Towards Unsupervised Object Detection from LiDAR Point Clouds

Lunjun Zhang **Anqi Joyce Yang** **Yuwen Xiong** **Sergio Casas**
Bin Yang[†] **Mengye Ren**[†] **Raquel Urtasun**
Waabi, University of Toronto
{lzhang, jyang, yxiong, sergio, urtasun}@waabi.ai

Abstract

In this paper, we study the problem of unsupervised object detection from 3D point clouds in self-driving scenes. We present a simple yet effective method that exploits (i) point clustering in near-range areas where the point clouds are dense, (ii) temporal consistency to filter out noisy unsupervised detections, (iii) translation equivariance of CNNs to extend the auto-labels to long range, and (iv) self-supervision for improving on its own. Our approach, ***OYSTER*** *(**O**bject Discover**y** via **S**patio-**Te**mporal **R**efinement), does not impose constraints on data collection (such as repeated traversals of the same location), is able to detect objects in a zero-shot manner without supervised fine-tuning (even in sparse, distant regions), and continues to self-improve given more rounds of iterative self-training. To better measure model performance in self-driving scenarios, we propose a new planning-centric perception metric based on distance-to-collision. We demonstrate that our unsupervised object detector significantly outperforms unsupervised baselines on PandaSet and Argoverse 2 Sensor dataset, showing promise that self-supervision combined with object priors can enable object discovery in the wild. For more information, visit the project website:* https://waabi.ai/research/oyster.

1. Introduction

When a large set of annotations are available, supervised learning can solve the task of 3D object detection remarkably well thanks to the power of neural networks, as proven by many successful object detectors developed in the past decade [24, 36, 47, 70]. However, since most existing data is unlabeled, human annotations are currently the bottleneck for data-driven learning algorithms, as they require tedious manual effort that is very costly in practice. While there has been some effort on using weaker supervision to train object detectors [4, 57, 68], it is worth noting that human and animal brains are able to perceive objects without explicit labels at all [2]. This naturally inspires us to ask whether we can design unsupervised learning algorithms that discover objects from raw streams of sensor data on their own.

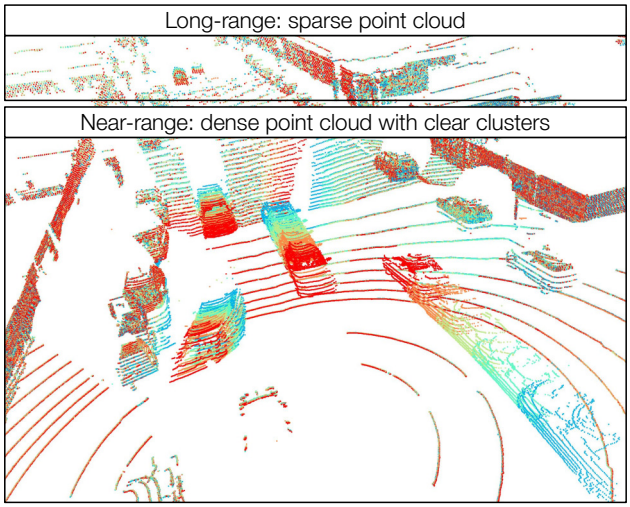

Figure 1. Visualization of a sequence of five 3D point clouds. At near range, the point clouds are very dense and we can clearly distinguish clusters, motivating us to use prior knowledge. At far range, the point clouds are quite sparse and the objects do not appear as obvious, leading us to explore zero-shot generalization.

Unsupervised object detection has long been studied in computer vision, albeit in various forms. For instance, numerous ways of unsupervised object proposals were considered as the first stage of an object detector [20]. These methods leverage a variety of cues including colors and edge boundaries [1], graph structures [18], and motion cues [50]. While those methods are no longer popular in today's object detectors due to end-to-end supervised training, they offer important intuitions of *what an object is*.

In recent years, unsupervised object detection has made a comeback under the name of *object-centric models* [6, 15, 33, 34, 38, 39, 58]. The essential idea is to train an auto-encoder with a structured decoder such that the network is forced to decompose the scene into a set of individual objects during reconstruction. Most of those models only show experiments on synthetic toy datasets, where the back-

[†]Work done at Waabi. Mengye is now at New York University.

ground lacks any fine-grained details, and the foreground objects have simplistic shapes and distinct colors. The reason why object-centric models have struggled to scale to realistic data is that their mechanism of object decomposition is based on careful balancing of model capacities between the foreground and background modules. Consequently, an increase in model capacity, which is needed for real-world data, breaks the brittle balance between modules and result in a failure in scene decomposition. Unsupervised object discovery in the wild remains an open challenge.

In this work, we study unsupervised object detection from point clouds in the context of self-driving vehicles (SDVs). This is a challenging task due to occlusion as well as the sparsity of the observations particularly at range. We refer the reader to Fig. 1 for an example. Despite its appeal, unsupervised object detection from LiDAR has received little attention. Recent work [66] exploits repeated traversals of the same region to understand the persistence of a point over time and discover mobile objects from that information. However, the assumption of repeated traversals in acquiring point clouds restricts the applicability of the method. Instead, we study the most generic setting of unsupervised detection given any raw sequence of point clouds.

Our method ***OYSTER*** (***O**bject Discover**y** via **S**patio-**T**emporal **R**efinement*) carefully combines key ideas from density-based spatial clustering, temporal consistency, equivariance and self-supervised learning in a unified framework that exploits their strengths while overcoming their shortcomings. Firstly, we exploit point clustering to obtain initial pseudo-labels to bootstrap an object detector in the near range, where point density is high (see Fig. 1). We then employ unsupervised tracking to filter out temporally inconsistent objects. Since point clustering does not work well in the long-range where observations are sparse, we exploit the translation equivariance of CNNs to train on high-quality, near-range pseudo-labels and zero-shot generalize to long range. To bridge the density gap between short and long-range at training vs. inference, we propose a novel random LiDAR ray dropping strategy. Finally, we design a self-improvement loop in which this bootstrapped model can self-train. At every round of self-improvement, we utilize the temporal consistency of objects to automatically refine the detections from the model at the previous iteration, and use these refined outputs as pseudo-labels for training. Our experiments on Pandaset [63] and Argoverse V2 Sensor [60] demonstrate that *OYSTER* clearly outperforms other unsupervised methods, both under standard metrics based on intersection-over-union (IoU) and our proposed metric based on *distance-to-collision* (DTC). We hope that our work serves as a step towards building perception systems that automatically improve with more data and compute without being bottlenecked by human supervision.

2. Related Work

In this section, we briefly review the literature on three subfields that are closely related to unsupervised object detection: object-centric models, unsupervised object proposal (discovery), and open-set detection.

Object centric models: A promising path towards unsupervised detection is broadly characterized as *vision as inverse graphics*: learning deep generative models with structured decoder (or renderer) such that the encoder needs to decompose a scene into objects and parts and infer their orientations. In the deep learning era, this type of models are usually called object-centric models. In the single-image setting, many prior works have tried to use a mixture latent to encourage object discovery, such as AIR [16], Neural EM [22, 53], SCAE [34], SPAIR [10], MONet [6], IODINE [21], GENESIS [15], SPACE [38], Generative Neuro-Symbolic machine [28], and Slot Attention [39]. Some effort has also been made in the setting of 3D view rendering, especially in light of recent progress in neural rendering models such as NeRF [40]: ObSuRF [51] and Object Radiance Fields [67] try to combine Slot Attention with NeRF to discover objects given multi-view inputs. Certain object-centric models have attempted to take advantage of temporal and sequential information in videos, such as Sequential AIR [33], SPAIR with tracking [11], SCALOR [29], physics as inverse graphs [27], SIMONe [31], and Flow Capsule [48]. Those methods have only been shown to work on toy datasets.

Unsupervised Object Discovery: Many methods directly use various cues to discover objects without necessarily using a generative model. Prior works have explored Probabilistic Latent Semantic Analysis [49], object co-segmentation [54], and a ranking based approach [55, 56]. For 2D object segmentation and object proposals, existing approaches include parametric min cut [8], selective search [52], and MCG [46]. When we have access to videos rather than just static images, motion information can be leveraged for object discovery, by utilizing motion segmentation [44], clustering the flow field [43], directly segmenting moving object for detection [19], or tracking and then segmenting [62]. In the case where we have 3D point clouds rather than an RGB image, prior techniques include point clustering [14], shape analysis [32] and range-based analysis [5]. Concurrently, [42] leveraged scene flow estimation to generate seed labels for training an unsupervised LiDAR detector. Our method initially applies point clustering, but our core contribution is a learning-based method that can iteratively self-improve. In contrast to our approach, [42] relies on unsupervised scene-flow to detect objects and thus is not able to detect static ones. Furthermore, it is limited to short ranges where point cloud alignment methods [3] are reliable. [66] does not need scene flow, but requires repeated

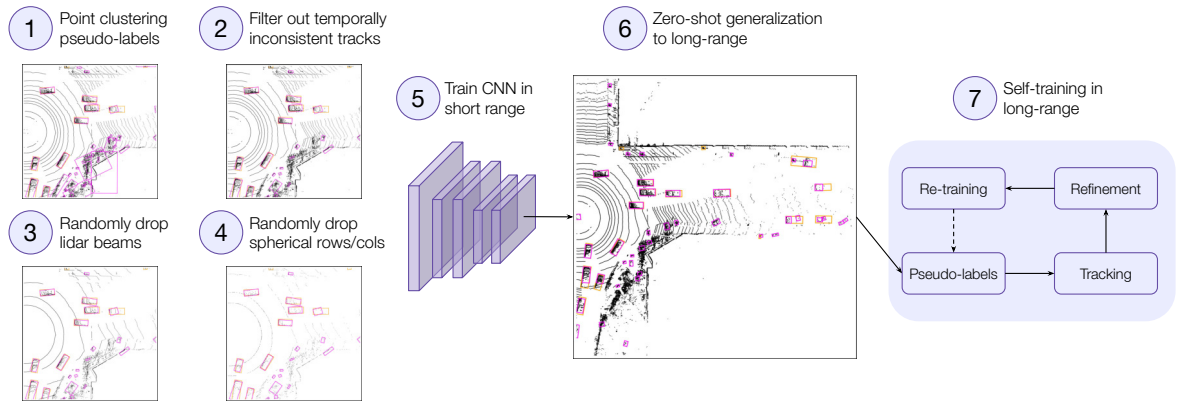

Figure 2. **Overview of *OYSTER*, our unsupervised detection method**. In the initial bootstrapping phase (step 1-6), we train a CNN on near-range point clustering results, and rely on translation equivalence of CNNs to produce pseudo-labels for full range. Our training applies random ray-dropping as data augmentation, and uses temporal-consistency based filtering on pseudo-labels. In the self-improvement phase (step 7), we propose a *Track, Refine, Retrain, and Repeat* framework that teaches an unsupervised detector to iteratively self-improve.

traversals of the same location. Concurrent work [59] requires unsupervised scene flow and camera inputs for 3D instance segmentation. Our method requires neither scene flow, camera inputs, nor repeated traversals, so it can be directly applied to any LiDAR sequence.

Open-Set Detection: Another branch of methods deal with a weakly supervised setting where the system is given labels on certain known objects but is expected to discover the unknown stuff on its own [12]. For 2D object detection, prior art includes Bayesian approaches [45] and dropout sampling for uncertainty estimation [41]. A new setting called Open-World Object Detection has been proposed as well [30], where the vision system sequentially receives new labeled instances on long-tailed classes [30]. A Detection Transformer [7] for the Open-World setting has also been studied [23]. For 3D LiDAR object detection, prior methods include OSIS [61] and open-set 3d-net [9]; however, they do not study a completely unsupervised setting.

3. Method: *OYSTER*

Our method has two phases of training: the **initial bootstrapping** phase, and the **self-improvement** phase.

The initial bootstrapping phase takes advantage of the fact that point clouds in the near range tend to be dense and have clear object clusters, so we can obtain reasonable near-range bounding box seed pseudo-labels via point clustering. Thanks to the translation equivariance property of convolutional nets, we find that a CNN detector trained on near-range labels can generalize to longer-range in a zero-shot manner with the help of data augmentations such as ray dropping that randomly sparsify the inputs.

The self-improvement phase utilizes temporal consistency of object tracks as a self-supervision signal. Given noisy detections across time, we employ an unsupervised offline tracker to find object tracks of various lengths. We discard short tracks, and refine long tracks. An object track should have the same object size across time, so our refinement process uses track-level information to update pseudo-labels in long tracks. We train a new detector on the updated pseudo-labels, dump its outputs as new pseudo-labels, track, refine, and repeat. Fig. 2 depicts the overall pipeline.

We describe the two training phases in more detail below.

3.1. Initial Bootstrapping via Point Clustering and Range Extension

Given a 3-dimensional LiDAR point cloud $p \in \mathbb{R}^{N\times 3}$ we obtain the initial seed pseudo-labels $\mathbf{B}^{(0)}$, via ground removal, point clustering, and bounding box fitting:

$$\mathbf{B}^{(0)} \leftarrow \text{FitBboxes}(\text{Clustering}(\text{RemoveGround}(p)))$$

where we employ the 2D Bird-Eye-View (BEV) detection representation [35, 65] and each BEV bounding box $\mathbf{b} = (x, y, l, w, \theta) \in \mathbf{B}^{(0)}$ consists of the centroid position (x, y), the length and width (l, w), and the heading θ. We perform this for each frame in the dataset $\mathcal{D}$ to obtain the full set of pseudo-labels $\mathcal{B}^{(0)}$. This process follows a classical pipeline prior to the deep-learning era [13,26]. We make simple choices for each module: for ground removal, we fit a linear ground plane similar to [66] and remove the estimated ground points before clustering; for clustering, we use DBSCAN [17]; for bounding box fitting of each point cluster, we use an off-the-shelf algorithm [69]. While much more sophisticated choices exist, those noisy initial labels are already good enough to kickstart our learning process.

In the near range (typically within 40m for a LiDAR with 64 beams), objects usually have clear point clusters (see Fig. 1), so the clustering labels suffice the bootstrapping purpose. However, in the long range, each object has much fewer points, making clustering results unreliable, so we should rely on neural net's ability to generalize even for initial label generation. Consequently, our goal is to *train a detector on near-range only, such that it can zero-shot*

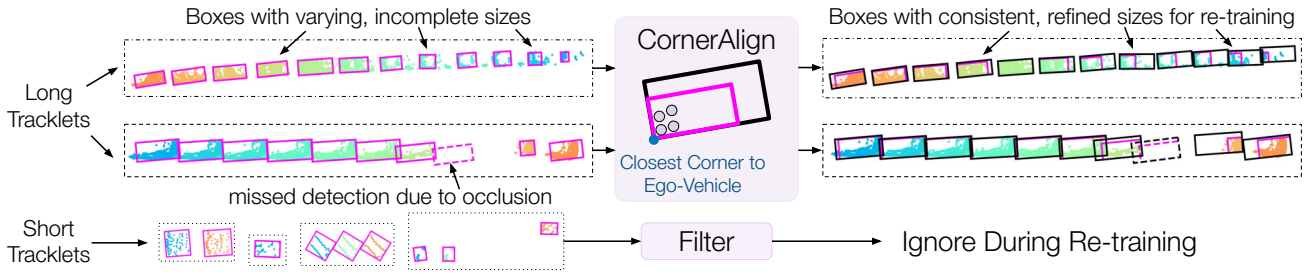

Figure 3. **Label Refinement Illustration**. With unsupervised tracking, we fill in missed detections and obtain per-object tracklets. For long tracklets, we leverage temporal observations to find a new, consistent object size and update the initial labels (in purple) with a corner-based alignment strategy [64]. We additionally filter out short tracklet labels, which will not be used in the next round of self-training.

generlize to longer range at test time. We utilize equivariance of convolutional nets to achieve this goal. Following [35, 65], we train a single-stage CNN detector on near-range only, where the inputs to the CNN are the voxelized point clouds. The voxelizer produces a binary voxel occupancy map $I \in \{0,1\}^{L\times W\times D}$, where $L \times W$ defines the input BEV grid resolution, and D is the number of channels in the height dimension. We use a ResNet [25] backbone with FPN [37] to produce 4x downsampled feature map, from which a simple convolutional header decodes the box parameters (x, y, l, w, θ) and the confidence c in a dense fashion (*i.e.*, per BEV pixel). After this, the set of object detections is obtained by applying confidence thresholding and non-maximum suppression (NMS). During training, the inputs to the backbone are near-range LiDAR BEV images of resolution $L \times W \times D$; during label generation (*i.e.*, at inference), the inputs are longer-range with size $\tilde{L}\times\tilde{W}\times D$, where $\tilde{L} > L$ and $\tilde{W} > W$.

Considering the difference in point density between near-range and long-range, we also propose to randomly drop rays during training, such that the convolutional kernels perform better on sparse regions, which is useful for the zero-shot generalization to long-range. This ray-dropping data augmentation is implemented by first randomly dropping LiDAR beams, and then randomly dropping points within evenly spaced rows and columns under a range-view image in spherical coordinates. This process aims to mimic how LiDAR beams tend to be more spaced out as range increases. We find that ray dropping helps CNNs generalize to longer range significantly better when only trained on near range, whose results outperform those from CNNs directly trained on full-range clustering labels.

$$\text{CNN}^{(0)} \leftarrow \text{Train}\Big(\Big\{\text{RayDrop}(p_i)\Big\}_{i\in\mathcal{D}}, \mathcal{B}^{(0)} \mid \text{near-range}\Big)$$

$$\mathcal{B}^{(1)} \leftarrow \Big\{\text{NMS}\Big(\text{CNN}^{(0)}(\text{Voxelize}(p_i)) \mid \text{full-range}\Big)\Big\}_{i\in\mathcal{D}}$$

The detection outputs above a certain confidence threshold will become our first iteration of self-training pseudo-labels $\mathcal{B}^{(1)}$, which are already able to discover many missed detections and remove false positives from initial point clusters $\mathcal{B}^{(0)}$. This self-correction phenomenon is also observed in [66], and is a result of the model's limited capacity to overfit to the inconsistent noises in the pseudo-labels.

3.2. Track, Refine, Retrain, and Repeat: Teaching Unsupervised Detectors to Self-Improve

Albeit better than the initial labels $\mathcal{B}^{(0)}$, the labels $\mathcal{B}^{(1)}$ after the initial training and range extension are far from perfect and require further refinement. We propose a framework *Track, Refine, Retrain, and Repeat* to teach unsupervised detectors to self-improve. Our framework consists of the following steps: run unsupervised tracking, refine the long tracks, discard the short tracks, re-train the detector on refined pseudo-labels, and repeat the process with thresholded detector outputs as the next round of pseudo-labels. The method uses temporal consistency as self-supervision to improve its own pseudo-labels and detection results. We will first describe our unsupervised tracker, and then discuss how we leverage the temporal information stored in tracklet states to refine and iterate.

Unsupervised Tracking: We follow the tracking-by-detection paradigm, where object detections are first obtained independently for each frame in a temporal sequence, and then discrete association between two consecutive time steps is performed iteratively to form the tracks. To do so, we employ a simple parametric online tracker. At each time step t in a sequence of frames, each object j has a tracklet $\mathbf{s}_t^j$ that stores the estimated state trajectories $\mathbf{S}_t = \{\mathbf{s}_t^j = (\mathbf{b}, v_x, v_y, c)_t^j, n_t^j\}$ where $\mathbf{b}$ corresponds to the previously introduced box parameters, (v_x, v_y) is the 2D velocity, n_t is the length of tracklet so far, and c is the confidence score of the tracklet. Given pseudo-labels $\mathcal{B}^{(k)}$ from self-training iteration k, we first forecast the new states of each tracklet and then match detections to predictions:

$$\mathbf{M}_t \leftarrow \text{Match}(\mathbf{B}_t^{(k)}, \text{Forecast}(\mathbf{S}_{t-1}))$$

We adopt the simplest strategy for forecasting and matching: forecasting assumes constant velocity between frames, and matching is greedy based on bounding box centroid distances. For a pseudo-label matched to tracklet j with tracklet length n_t^j so far, we set $m_t = n_t^j$, where m_t denotes the *temporal consistency score* of the pseudo-label;

if a pseudo-label is unmatched, we set $m_t = 0$. Meanwhile, the online tracker computes its current state $\mathbf{S}_t$ based on $\mathbf{S}_{t-1}$ and $\mathbf{M}_t$, which involves adding detections to their matched tracklets, growing unmatched tracklets (to handle occlusions and flickering detections), initializing new tracklets for unmatched detections, deleting tracklets that have been unmatched for too long, and incrementing each tracklet's length n_t^j. Finally, we simply run this tracker in both temporal directions, obtaining bi-directional tracklets $\overrightarrow{\mathbf{S}}_t$ and $\overleftarrow{\mathbf{S}}_t$. Note that each pseudo-label has also stored bi-directional temporal consistency scores $\overrightarrow{m}_t^j$ and $\overleftarrow{m}_t^j$, and we aggregate them with $\tilde{m}_t^{(k)} = \max(\overrightarrow{m}_t^{(k)}, \overleftarrow{m}_t^{(k)})$.

Pseudo-Label Refinement: Based on the temporal consistency scores $\tilde{m}_t^{(k)}$, we divide pseudo-labels into short vs. long tracklets by a threshold q. For the short tracklets with $\tilde{m}_t^{(k)} < q$, we ignore them in the next round of re-training. For the long tracklets with $\tilde{m}_t^{(k)} \geq q$, we refine them in a simple process. Due to the tightest-box-fitting nature of the seed clustering labels, the bbox sizes in $\mathcal{B}^{(k)}$ are smaller for partially observed objects, especially for those farther away from the ego-vehicle. To address this, for each actor trajectory, we set the new length and width of the actor as the top r percentile of all lengths and all widths detected throughout the tracklet. Next, following the corner-based alignment strategy from [64], we find the bbox corner that is the closest to the ego-vehicle, with the intuition that this corner is the most observed and likely to be the most reliable. We then update the bbox center with the anchored corner, the original bbox heading and the new bbox size. Fig. 3 illustrates the refinement process.

Iterative Self-Training: Following the refinement process, we obtain the refined pseudo-labels $\tilde{\mathbf{B}}_t^{(k)}$ and use them to guide the next round of training:

$$\tilde{\mathbf{B}}_t^{(k)} = \begin{cases} \text{Refine}(\overrightarrow{\mathbf{S}})_t^{(k)} & \tilde{m}_t^{(k)} \geq q \\ \emptyset & \text{otherwise} \end{cases}$$

$$\text{CNN}^{(k)} \leftarrow \text{Train}\Big(\Big\{\text{RayDrop}(p_i)\Big\}_{i\in\mathcal{D}}, \tilde{\mathcal{B}}^{(k)} \mid \text{full-range}\Big)$$

$$\mathcal{B}^{(k+1)} \leftarrow \Big\{\text{NMS}\Big(\text{CNN}^{(k)}(\text{Voxelize}(p_i)) \mid \text{full-range}\Big)\Big\}_{i\in\mathcal{D}}$$

where $\emptyset$ indicates that training loss is not applied at the location of this pseudo-object during detector re-training, *i.e.*, we only re-train on pseudo-labels that are temporally consistent and refined. With the re-trained detector, we derive the new generation of pseudo-labels $\mathcal{B}^{(k+1)}$, which can be further refined and used for re-training.

In summary, our *Track, Refine, Retrain, and Repeat* framework constructs an object discovery loop where the detector is iteratively re-trained on pseudo-labels of increasingly higher quality as self-training goes on.

4. Experimental Evaluation

In this section, we first outline the datasets and metrics we use, including a newly proposed detection metric based on distance-to-collision. Next, we show that our method outperforms state-of-the-art unsupervised detection methods. Finally, we present quantitative and qualitative insights into our contributions with a thorough ablation study of different components.

Datasets and Experiment Setting: We use Pandaset [63] and Argoverse 2 Sensor [60] (AV2 for short) to evaluate our method on two different sensor suites. Pandaset consists of 103 snippets of 8 seconds each, recorded in dense urban traffic in San Francisco as well as the El Camino Real highway. For our experiments, we use the spinning Pandar64 LiDAR. It features 28 different annotation classes, which we group into 3 main categories: vehicles, cyclists and pedestrians. No annotations are used during training; at test time, we conduct class-agnostic evaluation on three main classes combined, since our goal is to train an unsupervised object detector that detects any object. We split the dataset into 73 training and 30 validation snippets. Both splits evenly distribute across both San Francisco and El Camino Real. The AV2 dataset is collected in six distinct cities in the U.S. cities with diverse weather (from snowy to sunny). It consists of 850 snippets, each 15 seconds long. The LiDAR data comes from two 32-beam lidars, spinning at 10 Hz in the same direction, but separated in orientation by 180°. We use the official train and validation split with 700 and 150 snippets respectively. For our detection models, we focus on the front-range setting with a region of interest (ROI) of $[0, 80]$ meters longitudinally and $[-40, 40]$ meters laterally with respect to the traveling direction of the ego vehicle. The selected ROI allows us to evaluate both near-range and long-range detections.

Implementation Details: For label refinement, we use track length $q = 6$ for both Pandaset and AV2. To update bounding box sizes in long tracklets, we use $r = 100\%$ for Pandaset and $r = 95\%$ for AV2. For Pandaset we run three rounds of self-training, with long tracklet refinement only applied in the last training round. For AV2 we employ two rounds of self-training, with long tracklet refinement in every training round. More details of the detection model, the tracker, and training are included in the supplementary.

Metrics: Our metrics focus on the goal of evaluating a class-agnostic object detector that is capable of detecting any object. Given this safety-critical objective, it is imperative for the detection results to not only have high IoU (intersection-over-union) with the ground truths, but also accurately measure how far the detected objects are from the self-driving vehicle. After combining annotations from all classes (vehicle, pedestrian, and cyclist), we propose the

	AP @ IoU			Recall @ IoU			AP @ ΔDTC (m)			Recall @ ΔDTC (m)		
	0.3	0.5	0.7	0.3	0.5	0.7	1.5	1.0	0.5	1.5	1.0	0.5
DBSCAN [17]	3.5	1.1	0.3	28.6	15.9	8.3	10.9	9.9	8.0	50.9	48.3	43.1
DBSCAN + init-train	21.6	12.0	6.4	42.0	26.0	13.8	41.3	39.3	34.2	71.8	69.4	62.1
DBSCAN + self-train [66]	20.1	12.3	6.3	42.2	26.1	13.5	39.2	37.4	30.1	71.8	69.4	61.4
PP score [66]	6.4	2.0	0.4	32.8	18.7	8.8	14.0	12.2	9.2	48.5	45.0	38.9
PP score + init-train [66]	28.4	14.0	4.9	47.4	26.6	12.5	42.2	38.9	31.7	69.5	66.0	57.5
MODEST [66] (1 traversal)	22.8	7.5	2.8	49.7	28.9	14.9	38.8	36.4	30.2	70.2	66.9	59.0
Ours	**43.5**	**29.5**	**18.1**	**62.8**	**44.8**	**28.1**	**51.8**	**48.8**	**41.1**	**75.7**	**72.3**	**63.7**

Table 1. **[Pandaset] Comparison against state-of-the-art**. Evaluated on the range 0-80m on all classes (vehicle, pedestrian, cyclist).

	AP @ IoU			Recall @ IoU			AP @ ΔDTC (m)			Recall @ ΔDTC (m)		
	0.3	0.5	0.7	0.3	0.5	0.7	1.5	1.0	0.5	1.5	1.0	0.5
DBSCAN [17]	2.1	1.0	0.4	26.4	17.9	10.9	6.8	6.4	5.4	47.2	45.7	41.9
DBSCAN + init-train	15.5	10.7	5.7	29.7	19.3	10.1	27.6	26.6	23.3	58.0	56.6	51.7
DBSCAN + self-train [66]	12.5	9.1	5.5	30.1	19.6	10.3	23.9	23.1	20.4	58.4	57.0	52.2
PP score [66]	5.5	2.5	0.9	33.7	22.8	13.8	11.6	10.4	8.4	49.1	46.5	41.6
PP score + init-train [66]	24.4	15.8	7.4	43.9	27.4	14.2	38.9	36.8	31.2	71.5	68.0	59.5
MODEST [66] (1 traversal)	20.6	9.9	2.7	47.0	28.6	13.0	38.6	35.9	28.1	74.0	70.4	61.0
Ours	**35.4**	**24.5**	**12.9**	**55.1**	**37.5**	**21.4**	**46.4**	**43.9**	**38.1**	**74.5**	**71.0**	**63.1**

Table 2. **[AV2] Comparison against state-of-the-art**. Evaluated on the range 0-80m on all classes (vehicle, pedestrian, cyclist). DBSCAN + self-train, MODEST and ours all employ two additional rounds of self-training after the initial training.

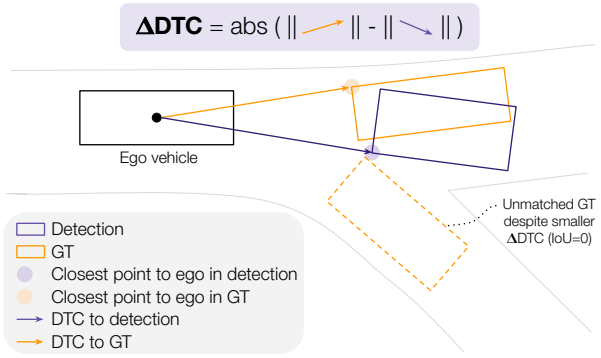

Figure 4. **Our novel matching criteria for detection metrics based on distance-to-collision (DTC)**. It prioritizes detecting the closest corner to the SDV correctly, as opposed to the full extent.

following evaluation steps to give a full picture about the performance of different methods:

- Measure Average Precision (AP) and Recall. AP is a more complete metric, measuring the area under the Precision Recall curve. However, we may identify properly detected objects as false positives due to the absence of annotations for objects in that class. On the other hand, Recall does not penalize false positive detections, making it more suitable for our experimental setting. Since arbitrarily large Recall can be achieved by producing a very large and diverse set of object detections, we evaluate all methods limiting the budget of detections to the 100 most confident ones.
- Compute these metrics for two different matching criteria, one based on the standard intersection over union (IoU) — which captures how well we predict the size of the objects, and a novel metric based on the difference in distance-to-collision (ΔDTC) — which measures how well we can predict how close objects are to the ego vehicle. Fig. 4 shows our proposed matching criteria based on ΔDTC. For a match between a pair of detection and ground-truth bounding boxes, we require that the two are under a certain ΔDTC threshold (in meters). Subject to this constraint, we perform IoU-based greedy matching to prevent associating detections with ground-truth bounding boxes corresponding to nearby actors (e.g., the dotted bounding box in the adjacent lane), as those may have a corner that is closer to the detection, but have no overlap.

Note that we only consider class-agnostic metrics rather than per-class metrics, since our task is to detect any object rather than predict their classes. As a result, naively computing per-class AP will incorrectly penalize true detections for other classes as false positives. Per-class Recall will be artificially low for rarer classes, since constraining the number of detections is often necessary for recall calculation (otherwise 100% recall is achieved by outputting detections everywhere), and highly represented classes in the data (*e.g.*, vehicles) tend to overwhelm the rest of the classes within a fixed number of detections.

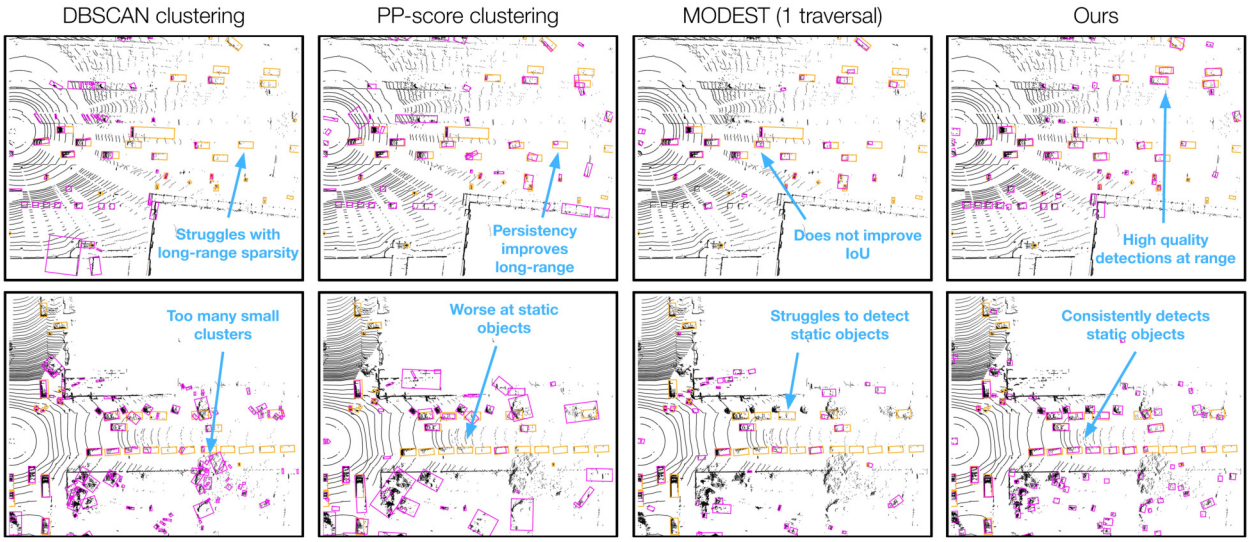

Figure 5. **[Pandaset] Qualitative results:** clustering algorithms (DBSCAN and PP-score), state-of-the-art (MODEST) and our method. We show detections in purple and ground-truth bboxes in orange.

Baselines: Following [66], we compare against the following unsupervised baselines. *DBSCAN* [17] performs density-based spatial clustering, grouping points with many nearby neighbors together. *DBSCAN + init-train* learns an object detector with the supervision of DBSCAN labels. *DBSCAN + self-train* adds two additional rounds of self-training after the initial training, where the thresholded detection outputs of the object detector are used as pseudo-labels for self-training. *PP score* [66] estimates the persistence of a point by measuring the variance in the number of LiDAR points within the point's neighborhood encountered in other observations of the same region. These PP-scores can then be used as a feature for clustering, retrieving mobile objects that have a low PP-score. Note that in the original paper multiple traversals over the same area are assumed, which pose a very strict requirement on data collection. To avoid imposing this strict requirement, we only consider a single traversal with multiple observations over a short period of time (*i.e.*, the length of the snippet). *PP score + init-train* trains a detector on the persistence-based clustering labels with one iteration of training. Finally, *MODEST (1 traversal)* adds two rounds of self-training after the initial training, where at each round the pseudo-labels coming from the latest self-trained model are filtered using PP scores to discard persistent clusters.

Benchmark against state-of-the-art: Tabs. 1 and 2 show a comparison with state-of-the-art unsupervised methods in Pandaset and Argoverse V2 Sensor datasets, respectively. The raw DBSCAN and PP score labels perform poorly since confidence scores are not output by these methods for ranking during metrics computation. Training with point clustering pseudo-labels improves the metrics for both DBSCAN and PP score. However, additional rounds of self-training lowered the AP for both IoU and DTC metrics. The results imply that MODEST (1 traversal) relies very heavily on multiple traversals over the same region. Our method *OYSTER* outperforms previous methods by a large margin across all IoU and distance thresholds, both for average precision and recall. Fig. 5 shows that our method can overcome failure modes exhibited by the baselines such as false negative detections for static objects, high density of false negatives at long-range, and some localization and size estimation errors.

Effect of the initial training range and ray-dropping: Since point clustering labels tend to be more reliable in the near range due to high point density, we find it beneficial to train on near-range LiDAR images with near-range clustering labels at first, and then rely on the translation-invariance property of ConvNets to generalize zero-shot to longer range. This is empirically demonstrated by the improvement from $M_1 \rightarrow M_2$ in Tab. 3, where the zero-shot generalization from short-range (0-40m) to full-range (0-80m) (M_2) performs much better than directly training on full-range (M_1). We also find that random ray dropping as a data augmentation technique during the initial training ($M_2 \rightarrow M_3$) can help the ConvNet generalize better to longer range in terms of detecting more objects, although it seems to slightly sacrifice localization accuracy as shown by the metrics evaluated at high IoU values, likely because ray dropping encourages the detector to focus more on one side of objects with the most points facing the ego vehicle rather than the overall object shapes.

Effect of short tracklet filtering: Using unsupervised tracking to ignore temporally inconsistent point clustering pseudo-labels is important. This makes the pseudo-labels less noisy, providing better supervision to the model. Training with these filtered set of labels is beneficial, as shown by the improvement from $M_3 \rightarrow M_4$ in Tab. 3.

ID	ITR	RD	SRef	ST2	LRef	AP @ IoU 0.3	AP @ IoU 0.5	AP @ IoU 0.7	Recall @ IoU 0.3	Recall @ IoU 0.5	Recall @ IoU 0.7	AP @ ΔDTC 1.5	AP @ ΔDTC 1.0	AP @ ΔDTC 0.5	Recall @ ΔDTC 1.5	Recall @ ΔDTC 1.0	Recall @ ΔDTC 0.5
M_1	80					21.6	12.0	6.4	42.0	26.0	13.8	41.3	39.3	34.2	71.8	69.4	62.1
M_2	40					23.2	13.0	5.3	42.9	25.7	12.8	41.8	39.5	33.9	72.0	69.5	62.1
M_3	40	✓				26.6	14.5	4.1	44.6	24.5	10.0	46.1	43.4	36.5	72.0	69.0	60.8
M_4	40	✓	✓			32.3	17.8	7.5	51.6	31.9	15.4	48.7	46.1	40.3	74.1	71.4	64.6
M_5	40	✓	✓	✓		**43.5**	26.2	13.2	58.6	38.4	20.6	**55.2**	**52.3**	**45.2**	**77.1**	**73.9**	**65.7**
M_6	40	✓	✓	✓	✓	**43.5**	**29.5**	**18.1**	**62.8**	**44.8**	**28.1**	51.8	48.8	41.1	75.7	72.3	63.7

Table 3. **[Pandaset] Ablation study**. All class evaluation in the range 0-80m. Legend: ID=Model identifier, ITR=Initial training range (first iteration) from 0 to X meters, RD=Ray-dropping, SRef=Tracking & short tracklet filtering, ST2=2 rounds of additional self-training, LRef=Long tracklet refinement.

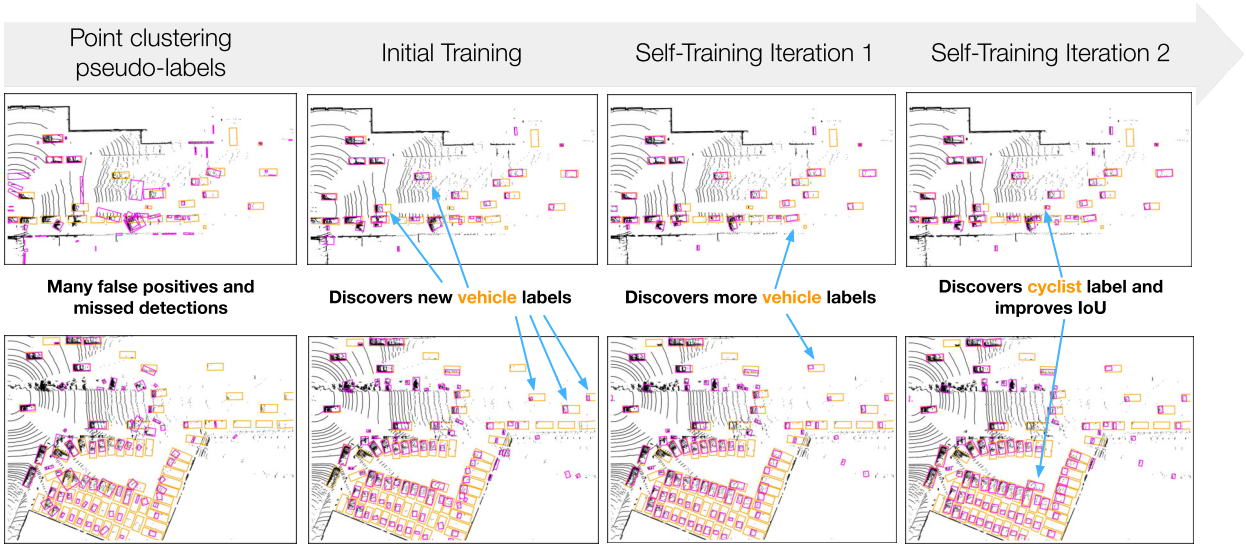

Figure 6. **[Pandaset] Evolution of self-training labels**: with initial training and more rounds of self-training, we are able to remove false positives, discover missed detections, and improve bbox accuracy. We show detections in purple and ground-truth bboxes in orange.

Effect of self-training loop and long tracklet refinement: Fig. 6 visually shows the evolution of the detections through self-training (loop shown in Fig. 2). Initially, point clustering labels are quite noisy, identifying many background regions as objects. After the first training iteration, the model can clearly distinguish foreground from background (*i.e.*, removes false positives), and discovers other vehicles missed by clustering. Further rounds of self-training discover additional objects such as vehicles and cyclists, and increases the object localization accuracy. Quantitatively, the effect of 3 rounds of self-training is shown in Tab. 3, where we can see M_5 and M_6 clearly outperform the other ablations. The difference between M_5 and M_6 is that in M_5 we skip the long tracklet CornerAlign refinement stage, going directly from tracking and short tracklet filtering to retraining. In M_6 we refine the long tracklets in the pseudo-labels provided by M_5, and add one more round of self-training with the refined pseudo-labels. We can see that the refinement stage is able to further improve the IoU metrics. Note that the DTC metrics dropped with long tracklet refinement, most likely because increasing the bounding box size with inaccurate heading negatively impacts the accuracy of the closest distance to the ego-vehicle. This showcases the importance of having both IoU and DTC metrics, as they focus on different aspects of detection accuracy.

5. Conclusion

We have proposed a novel method, *OYSTER*, for unsupervised object detection from LiDAR point clouds. Using weak object priors (near-range point clustering) as a bootstrapping step, our method can train an object detector with no human annotations, by first utilizing the translation equivariance of CNNs to generate long-range pseudo-labels, and then deriving self-supervision signals from the temporal consistency of object tracks. Our proposed self-training loop is highly effective for teaching an unsupervised detector to self-improve. We validate our results on two real-world datasets, Pandaset and Argoverse 2 Sensor, where our model outperforms prior unsupervised methods by a significant margin. Making self-supervised learning work on real-world robot perception is an exciting challenge for AI, and our work takes a step towards allowing robots to make sense of the visual world without human supervision.

References

[1] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2189–2202, 2012. 1

[2] Yoshua Bengio, Yann Lecun, and Geoffrey Hinton. Deep learning for ai. *Communications of the ACM*, 64(7):58–65, 2021. 1

[3] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. 2

[4] Hakan Bilen, Marco Pedersoli, and Tinne Tuytelaars. Weakly supervised object detection with convex clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1081–1089, 2015. 1

[5] Igor Bogoslavskyi and Cyrill Stachniss. Fast range image-based segmentation of sparse 3d laser scans for online operation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9-14, 2016*, pages 163–169. IEEE, 2016. 2

[6] Christopher P. Burgess, Loïc Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matthew Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *CoRR*, abs/1901.11390, 2019. 1, 2

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 3

[8] João Carreira and Cristian Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 3241–3248. IEEE Computer Society, 2010. 2

[9] Jun Cen, Peng Yun, Junhao Cai, Michael Yu Wang, and Ming Liu. Open-set 3d object detection. *CoRR*, abs/2112.01135, 2021. 3

[10] Eric Crawford and Joelle Pineau. Spatially invariant unsupervised object detection with convolutional neural networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3412–3420. AAAI Press, 2019. 2

[11] Eric Crawford and Joelle Pineau. Exploiting spatial invariance for scalable unsupervised object tracking. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 3684–3692. AAAI Press, 2020. 2

[12] Akshay Raj Dhamija, Manuel Günther, Jonathan Ventura, and Terrance E. Boult. The overlooked elephant of object detection: Open set. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2020, Snowmass Village, CO, USA, March 1-5, 2020*, pages 1010–1019. IEEE, 2020. 3

[13] Bertrand Douillard, James Underwood, Noah Kuntz, Vsevolod Vlaskine, Alastair Quadros, Peter Morton, and Alon Frenkel. On the segmentation of 3d lidar point clouds. In *2011 IEEE International Conference on Robotics and Automation*, pages 2798–2805. IEEE, 2011. 3

[14] Bertrand Douillard, James Patrick Underwood, Noah Kuntz, Vsevolod Vlaskine, Alastair James Quadros, Peter Morton, and Alon Frenkel. On the segmentation of 3d LIDAR point clouds. In *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*, pages 2798–2805. IEEE, 2011. 2

[15] Martin Engelcke, Adam R. Kosiorek, Oiwi Parker Jones, and Ingmar Posner. GENESIS: generative scene inference and sampling with object-centric latent representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 1, 2

[16] S. M. Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Koray Kavukcuoglu, and Geoffrey E. Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3225–3233, 2016. 2

[17] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996. 3, 6, 7

[18] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004. 1

[19] Katerina Fragkiadaki, Pablo Arbelaez, Panna Felsen, and Jitendra Malik. Learning to segment moving objects in videos. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 4083–4090. IEEE Computer Society, 2015. 2

[20] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1

[21] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2424–2433. PMLR, 2019. 2

[22] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. In *Advances in Neu-*

ral Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 6691–6701, 2017. 2

[23] Akshita Gupta, Sanath Narayan, K. J. Joseph, Salman H. Khan, Fahad Shahbaz Khan, and Mubarak Shah. OW-DETR: open-world detection transformer. *CoRR*, abs/2112.01513, 2021. 3

[24] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[26] Michael Himmelsbach, Felix V Hundelshausen, and H-J Wuensche. Fast segmentation of 3d point clouds for ground vehicles. In *2010 IEEE Intelligent Vehicles Symposium*, pages 560–565. IEEE, 2010. 3

[27] Miguel Jaques, Michael Burke, and Timothy M. Hospedales. Physics-as-inverse-graphics: Unsupervised physical parameter estimation from video. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 2

[28] Jindong Jiang and Sungjin Ahn. Generative neurosymbolic machines. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 2

[29] Jindong Jiang, Sepehr Janghorbani, Gerard de Melo, and Sungjin Ahn. SCALOR: generative world models with scalable object representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 2

[30] K. J. Joseph, Salman H. Khan, Fahad Shahbaz Khan, and Vineeth N. Balasubramanian. Towards open world object detection. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 5830–5840. Computer Vision Foundation / IEEE, 2021. 3

[31] Rishabh Kabra, Daniel Zoran, Goker Erdogan, Loic Matthey, Antonia Creswell, Matthew Botvinick, Alexander Lerchner, and Christopher P. Burgess. Simone: View-invariant, temporally-abstracted object representations via unsupervised video decomposition. *CoRR*, abs/2106.03849, 2021. 2

[32] Andrej Karpathy, Stephen D. Miller, and Li Fei-Fei. Object discovery in 3d scenes via shape analysis. In *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, pages 2088–2095. IEEE, 2013. 2

[33] Adam R. Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. Sequential attend, infer, repeat: Generative modelling of moving objects. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8615–8625, 2018. 1, 2

[34] Adam R. Kosiorek, Sara Sabour, Yee Whye Teh, and Geoffrey E. Hinton. Stacked capsule autoencoders. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 15486–15496, 2019. 1, 2

[35] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019. 3, 4

[36] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 1

[37] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 4

[38] Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. SPACE: unsupervised object-oriented scene representation via spatial attention and decomposition. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 1, 2

[39] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 1, 2

[40] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, pages 405–421. Springer, 2020. 2

[41] Dimity Miller, Lachlan Nicholson, Feras Dayoub, and Niko Sünderhauf. Dropout sampling for robust object detection in open-set conditions. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pages 1–7. IEEE, 2018. 3

[42] Mahyar Najibi, Jingwei Ji, Yin Zhou, Charles R Qi, Xinchen Yan, Scott Ettinger, and Dragomir Anguelov. Motion inspired unsupervised perception and prediction in au-

tonomous driving. In *European Conference on Computer Vision*, pages 424–443. Springer, 2022. 2

[43] Rahul Kumar Namdev, Abhijit Kundu, K. Madhava Krishna, and C. V. Jawahar. Motion segmentation of multiple objects from a freely moving monocular camera. In *IEEE International Conference on Robotics and Automation, ICRA 2012, 14-18 May, 2012, St. Paul, Minnesota, USA*, pages 4092–4099. IEEE, 2012. 2

[44] Anestis Papazoglou and Vittorio Ferrari. Fast object segmentation in unconstrained video. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 1777–1784. IEEE Computer Society, 2013. 2

[45] Trung Pham, B. G. Vijay Kumar, Thanh-Toan Do, Gustavo Carneiro, and Ian D. Reid. Bayesian semantic instance segmentation in open set world. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part X*, volume 11214 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2018. 3

[46] Jordi Pont-Tuset, Pablo Arbelaez, Jonathan T. Barron, Ferran Marqués, and Jitendra Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(1):128–140, 2017. 2

[47] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 1

[48] Sara Sabour, Andrea Tagliasacchi, Soroosh Yazdani, Geoffrey E. Hinton, and David J. Fleet. Unsupervised part representation by flow capsules. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 9213–9223. PMLR, 2021. 2

[49] Josef Sivic, Bryan C. Russell, Alexei A. Efros, Andrew Zisserman, and William T. Freeman. Discovering objects and their localization in images. In *10th IEEE International Conference on Computer Vision (ICCV 2005), 17-20 October 2005, Beijing, China*, pages 370–377. IEEE Computer Society, 2005. 2

[50] Andrew Stein, Derek Hoiem, and Martial Hebert. Learning to find object boundaries using motion cues. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007. 1

[51] Karl Stelzner, Kristian Kersting, and Adam R. Kosiorek. Decomposing 3d scenes into objects via unsupervised volume segmentation. *CoRR*, abs/2104.01148, 2021. 2

[52] Jasper R. R. Uijlings, Koen E. A. van de Sande, Theo Gevers, and Arnold W. M. Smeulders. Selective search for object recognition. *Int. J. Comput. Vis.*, 104(2):154–171, 2013. 2

[53] Sjoerd van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. 2

[54] Sara Vicente, Carsten Rother, and Vladimir Kolmogorov. Object cosegmentation. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, pages 2217–2224. IEEE Computer Society, 2011. 2

[55] Huy V. Vo, Patrick Pérez, and Jean Ponce. Toward unsupervised, multi-object discovery in large-scale image collections. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIII*, volume 12368 of *Lecture Notes in Computer Science*, pages 779–795. Springer, 2020. 2

[56] Huy V. Vo, Elena Sizikova, Cordelia Schmid, Patrick Pérez, and Jean Ponce. Large-scale unsupervised object discovery. *CoRR*, abs/2106.06650, 2021. 2

[57] Fang Wan, Pengxu Wei, Jianbin Jiao, Zhenjun Han, and Qixiang Ye. Min-entropy latent model for weakly supervised object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1297–1306, 2018. 1

[58] Tianyu Wang, Miaomiao Liu, and Kee Siong Ng. Spatially invariant unsupervised 3d object segmentation with graph neural networks. *CoRR*, abs/2106.05607, 2021. 1

[59] Yuqi Wang, Yuntao Chen, and Zhaoxiang Zhang. 4d unsupervised object discovery. *arXiv preprint arXiv:2210.04801*, 2022. 3

[60] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. 2, 5

[61] Kelvin Wong, Shenlong Wang, Mengye Ren, Ming Liang, and Raquel Urtasun. Identifying unknown instances for autonomous driving. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pages 384–393. PMLR, 2019. 3

[62] Fanyi Xiao and Yong Jae Lee. Track and segment: An iterative unsupervised approach for video object proposals. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 933–942. IEEE Computer Society, 2016. 2

[63] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, et al. Pandaset: Advanced sensor suite dataset for autonomous driving. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3095–3101. IEEE, 2021. 2, 5

[64] Bin Yang, Min Bai, Ming Liang, Wenyuan Zeng, and Raquel Urtasun. Auto4d: Learning to label 4d objects from sequential point clouds, 2021. 4, 5

[65] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018. 3, 4

[66] Yurong You, Katie Luo, Cheng Perng Phoo, Wei-Lun Chao, Wen Sun, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Learning to detect mobile objects from lidar scans without labels. In *CVPR*, 2022. 2, 3, 4, 6, 7

[67] Hong-Xing Yu, Leonidas J. Guibas, and Jiajun Wu. Unsupervised discovery of object radiance fields. *CoRR*, abs/2107.07905, 2021. 2

[68] Xiaopeng Zhang, Jiashi Feng, Hongkai Xiong, and Qi Tian. Zigzag learning for weakly supervised object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4262–4270, 2018. 1

[69] Xiao Zhang, Wenda Xu, Chiyu Dong, and John M Dolan. Efficient l-shape fitting for vehicle detection using laser scanners. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 54–59. IEEE, 2017. 3

[70] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 1