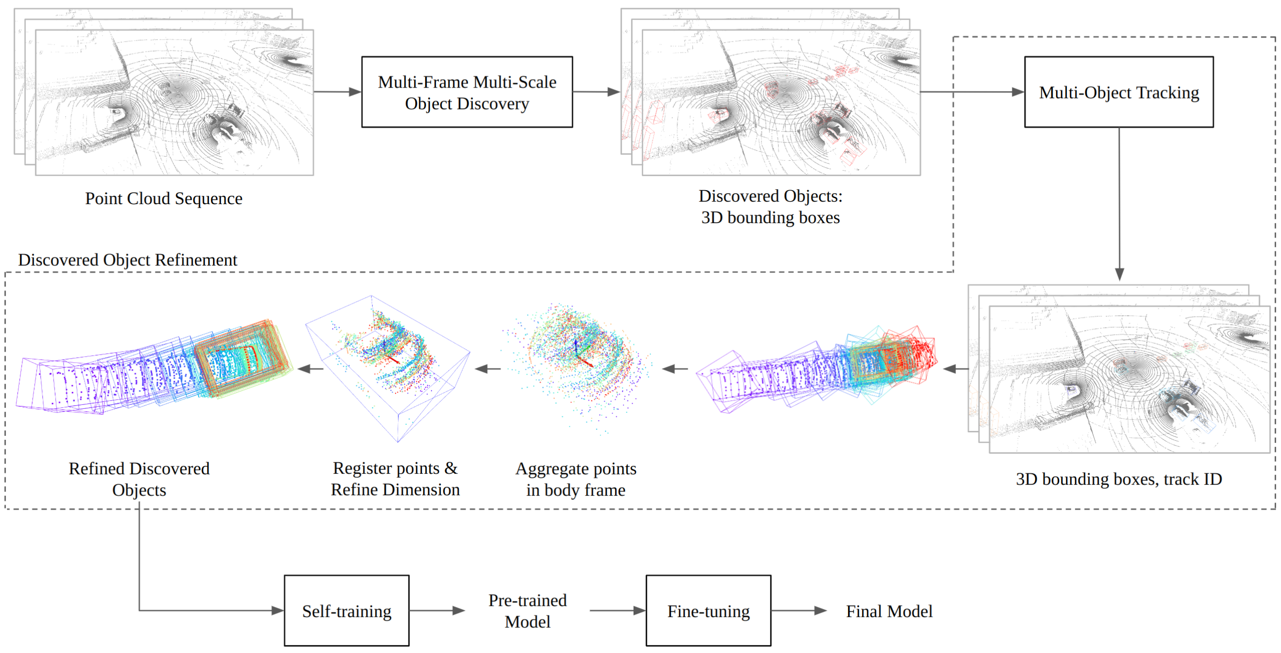
2) Object Discovery At Multiple Scales: We observe that a large vehicle (e.g., bus, truck) can exhibit large gaps between points due to the large object size and sparse lidar patterns, especially when it is at a long distance from the RSU. These gaps cause DBSCAN to segment its points into multiple small clusters, which would be then disregarded because of their small sizes. Our solution to this issue is to sequentially scaling the input point cloud with different factor from large to small and clustering on each scale to discover regular to large vehicles. The motivation is that clustering at a large scale picks up clusters of regular-sized vehicles. On the other hand, scaling the input point cloud with a small factor makes large groups of points smaller, thus easier to detect clusters of large vehicles. A failure mode of our scaling approach is when two large vehicles are near others. In this case, both of them are not detected on the large scales and are grouped together by DBSCAN on a small scale, resulting in an abnormally large vehicle. This large detection is removed by the dimension-based filtering step, thus causing two false negatives. Our multi-frame multi-scale object discovery is presented in Alg. 1.

Algorithm 1: Object Discovery

```
Input:
$\mathcal{P}^t$: the point cloud at the current timestep $t$
$\{\mathcal{P}^{t_i} | i = 1, ..., k\}$: $k$ point clouds at different timesteps
Output: $\mathcal{B} = \{\mathbf{b}_i\}$: discovered objects in $\mathcal{P}^t$
for $i = 1, ..., k$ do
    $\mathcal{S}^i = \text{scene\_flow}(\mathcal{P}^{t_i}, \mathcal{P}^t)$
    // with $\mathcal{S}^i := \{\mathbf{f}_j = [f_x, f_y, f_z]\}$
    $\mathcal{P}^{t_i} = \text{translate}(\mathcal{P}^{t_i}, \mathcal{S}^i)$
    $\mathcal{P}^t = \mathcal{P}^t \cup \mathcal{P}^{t_i}$
end
$\mathcal{P}^t = \text{remove\_ground}(\mathcal{P}^t)$
$\mathcal{B} = \emptyset$
for $s$ in list of scales do
    $\mathcal{P}_s = \text{scale}(\mathcal{P}^t, s)$
    $\{\mathcal{C}_i\} = \text{clustering}(\mathcal{P}_s)$
    for each cluster $\mathcal{C}_i$ do
        $\mathbf{b}_i = \text{box\_fitting}[\text{inverse\_scale}(\mathcal{C}_i, s)]$
        $\mathcal{P}^t = \mathcal{P}^t \setminus \text{inverse\_scale}(\mathcal{C}_i, s)$
        $\mathcal{B} = \mathcal{B} \cup \mathbf{b}_i$
    end
end
```

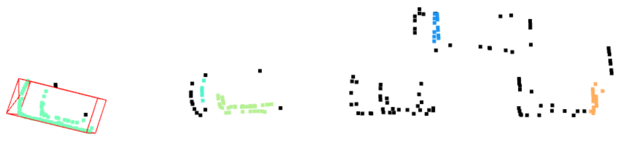
D. Objects Refinement

The cluster-driven nature of our object discovery method makes detection inherently challenging when objects are only partially visible. We leverage the intuition that an objects'

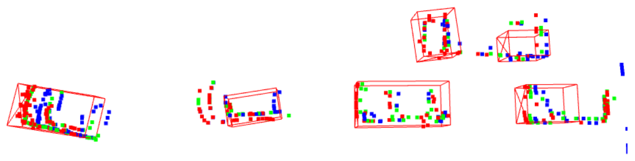


Point Cloud Sequence
Multi-Frame Multi-Scale Object Discovery
Discovered Objects: 3D bounding boxes
Multi-Object Tracking
Discovered Object Refinement
3D bounding boxes, track ID
Aggregate points in body frame
Register points & Refine Dimension
Refined Discovered Objects
Self-training
Pre-trained Model
Fine-tuning
Final Model

Fig. 2. Overview of our method



(a) Object discovery results using one point cloud. Points are colored according to their cluster index. Black points are outliers.



(b) Object discovery results using the aggregation of points at (t-1), t, and (t+1).

Fig. 3. Comparison of discovered objects using one point cloud, and the concatenation of three point clouds.

visibility can improve over time due to their motion, we devise a refinement module based on an objects' trajectory, which we refer to as a *tracklet*. Formally, a tracklet $\mathcal{T}$ is a sequence of bounding boxes, refered to as *instances*, representing the pose and dimension of an object at different timesteps when this object is observable.

$$\mathcal{T} = \left\{\mathbf{b}^{t_i} = [c_x, c_y, c_z, l, w, h, \theta, v_x, v_y] \,|i = 1, ..., k\right\} \quad (1)$$

Once a tracklet is formed, the dimension of the corresponding object and the refined pose of its instances are calculated following the procedure in Sec. III-D.2 and Sec. III-D.3.

1) Multi-Object Tracking: To form tracklets, we use a track-by-detection algorithm [25]. This algorithm extends a tracklet, formed at timestep $(t-1)$, to the current timestep t by finding the bounding box $\mathbf{b}_t$ that matches with the prediction of its state. The prediction of a tracklet's states is done using a constant velocity model. The matched bounding box is used to update the prediction of the tracklet's state according to the formulation of the Kalman filter.

2) Refining Dimension: We build a complete reconstruction of an object from its tracklet by first aggregating points $\left\{{}^{w}\mathbf{p}_j^{t_i}\right\}$ residing inside the bounding box of instance $\mathbf{b}^{t_i}$, then transforming aggregated points from the world frame $\mathcal{F}_w$ to the object's body frame $\mathcal{F}_o$, and finally aligning them using ICP.

$$\left\{{}^{o}\mathbf{p}_j^{t_i}\right\} = \text{ICP}\left(\text{rigid\_body\_transform}\left({}^{w}\mathbf{T}_o^{-1}, \left\{{}^{w}\mathbf{p}_j^{t_i}\right\}\right)\right) \quad (2)$$

here, ${}^{w}\mathbf{T}_o$ is the rigid body transformation that transforms points from the body frame $\mathcal{F}_o$ of the instance $\mathbf{b}^{t_i}$ to the world frame $\mathcal{F}_w$

$${}^{w}\mathbf{T}_o = \begin{bmatrix} \cos\theta^{t_i} & -\sin\theta^{t_i} & 0 & c_x^{t_i} \\ \sin\theta^{t_i} & \cos\theta^{t_i} & 0 & c_y^{t_i} \\ 0 & 0 & 1 & c_z^{t_i} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

A bounding box is fit to the resulting points to obtain the dimension of the object.

3) Updating Object Pose: We "align" the pose (i.e., position and orientation) of instance $\mathbf{b}^{t_i}$ by seeking the transformation ${}^{w}\mathbf{T}'_o$ that best explains the coordinates of the instance's points in the object frame $\mathcal{F}_o$ as computed by Eq. (2). We solve the following optimization problem

$${}^{w}\mathbf{T}'_o = \underset{{}^{w}\mathbf{T}_o}{\text{argmin}} \sum_i \left\| {}^{w}\mathbf{T}_o^{-1} \cdot \underline{{}^{w}\mathbf{p}_j^{t_i}} - \underline{{}^{o}\mathbf{p}_j^{t_i}} \right\| \quad (4)$$

here, $\underline{\mathbf{p}}$ denotes the homogeneous coordinate of the 3D point $\mathbf{p}$. The entire refinement pipeline is presented in Alg. 2.

Algorithm 2: Object Refinement

```
Input: $\mathcal{T} = \{(\mathbf{b}^{t_i}, \{{}^w\mathbf{p}_j^{t_i}\})\}$ // instances of a tracklet and their points
for each $t_i$ do
    $\{{}^o\mathbf{p}_j^{t_i}\}$ = rigid_body_transform $({}^w\mathbf{T}_o, \{{}^o\mathbf{p}_j^{t_i}\})$
end
$\{{}^o\mathbf{p}_j^{t_*}\}$ = points of the largest instance
$\mathcal{P} = \{{}^o\mathbf{p}_j^{t_*}\}$
for each $t_i \neq t_*$ do
    $\{{}^o\mathbf{p}_j^{t_i}\}$ = ICP $(\{{}^o\mathbf{p}_j^{t_i}\}, \mathcal{P})$   // Eq.(2)
    $\mathcal{P} = \mathcal{P} \cup \{{}^o\mathbf{p}_j^{t_i}\}$
end
$\mathbf{b}_*$ = box_fitting $(\mathcal{P})$   // refine dimension
for each $t_i$ do
    new pose = solve Eq. (4)
end
```

E. Self-Training

We use the discovered objects to start a self-training process, where a learning-based detection model iteratively improves its performance. In the first iteration, a detection model $\mathfrak{D}^0$ is trained from scratch with the discovered objects as labels. Once $\mathfrak{D}^0$ converges, the high-confidence detections from $\mathfrak{D}^0$ on point clouds from the training set act as labels in the second iteration, which is why they are referred to as *pseudo labels*. This process is repeated until the maximum number of iterations is reached or the performance saturates.

F. Fine-Tuning

The benefit of self-training is that the model is able to learn patterns associated with the presence of objects and thus is able to detect objects that are missed in the discovery phase. This extrapolation ability is at the cost of false positive detections. At any self-training round, the detection filtering based on confidence score can not prevent these false positives from propagating to the next round because some of them have relatively high confidence. As a result, the model consolidates its false belief about the appearance of foreground objects. The correction of this false belief requires human intervention which we carry out by fine-tuning the model trained in the self-training phase on the manually-labeled part of the training set.

Beside the straightforward implementation of fine-tuning that is to re-train the model from the pre-trained weights on new data, we introduce a scheme that mixs self-training and fine-tuning to obtain higher performance with the same amount of labeled data. Particularly, given a self-trained model, we repeat m times the following process (1) one-iteration self-training, and (2) fine-tuning. Our scheme is motivated by the observation that self-training increases the model's recall rate, resulting in more true positives and false positives. Fine-tuning helps remove those false positives, thus increasing the final precision.

IV. EXPERIMENTS

Datasets. We validate our method on two datasets: V2X-Sim 2.0 [13] and A9-Intersection, or A9 for short, [14]. V2X-Sim 2.0 is a synthetic dataset made with CARLA [26]. It contains 100 sequences, each of which has 100 samples containing point clouds obtained by 1 RSU positioned at the center of an intersection of a town of CARLA and up to 5 connected vehicles. Due to the absence of a dataset containing multiple RSUs, we use the connected vehicles in V2X-Sim to simulate the scenario where an intersection is covered by more than 1 RSUs. We put 8900 data samples collected at Town 04 and 05 to the training set and 1100 samples from Town 03 to the validation set. A9 is a real-world dataset made of two RSUs positioned at 7 meters height in an intersection in Garching, Germany. Each RSU has an Ouster OS1-64 LiDAR to obtain point clouds of the intersection at 10Hz and in synchronization with the other. A9 has 1920 samples for training and 240 samples for validation.

Evaluation metric. As we design our approach to detect vehicles, we merge the ground truth of every wheeled-vehicle class to a unified *vehicle* class. We use mean Average Precision (mAP) and Detection Score (NDS) of the nuScenes dataset [27] as the evaluation metrics because they correlate well with downstream driving tasks [28]. NDS is the weighted sum of mAP and positive metrics including translation error, scale error, orientation error, and velocity error, averaged over all true positive detections. In the following tables, the best and second best performance in each metric are shown in **bold** and underline, respectively.

Implementation. We demonstrate our approach using VoxelNext [29]. Nevertheless, our findings are applicable to other detection models. To facilitate reproducibility, our implementation is based on the open-source code from OpenPCDet [30]. We use the multi-object tracking implemented by [31]. The hyper-parameters of VoxelNext are kept identical to the default setting of OpenPCDet, with the exception of reducing the number of epochs to 10 and setting the total batch size to 12. Following the evaluation protocol of nuScenes for wheeled-vehicle class, the detection range is set to 50 meters from the center of the intersection. The number of self-training iterations is 3 on both datasets. The training takes place on a single NVIDIA A6000 GPU.

A. Object Discovery Results

We compare our object discovery method against two baselines: DBSCAN and a state-of-the-art object discovery method - MODEST [4]. Our implementation on V2X-Sim dataset uses 3 frames (i.e. point clouds); therefore, we also feed up to 3 frames to the two baselines. As MODEST is originally developed to discover objects using multiple traversals of the same route, we adapt this method to the setting of RSUs by reasoning that each RSU's point cloud is equivalent to a traversal because it is stationary. Similar to our method, the two baselines use the aggregation of point clouds from every available RSU as their input, thus having dense and more complete point clouds of the scene.

The comparison result in Tab. I shows that our method outperforms the best setting of DBSCAN of MODEST by 122.7% and 42.3% in terms of mAP. To highlight the ability of detecting more ground truth at high precision, we present the comparison of recall rate computed by matching discovered objects with ground truth at the Intersection-over-Union threshold 0.3. It is worth noticing that the two baselines and our discovery method do not produce confidence score for discovered objects;, therefore, the calculation of recall rate in Tab. I performs regardless of this quantity. The underperformance of MODEST compared to ours can be explained by its restriction to dynamic objects. As RSUs are positioned at intersections where only traffic in one direction is allowed to move, their environments contain less dynamic objects compared to those of vehicles which MODEST is designed for. Therefore, the exclusion of static objects limit MODEST's recall rate.

TABLE I

COMPARISON OF DIFFERENT OBJECT DISCOVERY METHODS ON THE VALIDATION SET OF V2X-SIM.

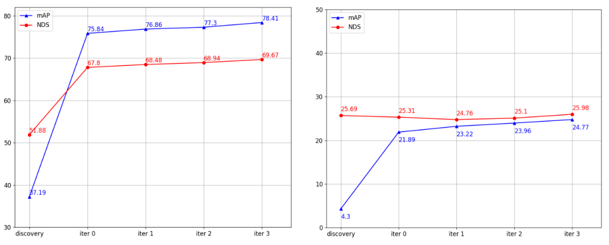
	mAP	NDS	Recall
DBSCAN - 1 Frame	12.46	27.03	29.98
DBSCAN - 2 Frames	14.30	40.28	38.47
DBSCAN - 3 Frames	16.70	36.94	44.34
MODEST - 2 traversals[1]	23.76	38.98	43.13
MODEST - 3 traversals[1]	26.14	40.41	45.87
Our object discovery	35.82	35.12	**70.71**
+ refinement	**37.19**	**51.88**	53.70

The importance of the refinement module is indicated by 47.7% improvement in NDS, from 35.12 to 51.88, when it is integrated into our object discovery module. This improvement is due to a 43.2%, 66.0%, and 99.4% reduction in translation error, scale error, and orientation error, respectively, averaged over all true positive detections. The refinement module also results in a 24% drop of recall rate, from 70.71 to 53.70. This is because of the filtering of tracklets having few instances, which are predominantly tracklets of spurious detections. While reducing the number false positives, this filtering process inevitably removes some true positives, especially when the tracking algorithm fails to make the association between tracklets and detections.

B. *Self-Training Results*

Fig. 4 shows the evolution of detection performance during the self-training process. The most substantial improvement, by 103.9% (from 37.19 to 75.84) on V2X-Sim and 409.1% (from 4.3 to 21.89) on A9 in terms of mAP, takes place at the first iteration where discovered objects are used as labels, showing the ability of capturing patterns in the training data of deep learning models. As the self-training progresses, the rate of improvement quickly reduces to roughly 1 mAP per iteration, showing that the set of pseudo labels is consistent.

[1]Our adaptation to the setting of RSU data



(a) V2X-Sim (b) A9

Fig. 4. The evolution of detection performance during self-training.

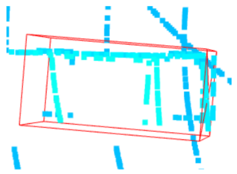
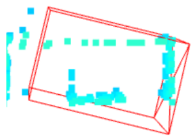
Unlike mAP, NDS shows a lower rate of improvement during our experiments in V2X-Sim dataset while being stagnated in A9 dataset. The breakdown of NDS indicates that all positive metrics remain unchanged throughout the self-training process in V2X-Sim dataset. Therefore, its improvement is solely due to the increase of mAP. The evolution of positive metrics in A9 dataset share the same pattern except for the orientation error. This error increases from 1 degree of discovered objects to 60 degrees in the first self-training iteration, then remains unchanged during subsequent iterations. This shows that the self-trained model, while successfully detecting the pattern that indicating the presence of objects in point clouds, encounters difficulty in learning to precisely predict their location, dimension and orientation,. This observation highlights the importance of precisely localizing objects in the discovery phase, which we achieve with the refinement module.

The comparison of our method against MS3D++ [21] on the V2X-Sim dataset is shown in Tab. II. Here, [21] uses an ensemble of two VoxelRCNN [32] trained fully supervised on the entire training set of nuScenes [27] and Lyft dataset, respectively. As the input to our object discovery consists of three frames, we also pass three frames to [21] for generating pseudo labels. Benifiting from large amount of manully-generated labels in nuScenes and Lyft, [21] achieves almost double the precision of ours in the object discovery phase. However, we surpass their performance after the self-training phase due to the domain gap between RSUs'data and autonomous driving data, caused by the different appearance of objects on RSUs' point clouds due to their elevated height and multi views.

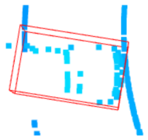
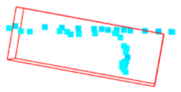
TABLE II

COMPARISON BETWEEN OUR SELF-TRAINED MODEL AND MS3D ON THE VALIDATION SET OF V2X-SIM.

	mAP	NDS
[21] - pseudo labels	68.41	79.34
[21] at iter 3	71.20	72.00
Ours - discovered objects	37.19	51.88
Ours at iter 3	78.41	69.67

(a) Visual comparison between a correctly detected truck (left) and falsely detected bus stop (right)

(b) Visual comparison between a correctly detected car (left) and falsely detected wall (right)

Fig. 5. False positive predictions of the self-trained model due to the similar appearance of vehicles and background objects.

C. Fine-Tuning Results

Sec. IV-B shows that self-training drastically improves the detection performance of the object discovery; however, the final self-trained model still underperforms the *fully supervised model*. Here, the the term "fully supervised model" refers to the model that is trained from randomly initialized weights on 100% of the manually labeled training set as the fully supervised model. This underperformance is due to (1) false positive detections caused by the appearance similarity between vehicles and background objects (e.g., wall and bus stops) illustrated in Fig. 5 and (2) the difficulty in learning to predict precisely objects' location, dimension, and orientation as discussed in Sec. IV-B.

Tab. III shows that fine-tuning the self-trained model can overcome these two issues as an increase in the volume of manually labeled data available for fine-tuning leads to an increase in the model's performance. With 20% and 50% of manually labeled data of the training set of V2X-Sim and A9, the fine-tuned model can reach the performance of the fully supervised one. Moreover, the comparison between fine-tuning the self-trained model and the one trained from randomly initialized weights given the same amount of manually labeled data shows a favorable result to fine-tuning, emphasizing the importance of object discovery and self-training especially when labeled data is scarce.

The comparison between the traditional fine-tuning and our scheme of mixing self-training and fine-tuning is presented in Tab. IV. In this experiment, the amount of lableled data for both method is 100 point clouds which accounts for 1% and 5% of the training set of V2X-Sim and A9, respectively. The number of times that self-training followed by fine-tuning is repeated, m, is 2. Our mixed fine-tuning achieves higher performance than traditional fine-tuning given the same amount of labeled data and yields 99% and 96% performance of the fully supervised baseline, further proving the data-efficiency of our method.

TABLE III

COMPARISON OF NDS BETWEEN FINE-TUNING AND TRAINING FROM RANDOMLY INITIALIZED WEIGHTS

Labeled data	V2X-Sim		A9	
(% of training set)	Fine-tune	Train from random init.	Fine-tune	Train from random init.
0	69.67	0.00	25.98	0.00
1	78.51	36.54	36.83	0.00
2	80.79	57.79	46.46	0.00
5	85.14	77.77	53.98	7.75
10	88.32	84.77	64.04	15.75
20	89.33	89.75	69.46	58.58
50	<u>91.03</u>	90.33	<u>72.14</u>	71.05
100	**93.42**	91.00	**75.68**	71.40

TABLE IV

COMPARISON BETWEEN THE TRADITIONAL FINE-TUNING AND MIXED FINE-TUNING & SELF-TRAINING.

Dataset	Method	Labeled data (% training set)	mAP	NDS
V2X-Sim	Fine-tuning	1%	87.71	78.51
	Mixed self-training & fine-tuning	1%	91.66	83.48
	Fully supervised	100%	93.00	91.00
A9	Fine-tuning	5%	54.59	53.99
	Mixed self-training & fine-tuning	5%	67.10	67.81
	Fully supervised	100%	69.64	71.40

D. Real-World Experiments

The dataset for real-world testing was collected using a RSU located at an urban intersection in Sydney, Australia. It has two LiDAR sensors: an Ouster OS1 64 beams and an Ouster Dome. The LiDARs are synchronized and phase-locked. The data for this qualitative evaluation was obtained during standard operation in moderate traffic conditions. Since there is no pre-existing annotated dataset for this specific setup, this paper presents these findings as qualitative results in Fig.6, showcasing the practical applicability of our method with real-world data. An extended visualization of our object discovery and refinement module can be found in our video demonstration.

V. Conclusion

We propose the first autolabeling framework for RSUs' point clouds based on unsupervised object discovery. In comparison to existing works, our framework comprises two novel modules: one for discovering objects at multiple scales through spatial and temporal aggregation of point clouds, and another for refining the discovered objects. With only 100 manually-labeled point clouds provided for fine-tuning, the model pretrained with labels generated by our framework achieves a performance comparable to that of the model trained fully supervised on thousands of manually-labeled point clouds. This results in a label-efficient object detection method for RSUs.

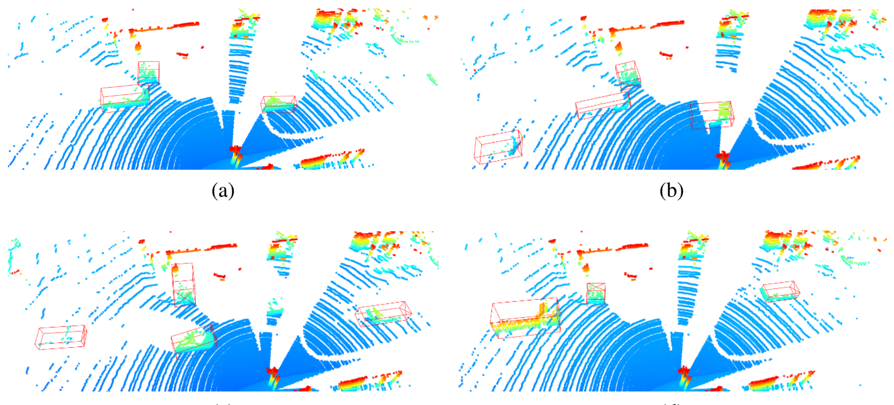

(a) (b) (c) (d)

Fig. 6. Visualization of discovered objects in our real-world data

References

[1] Y. Song, M. V. Chitturi, and D. A. Noyce, "Automated vehicle crash sequences: Patterns and potential uses in safety testing," *Accident Analysis & Prevention*, vol. 153, p. 106017, 2021.

[2] Y. Li, S. Ren, P. Wu, S. Chen, C. Feng, and W. Zhang, "Learning distilled collaboration graph for multi-agent perception," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[3] R. Xu, H. Xiang, Z. Tu, X. Xia, M.-H. Yang, and J. Ma, "V2x-vit: Vehicle-to-everything cooperative perception with vision transformer," in *European conference on computer vision*. Springer, 2022.

[4] Y. You, K. Luo, C. P. Phoo, W.-L. Chao, W. Sun, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Learning to detect mobile objects from lidar scans without labels," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[5] Y. Wang, Y. Chen, and Z.-X. ZHANG, "4d unsupervised object discovery," *Advances in Neural Information Processing Systems*, 2022.

[6] M. Najibi, J. Ji, Y. Zhou, C. R. Qi, X. Yan, S. Ettinger, and D. Anguelov, "Motion inspired unsupervised perception and prediction in autonomous driving," in *European Conference on Computer Vision*. Springer, 2022, pp. 424–443.

[7] L. Zhang, A. J. Yang, Y. Xiong, S. Casas, B. Yang, M. Ren, and R. Urtasun, "Towards unsupervised object detection from lidar point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 9317–9328.

[8] M. Najibi, J. Ji, Y. Zhou, C. R. Qi, X. Yan, S. Ettinger, and D. Anguelov, "Unsupervised 3d perception with 2d vision-language distillation for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 8602–8612.

[9] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[10] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering." *J. Open Source Softw.*, vol. 2, no. 11, p. 205, 2017.

[11] X. Zhang, W. Xu, C. Dong, and J. M. Dolan, "Efficient l-shape fitting for vehicle detection using laser scanners," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 54–59.

[12] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 687–10 698.

[13] Y. Li, D. Ma, Z. An, Z. Wang, Y. Zhong, S. Chen, and C. Feng, "V2x-sim: Multi-agent collaborative perception dataset and benchmark for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 914–10 921, 2022.

[14] W. Zimmer, C. Creß, H. T. Nguyen, and A. C. Knoll, "A9 intersection dataset: All you need for urban 3d camera-lidar roadside perception," *arXiv preprint arXiv:2306.09266*, 2023.

[15] K. Z. Luo, Z. Liu, X. Chen, Y. You, S. Benaim, C. P. Phoo, M. Campbell, W. Sun, B. Hariharan, and K. Q. Weinberger, "Reward finetuning for faster and more accurate unsupervised object discovery," *arXiv preprint arXiv:2310.19080*, 2023.

[16] N. Zhao, T.-S. Chua, and G. H. Lee, "Sess: Self-ensembling semi-supervised 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[17] H. Wang, Y. Cong, O. Litany, Y. Gao, and L. J. Guibas, "3dioumatch: Leveraging iou prediction for semi-supervised 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 615–14 624.

[18] J. Yin, J. Fang, D. Zhou, L. Zhang, C.-Z. Xu, J. Shen, and W. Wang, "Semi-supervised 3d object detection with proficient teachers," in *European Conference on Computer Vision*. Springer, 2022.

[19] C. Liu, C. Gao, F. Liu, P. Li, D. Meng, and X. Gao, "Hierarchical supervision and shuffle data augmentation for 3d semi-supervised object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 23 819–23 828.

[20] D. Tsai, J. S. Berrio, M. Shan, E. Nebot, and S. Worrall, "Ms3d: Leveraging multiple detectors for unsupervised domain adaptation in 3d object detection," *arXiv preprint arXiv:2304.02431*, 2023.

[21] ——, "Ms3d++: Ensemble of experts for multi-source unsupervised domain adaption in 3d object detection," *arXiv preprint arXiv:2308.05988*, 2023.

[22] Y. Lin and H. Caesar, "Icp-flow: Lidar scene flow estimation with icp," *arXiv preprint arXiv:2402.17351*, 2024.

[23] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 2013, pp. 160–172.

[24] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606.

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

[26] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[27] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[28] T. Schreier, K. Renz, A. Geiger, and K. Chitta, "On offline evaluation of 3d object detection for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[29] Y. Chen, J. Liu, X. Zhang, X. Qi, and J. Jia, "Voxelnext: Fully sparse voxelnet for 3d object detection and tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21 674–21 683.

[30] O. D. Team, "Openpcdet: An open-source toolbox for 3d object detection from point clouds," https://github.com/open-mmlab/OpenPCDet.

[31] Q. Wang, Y. Chen, Z. Pang, N. Wang, and Z. Zhang, "Immortal tracker," https://github.com/ImmortalTracker/ImmortalTracker, 2020.

[32] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li, "Voxel r-cnn: Towards high performance voxel-based 3d object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 2, 2021, pp. 1201–1209.