`SeMoLi`: What Moves Together Belongs Together

Jenny Seidenschwarz[1,2*]   Aljoša Ošep[2]   Franceso Ferroni[2]   Simon Lucey[3]   Laura Leal-Taixé[2]

[1]Technical University of Munich   [2]NVIDIA   [3]University of Adelaide
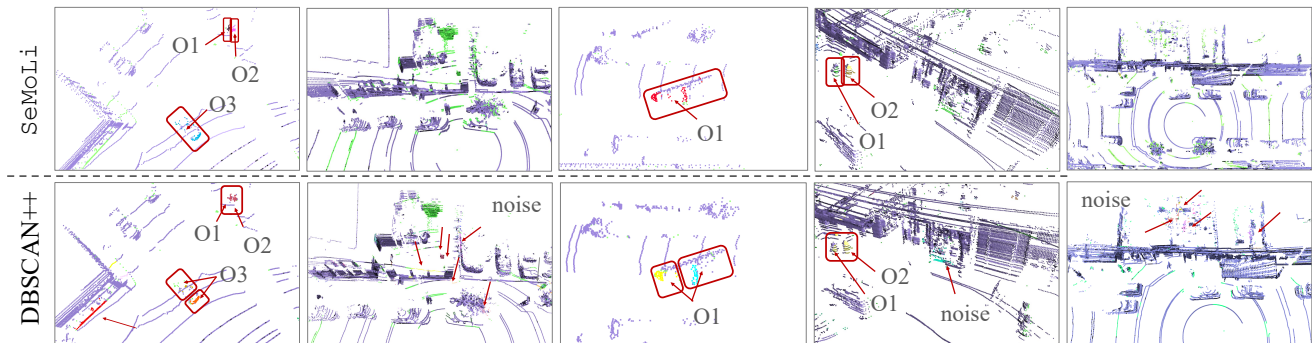
Figure 1. **Towards learning to pseudo-label**: We propose `SeMoLi`, a data-driven approach for segmenting moving instances in point clouds (*top*), that we utilize to learn to detect moving objects (O) in Lidar. We visually contrast `SeMoLi` to prior art, that tackles similar problem via density-based clustering (DBSCAN) [24]. We visualize the whole point cloud in purple, and dynamic points, used as input to our method and baseline to localize moving instances, in green. We color-code individual segmented instances. From left to right `SeMoLi` (i) segments objects even for sparse point clouds and suffers less from under-segmentation, (ii) is able to learn to filter noise from the filtered point cloud, (iii) leads to less over-segmentation, and (iv) generalizes better to different classes. *Best seen in color, zoomed.*

Abstract

We tackle semi-supervised object detection based on motion cues. Recent results suggest that heuristic-based clustering methods in conjunction with object trackers can be used to pseudo-label instances of moving objects and use these as supervisory signals to train 3D object detectors in Lidar data without manual supervision. We re-think this approach and suggest that both, object detection, as well as motion-inspired pseudo-labeling, can be tackled in a data-driven manner. We leverage recent advances in scene flow estimation to obtain point trajectories from which we extract long-term, class-agnostic motion patterns. Revisiting correlation clustering in the context of message passing networks, we learn to group those motion patterns to cluster points to object instances. By estimating the full extent of the objects, we obtain per-scan 3D bounding boxes that we use to supervise a Lidar object detection network. Our method not only outperforms prior heuristic-based approaches (57.5 AP, +14 improvement over prior work), more importantly, we show we can pseudo-label and train object detectors across datasets.

---

* Correspondence to j.seidenschwarz@tum.de.

1. Introduction

We tackle semi-supervised object detection in Lidar data in the context of embodied navigation.

Status quo. The established approach to 3D Lidar object detection is to pre-define object classes of interest and train data-driven models with manually labeled data in a fully supervised fashion. Thanks to Lidar-centric labeled datasets [3, 10, 33] and developments in learning representations from unordered point sets [28, 29], the quality in Lidar-based detection continues to improve. However, this process that relies on manual labeling is notoriously expensive and, importantly, does not scale well to rare classes. Before the advent of end-to-end data-driven Lidar perception, pioneering works [12, 13, 22, 23, 34, 35, 47] were inspired by the Gestalt principle [40] of spatial proximity, *i.e.*, the intuition that spatially close-by points belong together. To this end, these works utilize local, distance-based point clustering methods to segment objects via bottom-up grouping. They are class-agnostic and general, but inherently unable to benefit from the increasing amount of available data, and therefore, fall behind data-driven methods. Recently, heuristic approaches resurfaced within multi-stage pipelines for data auto-labeling in the context of semi-supervised 3D

object detection [24, 26, 48]. These auto-labeling pipelines (i) segment objects using local, bottom-up clustering methods, (ii) use Kalman filters to track segmented regions across time, and (iii) register segments across time to estimate amodal extents of objects to generate pseudo-labels for object detection training. While impressive, they still suffer from the same drawback: they are inherently unable to improve pseudo-labeling performance in a data-driven fashion.

Stirring the pot. We re-think this approach and suggest that both object detection training based on pseudo-labels, as well as the pseudo-labeling process can be tackled in a data-driven manner. Prior efforts, too, leverage *some* amount of labeled data to tune the hyper-parameters of the auto-labeling engine and consolidate design decisions. By contrast, we embrace a data-driven approach fully. We study how to train a pseudo-labeling network in a way that it can still be general and applied to an open-world setting as well as how its performance changes with different amounts of training data.

Learning to *motion*-cluster. Our approach *Segment Moving in Lidar* (SeMoLi) receives a set of Lidar point trajectories, obtained from a sequence of Lidar data. In contrast to prior heuristic approaches, we explore Gestalt principles to *train* a Message Passing Network (MPN) [11] with the objective of grouping points in a class-agnostic manner. Additionally to proximity, we leverage the intuition of *common fate*, *i.e.*, we learn to group points based on the observation that *what moves together, belongs together* given *some* labeled instances of moving objects. This way, we make motion a *first-class citizen*. This, in turn, streamlines pseudo-labeling and alleviates the need for manual geometric clustering. We then use SeMoLi to identify motion patterns in unlabeled sequences and pseudo-label these at the per-point segmentation level. By extracting bounding boxes and inflating them to better match the amodal ground truth bounding boxes we train an off-the-shelf 3D detection network.

Impact. Our data-driven approach not only allows us to perform better as compared to a hand-crafted pipeline based on density-based clustering [24] – more importantly, we demonstrate that SeMoLi's performance improves with increasing amount of data while still remaining a general approach applicable on different classes and datasets. Just as important, we democratize research of motion-inspired auto-labeling in Lidar streams by making our models, code, as well as previously inaccessible baselines [24] publicly available to foster future research.

To conclude, as our **main contribution** we (i) show that class-agnostic, motion-based segmentation for pseudo-labeling can be tackled in a data-driven manner. To operationalize this idea, we (ii) propose a *data-driven* point clustering method leveraging motion patterns that utilizes

MPNs, constructs a graph over a Lidar point cloud, and learns which points *belong together* based on spatial and motion cues. We (iii) demonstrate this network can be used to successfully pseudo-label unlabeled sequences even across unknown classes and datasets without the requirement of re-training. Finally, we (iv) make this field of research reproducible and establish a solid ground for making progress in this field of research as a community. We hope that this is a stepping stone towards a better world in which friendly robots do not accidentally crash into rare moving objects just because no one wants to label data manually.

2. Related Work

Lidar-based 3D Object Detection. Amodal 3D localization of objects is instrumental for situational awareness in dynamic environments and has been playing a pivotal role in autonomous navigation since its inception [27, 36]. Fueled by the recent availability of high-quality Lidar-perception-centric datasets [4, 10, 33], several methods directly learn a representations from unordered point sets [28, 29] to localize and classify objects [20, 30–32, 45], or utilize sparse 3D convolutional backbones [5, 43, 49]. Several methods [44, 46] flatten 3D voxel representation along the height dimension, while [17] replaces regular grid-based discretization with vertical columns. All aforementioned methods rely on manually labeled amodal 3D bounding boxes that encapsulate the full extent of the object. Manual annotations of such regions are costly, and, for practical purposes, constraint *target* classes that must be detected to a fixed, pre-defined set of objects.

Lidar-based bottom-up grouping. Prior to the advent of end-to-end object detection models, Lidar perception models have been relying on bottom-up point grouping [8]. Several methods utilize the connected components algorithm on graphs, implicitly defined over the rasterized birds-eye-view representation of point clouds [16, 35] or the full point cloud [38] to localize disconnected regions, *i.e.*, objects in Lidar scans. Instead of utilizing graph-based representations, several methods [2, 14, 21, 24, 25, 42] utilize robust density-based clustering methods (DBSCAN) [9] to group points. The advantage of these methods is that the segmentation is class-agnostic, largely relying on the Gestalt principle [40] of spatial proximity, which is a strong cue in the Lidar domain. However, these methods do not benefit from the increasing amount of (labeled) data to improve segmentation performance. Our work is inspired by prior efforts that localize objects as connected components in graphs [35, 38]. We revisit these ideas in the context of data-driven grouping while still remaining class-agnostic behavior.

Motion-supervised object detection in Lidar. Tracking-based semi-supervised learning in Lidar is investigated
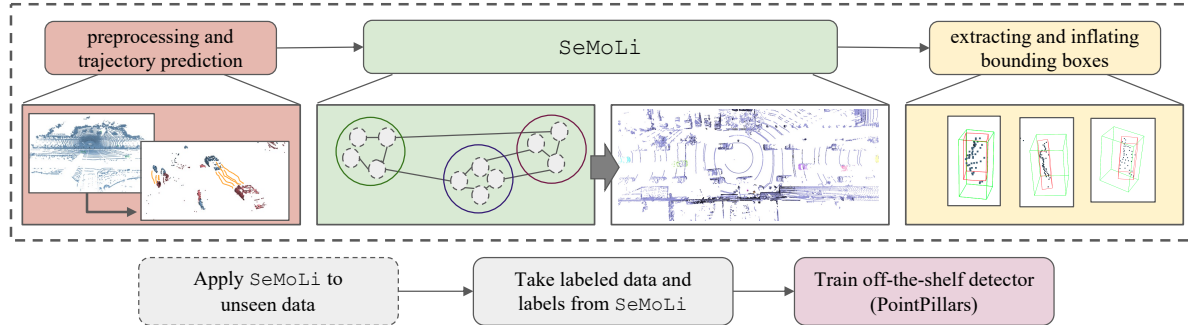
Figure 2. **Segment Moving in Lidar for Pseudo-Labeling:** We first preprocess the point cloud to remove static points and predict per-point trajectories on the filtered point cloud (*preprocessing and trajectory prediction*). Then, we extract velocity-based features from the trajectories and *learn* to cluster, *i.e.*, segment points based on motion-patters using a Message-Passing Netowrk [11] in a fully data-driven manner (SeMoLi). From point segments, we extract bounding boxes and inflate them (*extracting and inflating bounding boxes*). Finally, we apply our approach on *unlabeled* Lidar streams to obtain pseudo-labels, that we use to train object detectors.

in [34] by performing bottom-up segmentation of Lidar point clouds, followed by Kalman-filter-based object tracking of Lidar segments to expand the core labeled training set via tracking. A recent work by [24] follows a similar pipeline, however, first filters out stationary points to obtain supervision for moving objects, as needed to train object detectors. Another related approach by [48] follows a similar pipeline, and additionally proposes a novel data augmentation that allows object detectors to generalize to far ranges solely based on instances observed in near sensing areas. Unfortunately, the aforementioned are proprietary and do not share models, experimental data, or evaluation/implementational details. As these consist of hand-crafted pipelines, results reported by these methods are difficult to reproduce, hindering the community's progress in this field of research.

We tackle a similar problem, however, by contrast, we propose a data-driven, graph-based model that learns a representation of the point cloud, suitable for localizing moving objects in Lidar. Our method performs favorably compared to state-of-the-art even when trained on a tiny amount of labeled data, and further improves with an increased amount of training data. Moreover, we show we can apply our approach to other datasets without any re-training of our pseudo-labeling model. To foster future research, we make our models, code, and experimental data publicly available.

3. Learning Moving Objecs in Lidar

We present *Segment Moving in Lidar* (SeMoLi), our data-driven approach for motion-inspired pseudo-labeling for 3D object detection. Our *teacher* network learns to cluster points in Lidar point clouds based on their motion patterns (Sec. 3.1). This step can also be seen as point cloud segmentation and corresponds to localizing moving object instances. Once trained, we utilize SeMoLi to segment *unlabeled* Lidar data, estimate amodal object-oriented bounding

boxes from the segments and use these as a supervisory signal to train a *student* 3D object detector (Sec. 3.2).

3.1. Learning to Segment Moving Objects

Given input Lidar point clouds $\mathcal{P}$, our task is to localize individual moving objects as 3D bounding boxes. Each object will be represented by an a-priori unknown number of points, which we group together based on cues derived from Gestalt principles [40]. We propose to use *proximity*, *i.e.*, nearby points likely belong to the same object, and *common fate*, *i.e.*, what moves together belongs together. Prior works cluster points using, *e.g.*, density [9, 24, 39], or graph-based clustering methods [38] that base grouping decisions on the local context and do not leverage the benefits of learning-based algorithms. This is problematic for Lidar. On the one hand, signal sparsity and the vast range of object sizes that appear in the dataset (*e.g.*, pedestrian *vs*. truck) often lead to over- and under-segmentation [13]. On the other hand, the performance of such approaches suffers heavily in the presence of background noise.

Correlation clustering revisited. To segment point clouds into a set of instances, we lean on correlation clustering [1]. We represent a point cloud as a weighted graph $G = (V, E)$ with nodes V and edges E. A node n_i represents a point and encodes its spatial position and motion as node features $h_i^{(0)}$ while an edge $e_{i,j}$ represents the geometrical connection between two points i and j with edge features $h_{ij}^{(0)}$ representing their relationships. The clustering algorithm then cuts edges to obtain a set of connected components which represent point cloud instance segmentations. We revisit correlation clustering in the context of Message Passing Networks (MPNs) [11]. Hence, we define node and edge features in a data-driven manner to *learn* edge scores given *some* labeled data.

Talk to your neighbors. MPNs propagate information across the graph and ensure that the *learned* graph parti-

tioning does not *only* rely on the local relationships between points. Initial node $h_i^{(0)}$ and edge features $h_{i,j}^{(0)}$ are updated in an iterative manner over L layers. This ensures that the final edge features $h_{i,j}^{(L)}$ contains global information, providing the necessary context needed to decompose the graph into object instances. We utilize a binary edge classifier on top of $h_{i,j}^{(L)}$ to obtain our edge scores. In the following, we discuss how we construct and parameterize our input graph, and how we learn the representation using a MPN.

3.1.1 Graph Construction

Point cloud sequence preprocessing. Given a Lidar point cloud sequence $\mathcal{P} = \{P^t \in \mathbb{R}^{N^t \times 3}\}, t \in 1, \ldots, T$. We first remove static points from the raw point clouds in P^t. This yields a sequence of stationary point clouds $\tilde{\mathcal{P}} = \{\tilde{P}^t \in \mathbb{R}^{M^t \times 3}\}, t \in 1, \ldots, T$, where $M^t \leq N^t$. Our pre-filtering step closely follows prior work [7, 24] (see appendix for details). For simplicity, we omit index t in the remainder of the paper.

Motion cues. For each filtered point cloud $\tilde{P}$ we predict ego-motion compensated point trajectories $\mathcal{T} \in \mathbb{R}^{M \times (3 \times 24)}$ using a self-supervised trajectory prediction network [37]. Each point trajectory $\tau_i \in \mathcal{T}$ is defined as a sequence of point positions $\tau_i = \{p_i^k\}_{k=0}^{24}$. We utilize trajectory information to encode motion cues into our graph features, as described below.

Graph nodes V. For each filtered point cloud $\tilde{P}$, we view points as nodes in a graph with corresponding node features.

Node parametrization. The node feature $h_i^{(0)}$ includes its spatial coordinates (x, y, z) and the statistical measures (mean, min, max) of the velocities along its trajectory:

$$h_i^{(0)} = (x_i, y_i, z_i, \text{mean}(v_{\tau_i}), \min(v_{\tau_i}), \max(v_{\tau_i})). \quad (1)$$

These features capture both the spatial position and the dynamic behavior of each point. The velocity at each time step is calculated as the difference between consecutive points in the trajectory: $v_{\tau_i} = \{p_i^{k+1} - p_i^k\}_{k=0}^{k=24}$.

Graph edges E. We connect nodes $n_i \in V$ with edges $e_{i,j}$ that hypothesize point-to-instance memberships. Utilizing a fully connected graph is computationally infeasible and introduces a large amount of negative edges that impedes the learning process. Hence, we leverage proximity as an inductive bias based on the intuition that "far away" points are unlikely to belong to the same object instance. We constrain node connectivity to a set of k-nearest neighboring nodes in terms of Euclidean distance. We discuss alternative graph construction strategy in the experimental section (Sec. 4.2). Note that the final edge set is still biased towards negative edges.

Edge parametrization. We parametrize edges e_{ij} connecting nodes i and j via initial edge features as the difference in their spatial coordinates:

$$h_{ij}^{(0)} = (x_i - x_j, y_i - y_j, z_i - z_j) \quad (2)$$

This captures the relative spatial relationship between the points in the point cloud. To ensure each edge obtains a global view of the point cloud, as needed for reliable clustering, we perform several Message Passing iterations, as described below.

3.1.2 Representating Learning via Message Passing

The message passing algorithm involves iteratively updating node and edge embeddings. For a fixed number of iterations L, we perform updates as follows:

Edge update. At each step $l \in \{1, \ldots, L\}$, we update the embedding of an edge e_{ij} connecting nodes i and j based on its previous embedding $h_{ij}^{(l-1)}$ and the embeddings of the adjacent nodes $h_i^{(l-1)}, h_j^{(l-1)}$:

$$h_{ij}^{(l)} = f\left(h_{ij}^{(l-1)}, h_i^{(l-1)}, h_j^{(l-1)}\right), \quad (3)$$

where $f(\cdot, \cdot, \cdot)$ is an shared-weight update function, that consists of a linear, a normalization, and a dropout layer.

Node update. Similarly, we update node embeddings based on their previous embeddings as well as their neighbors previous embeddings:

$$m_{i,j}^{(l)} = g\left(h_{ij}^{(l)}, h_j^{l-1}, h_i^{(l-1)}\right), h_i^{(l)} = \phi(\{m_{i,j}^{(l)}\}_{j \in N_i}), \quad (4)$$

where $g(\cdot, \cdot)$ is the node update function with shared weights over all layers that constitutes of a linear layer, a normalization layer, and a dropout layer, ϕ is a mean aggregation function and N_i denotes the set of nodes adjacent to node i. The iterative process of updating node and edge embeddings allows for the integration of local and global information in the graph, enabling the algorithm to capture complex patterns and relationships within the data.

Edge classification. To determine the final edge score, we feed the final edge features $h_{ij}^{(L)}$ through a final linear layer f_f followed by a sigmoid layer σ:

$$\tilde{h}_{ij}^{(L)} = \sigma\left(f_f(h_{ij}^{(L)})\right). \quad (5)$$

For training details we refer the reader to appendix.

Postprocessing. During inference, we first cut negative edges with score $\tilde{h}_{ij}^{(L)} < 0.5$. To ensure robustness towards a small set of possibly miss-classified outlier edges, we apply correlation clustering [1] using our learned edge scores (details in the appendix) on top of this graph. We discard all singleton nodes without edges. The resulting point clusters $c_c \in C_c$ represent our segmented object instances.

3.2. Learning to Detect Moving Objects

Training the student network requires to, first, transform a segmented point cluster c_c to a bounding box b_c. Then we enhance them by inflation to obtain our final pseudo-labels.

From point clusters to bounding boxes. Given a set of points that constitutes a point cluster $c^c \in C_c$, we determine the translation vector t_c by taking the midpoint of all points. Since each point i has a trajectory τ_i assigned to it, we can compute the mean trajectory $\hat{\tau}_c = \{\hat{p}_c^k\}_{k=0}^{24}$ and leverage $\hat{p}_c^1$ and $\hat{p}_c^2$ to determine the heading of the object in xy-direction α_c. With the heading α_c and the translation vector t_c we can transform the points to determine axis-aligned length, width, and height of the bounding box lwh_c which yields the 3D bounding box $b_c = [t_c, lwh_c, \alpha_c]$.

Bounding Box Inflation. The bounding boxes b_c represent the most compact enclosing axis-aligned cuboid. Ground truth bounding boxes are represented by typically more loose, amodal bounding boxes. To adapt b_c to the corresponding ground truth data, we inflate them to have a minimum length, width, and height of x_{min}, y_{min}, and z_{min}, respectively, to obtain our final pseudo-labels.

Detector training. We train PointPillars [18] (PP) two-stage object detector in a class-agnostic setting using our generated pseudo-labels. We utilize the detector implementation as well as hyperparameters settings from [6] (see appendix), and perform the following changes. To account for objects of various sizes, we adapt anchor box generation with various size parameters (see appendix for details). We adapt the detection region to a planar $100x40m$ field, centered at the egovehicle (*cf*., [24]). We use binary cross-entropy loss to train our object classifier (`object`, `background`).

4. Experimental Validation

In this section, we thoroughly ablate our motion-inspired, data-driven approach *Segment Moving in Lidar* (SeMoLi) and its individual components. In Sec. 4.1, we outline our evaluation test-bed, which we use in Sec. 4.2 to discuss several design decisions behind our pseudo-labeling network. In Sec. 4.4, we evaluate the performance of our object detector, trained using our pseudo-labels.

4.1. Experimental Setup

Datasets. We evaluate our method using Waymo Open dataset [33] and only utilize Lidar data. The dataset provides labels as amodal 3D bounding boxes for `pedestrian`, `vehicle` and `cyclist` classes. We assess the generalization of our method on Argoverse2 dataset [41] which provides finer-grained semantic labels. Both datasets were recorded with different types of (proprietary) Lidar sensors (for details we refer to the appendix).
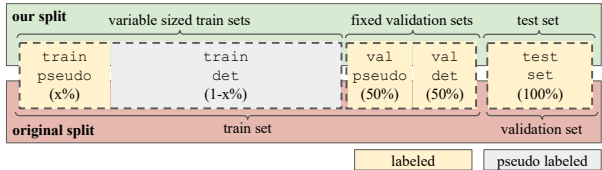

Figure 3. **Train and validation splits:** We conduct our experiments using Waymo training set, for which manual labels are available. We pre-fix two separate validation sets, one for validating pseudo-labels (`val_pseudo`), and one for end-model detector performance (`val_det`). We report performance on varying ratios x for training SeMoLi (`train_pseudo`) and generating pseudo-labels for training our detector (`train_det`).

Evaluation setup. Following [24], we evaluate SeMoLi as well as the final object detector in a $100x40m$ rectangular region (region of highest importance for autonomous vehicles), centered in the ego-vehicle Lidar sensor center. As our pseudo-labels and trained detectors do not provide fine-grained semantic information, we follow the literature [24, 48] and evaluate both, SeMoLi and DeMoLi in a class-agnostic setting. We assess our object detectors on `moving-only` (*cf*., [24]) as well as `all` labeled objects to see how well training instances mined from moving regions generalize to non-moving objects. We consider an object as moving if its velocity is larger than $1m/s$. We treat `non-moving` instances as ignore regions, *i.e.*, we ignore detections having *any* intersection with static objects, consistent with the methodology used in prior works [24].

Semantic Oracle. For per-class recall analysis we assign labels to class-agnostic detections if they have any 3D IoU overlap with labeled boxes.

Metrics. To evaluate *pseudo-label generation*, we report the F1 score, defined as the harmonic mean of precision and recall. This metric is suitable for evaluating a set of predictions that are not ranked, as is commonly employed in the segmentation community [15]. As the output of this step is point cloud instance segmentation, we report results using both, the mask intersection-over-union (SegIoU) criterion, as well as 3D bounding box IoU (3DIoU) criterion for quantifying true positives and false negatives. This is important as it assesses how well we can recover labeled amodal bounding boxes that we need to use to train object detectors. For *Object detection*, the evaluate the performance using the standard Average Precision metric that assumes as input a ranked set of object detections. We focus on 3D bounding box IoU as an evaluation criterion. For pseudo-label generation as well as object detection, we report performance using localization thresholds of $T \leq \{0.7, 0.4\}$ with 0.7 being the standard evaluation threshold on Waymo Open dataset and 0.4 the threshold used in [24].

Data splits. Each approach for pseudo-labeling requires a certain percentage of labeled data to tune hyperparameters. We embrace a fully data-driven approach

to pseudo-labeling: given *some* labeled data, we train `SeMoLi` to localize moving objects in Lidar sequences. We then pseudo-label the unlabeled set of training data, which we use to train our *student* object detection network. This requires a careful evaluation protocol, outlined in Fig. 3. We split the official Waymo training set into fixed-size training (`train`) and validation (`val`) splits.

Val. As we need to train and validate *teacher* and *student* models independently, we split the `val` set into two equal-sized splits. We use `val_pseudo` for the validation of the *teacher* network, and `val_det` to independently validate the *student* (detection) network (Fig. 3).

Train. We split the `train` set into (labeled) training set `train_pseudo`, used to train the teacher network, then we utilize this network to pseudo-label the *student* training set `train_pseudo`. We use pseudo-labels to train the detection network. Note that we split the dataset along sequences, *i.e.* we do not allow frames of the same sequence to be in different splits.

Varying train set size. To study the amount of data needed to train our model, we sample varying-sized subsets for training the *teacher* and *student* networks, *i.e.*, `train_pseudo` and `train_det`. Consistent with prior work [24], we use the official Waymo Open *validation* for the end-model evaluation. This set provides labels and allows us to focus our evaluation on moving objects.

4.2. Ablations Studies

This section ablates the impact of different graph construction as well as parametrization strategies and bounding box inflation on `val_pseudo` dataset. Features based on velocity and position are defined as in Sec. 3.1.1.

Graph construction strategies: As constructing a fully connected graph is not feasible, we investigate different strategies to construct a k-nearest-neighbor (kNN) graph (Sec. 3.1.1) based on distance and velocity as similarity measures. In Tab. 1) we report the oracle performance for both, *i.e.*, the performance we achieve with each strategy assuming all edges are classified correctly. We base our design choice of utilizing position on its superior oracle performance (90.9@0.4 F1) (see appendix for deeper analysis).

Node features: As we show in Tab. 1, combining velocity and position in our node features significantly aids the learning process. Solely utilizing velocity lowers recall and slightly increases precision compared to position-based features, indicating an increased amount of rejected positive edge hypotheses due to possibly noisy velocity predictions. On the other hand, position as node features leads to a severe drop in precision and a slight drop in recall indicating that `SeMoLi` fails to correctly classify negative edges. Concatenating both, velocity enables `SeMoLi` to reject negative edges between points in close proximity if they do not move together and clusters points *that move together*.

Method		Pr 0.7	Re 0.7	F1 0.7	Pr 0.4	Re 0.4	F1 0.4
Graph	**Oracle** Velocity kNN	35.7	67.1	46.6	39.4	74.1	51.4
	Oracle Position kNN	85.0	87.9	86.4	89.4	92.5	90.9
Node f.	Velocity	61.2	52.7	57.0	73.0	62.2	67.1
	Position	57.6	61.7	59.6	65.2	69.9	67.5
	Velocity + Position	69.4	58.0	63.2	77.9	65.1	70.9
Edge f.	Velocity	58.9	48.6	53.2	69.6	57.4	62.9
	Position	69.4	58.0	63.2	77.9	65.1	70.9
	Velocity + Position	68.2	57.1	62.2	77.8	65.1	70.9

Table 1. **`SeMoLi` 10 ablation (SegIoU):** We discuss different strategies on `SeMoLi` **graph construction**, as well as **edge** and **node feature** parametrization.

	3DIoU			*SegIoU*		
	Pr 0.4	Re 0.4	F1 0.4	Pr 0.4	Re 0.4	F1 0.4
Initial	33.2	27.8	30.3	77.9	65.1	70.9
Inflated	59.1	48.3	53.1	80.9	66.2	72.8

Table 2. **Bounding box inflation:** We inflate tight bounding boxes that enclose point clusters to a minimum width, length, and height. The segmentation performance changes only insignificantly while the detection performance improves drastically. `SeMoLi` 10 clusters points together correctly, but generates bounding boxes that are significantly tighter around the objects.

Method	Pr 0.7	Re 0.7	F1 0.7	Pr 0.4	Re 0.4	F1 0.4
Oracle pseudo-label quality with ground truth flow / trajectories						
DBSCAN++	20.8	19.1	19.9	70.7	64.9	67.7
DBSCAN++†	20.6	19.1	19.8	72.0	66.2	69.0
SeMoLi 10	24.7	23.1	23.9	76.0	71.2	73.5
Pseudo-label quality with our computed flow / trajectories						
DBSCAN	0.9	5.7	1.5	6.0	39.3	10.4
DBSCAN†	1.5	5.8	2.4	9.9	38.9	15.8
DBSCAN++ [24]	1.4	6.1	2.2	8.9	39.9	14.5
DBSCAN++†	1.6	6.2	2.5	10.2	40.3	16.2
DBSCAN++$^†_l$	0.9	5.6	1.6	6.3	39.2	10.9
SeMoLi 10_{sf}	9.0	8.2	9.1	52.6	48.3	50.4
SeMoLi 10	10.1	8.2	9.1	59.0	48.3	53.1
SeMoLi 50	9.0	8.9	9.0	56.7	55.7	56.1
SeMoLi 90	8.8	9.1	9.0	56.9	58.4	57.6

Table 3. **Pseudo-label quality comparison (3DIoU):** We compare our `SeMoLi` to different variants of DBSCAN [24], augmented with scene flow (DBSCAN++), long-term trajectory information (DBSCAN++$_l$) and outlier filtering (†). In gray (*top*) we report results using ground truth scene flow and trajectories, and below we report scene flow and motion trajectories obtained using [37]. `SeMoLi` consistently performs favorably compared to all DBSCAN variants, when using perfect "oracle" motion cues, as well as when using the estimated (noisy) scene flow method.

Edge features: As in graph construction, we consider $L2$ distance between position- and velocity-based features, as well as the concatenation of both. As we show in Tab. 1, adding velocity to or completely omitting position from

	DBSCAN++[†]		SeMoLi [10]	
	3DIoU Re (Pr) 0.4	SegIoU Re (Pr) 0.4	3DIoU Re (Pr) 0.4	SegIoU Re (Pr) 0.4
Vehicle	36.7	70.1	54.5	76.9
Pedestrian	46.3	67.9	41.8	55.1
Cyclist	77.1	1.0	81.3	95.8
Class-agnostic	40.3 (10.2)	66.6 (16.8)	48.3 (59.1)	66.2 (80.9)
uFP	**72.0**		**14.5**	

Table 4. **Class-wise evaluation of pseudo-labels:** For class-wise evaluation, we assign GT classes to pseudo-labels that have *any* overlap GT. We additionally report the % unmatched false positives (uFP), *i.e.*, pseud-labels not matched to any GT box.

edge features harms the performance. Therefore, we use position-based encoding for edges.

Bounding Box Inflation We inflate our bounding boxes to a minimum length, width, and height and show the impact on the performance on Waymo Open Dataset in Tab. 2. While the performance based on SegIoU does not change significantly, evaluating the performance based on 3DIoU improves drastically. SeMoLi segments points correctly, but generates significantly more tight bounding boxes compared to ground truth (see appendix for details).

4.3. Evaluation of Pseudo-Label Quality

Baselines. We compare SeMoLi to three baselines: vanilla DBSCAN, DBSCAN++ (*cf*., [24]), and its variant DBSCAN++$_l$ that utilizes our (long-term) velocity-based motion feature for a fair comparison. [†] indicates heuristic filtering based on bounding box dimensions (detailed in the appendix). We also report variant of our method that utilizes local motion cues (scene flow), SeMoLi $_{sf}$.

Discussion. We discuss results in terms of 3DIoU, as the output of this step is used to train object detectors that assume amodal bounding boxes. In Tab. 3 (*top*) we report results we obtain with ground-truth motion information (oracle). In this setting, both DBSCAN (69.0@0.4 F1) and SeMoLi (73.5@0.4 F1) perform well. However, when utilizing, estimated motion information, all DBSCAN variants struggle with precision – even the variant with outlier removal is unable to surpass 16.2@0.4 F1. By contrast, SeMoLi can learn to filter noise and undergoes a significantly less severe performance drop as compared to its motion oracle. All SeMoLi variants surpass 50@0.4 F1, outperforming DBSCAN in terms of precision *and* recall.

Long-term motion cues. As can be seen, by contrast to DBSCAN, our SeMoLi [10] method can learn to utilize long-term motion cues, and performs favorably compared to the variant that relies on local motion estimates, SeMoLi $_{sf}^{10}$.

More data. Finally, we show SeMoLi benefits from the increased amount of training data. SeMoLi's performance monotonically increases with expanding training set, eventually reaching 57.6@0.4 F1.

	DBSCAN++[†]		SeMoLi [90]	
	3DIoU Re (Pr) 0.4	SegIoU Re (Pr) 0.4	3DIoU Re (Pr) 0.4	SegIoU Re (Pr) 0.4
Bicyclist	41.7	86.4	56.3	86.4
Box Truck	5.0	57.1	0	44.5
Bus	0	32.0	0.4	39.3
Large Vehicle	6.9	30.1	17.1	44.4
Motorcyclist	89.5	89.5	100	100
Pedestrian	29.7	59.8	42.5	57.1
Regular Vehicle	44.5	75.2	56.9	77.9
Stroller	0	0	0.5	0.5
Truck	0	97.1	4.4	94.1
Vehicular Trailer	3.5	14.0	0	14.0
Class-agnostic	33.3 (7.5)	59.3 (13.4)	45.8 (40.1)	65.2 (57.0)

Table 5. **Cross-dataset generalization:** We evaluate SeMoLi, trained on 90% labeled Waymo Dataset, on Argoverse2 dataset. Note that we never train our approach on Argoverse2. We merge Bicycle and Bicyclist as well as Motorcycle and Motorcyclist since they are not distinguishable by motion.

Per-class analysis. In Tab. 4 we discuss the per-class analysis (based on our oracle classifier, see Sec. 4.1). While we observe similar recall among SeMoLi and DBSCAN++[†], we note that the latter produces significantly more false positive pseudo-labels, quantified via %uFP, the percentage of pseudo-labels not matched to any ground truth object. We conclude that our approach produces a pseudo-label set that has a significantly better signal-to-noise ratio.

Cross-dataset evaluation. We further examine the ability of cross-dataset generalization by evaluating our model SeMoLi $_{90}$ on the Argoverse2 dataset. Importantly, we *never* our SeMoLi on Argoverse2, but solely on the train_pseudo split of Waymo Open dataset with $x = 90\%$. Argoverse2 contains finer-grained semantic labels compared to Waymo, which allows us to assess generalization to a wide range of classes. We note that Waymo and Argoverse2 are recorded with different, proprietary Lidar datasets, requiring our approach to generalize across sensors. In Tab. 5 we show, SeMoLi again consistently performs favorably compared to the DBSCAN baseline.

4.4. Object Detection

Finally, we utilize pseudo-labeled data to train an off-the-shelf object PointPillars (PP) object detector [19].

Experimental setting. As a baseline, we train PP in the standard setting with fine-grained semantic information (*class-specific*) as well as *class-agnostic*. The latter more closely resembles our setup, in which we do not have access to semantic information. Further, we train PP using all objects (*stat.* + *mov*), as well as moving-only (*mov-only*). We evaluate detectors on *all* objects (Tab. 6, *top*), as well as only on *moving* objects (Tab. 6, *bottom*, as reported in [24]). Additionally, we mix pseudo-labeled with 10% of labeled data (for finer-grained analysis see appendix) by re-using the train_pseudo splits (Sec. 4.1).

		% Pseudo	% GT	Pr 0.7	Re 0.7	AP 0.7	mAP 0.7	Pr 0.4	Re 0.4	AP 0.4	mAP 0.4
All (Mov. + stat.)	Stat. + Mov., class-specific	0	100	35.5	55.5	–	37.1	69.7	44.6	–	80.3
	Stat. + Mov., class-agnostic	0	100	31.8	41.2	36.1	–	51.2	66.5	64.7	–
	Mov-only, class agnostic	0	100	34.4	19.9	15.1	–	63.9	37.1	35.0	–
	DBSCAN++[†]	90	10	7.4	35.1	31.1	–	11.5	55.0	52.0	–
	DBSCAN++[†]	100	0	0.8	3.4	0.8	–	5.9	25.9	14.9	–
	SeMoLi	90	0	3.8	3.4	1.8	–	26.9	24.1	19.5	–
	SeMoLi	90	10	25.4	35.4	31.8	–	40.7	56.8	54.6	–
Moving only	Stat. + Mov., class-specific	0	100	30.5	34.5	–	43.2	36.4	41.1	–	85.6
	Stat. + Mov., class-agnostic	0	100	16.1	52.8	44.8	–	28.1	92.4	88.7	–
	Mov-only, class agnostic	0	100	33.9	53.7	44.3	–	57.6	91.2	89.0	–
	DBSCAN++[†]	90	10	2.0	34.5	29.8	–	4.1	70.1	61.0	–
	DBSCAN++[†]	100	0	5.9	9.7	2.4	–	3.6	58.6	43.2	–
	SeMoLi	90	0	3.8	10.7	4.2	–	23.3	66.0	57.5	–
	SeMoLi	90	10	9.4	35.4	31.7	–	19.7	74.6	66.2	–
	DBSCAN++ [24]	100	0	–	–	–	–	–	–	–	40.4

Table 6. **Semi-supervised 3D object detction on Waymo Open Dataset:** We evaluate models on **all** (*top*) and **only moving** (*bottom*) on Waymo Open validation set. % GT indicates the amount of labeled training data, % Pseudo indicates the amount of pseudo-labeled data.

		P%	L%	Pr 0.4	Re 0.4	AP 0.4
All	Labeled	0	100	52.3	39.3	35.5
	SeMoLi	100	0	12.2	30.7	22.9
Mov.	Labeled	0	100	45.1	85.5	82.4
	SeMoLi	100	0	8.0	64.7	57.6

Table 7. **Cross dataset results:** We train PP detector on ground truth data as well as on pseudo labels generated with SeMoLi trained on Waymo Open Dataset.

All objects. In Tab. 6 (*top*) we discuss results on *all*, *i.e.*, static and moving objects. We focus this discussion on 0.4 threshold. As can be seen, with *vanilla* PP detector, we obtain 80.3 mAP. In the class agnostic setting, we obtain 64.7 AP, which drops to 35 AP when only training with GT boxes labeled as moving. When using 10% of GT labels, we obtain 52.0 and 54.8 AP with DBSCAN and SeMoLi, respectively. Remarkably, when utilizing *any* pseudo-labels in conjunction with 10% labeled data, we obtain higher AP compared to the variant, trained with moving-only GT labels. This is likely because due to the noisy estimated flow, we retain *some* static objects. The moving-only GT version learns to only predict objects in regions where moving objects are likely to appear. Additionally, pseudo-labels may induce generalization to the training process. When not utilizing any labeled data, we obtain 14.9 and 19.5 AP, respectively. With SeMoLi, we, therefore, recover 56.6% performance of the variant, trained on GT moving-only labels.

Moving objects. When analyzing the performance on moving objects only in Tab. 6 (*bottom*), models, trained on pseudo-labeled bounding boxes, are significantly closer to fully supervised models (88.7 trained on all data, and 89.0 when trained with moving only). Utilizing no labeled data,

DBSCAN reaches 43.2 (48% of GT model), while SeMoLi reaches 57.5 (64% of GT model). When using 10% of labeled data, SeMoLi reaches 66.2 AP (64% of GT model). We note [24] reports 40.4 mAP on the validation set (last row), however, this is not an apple-to-apple comparison, as [24] does not provide implementation or detailed description of how the analysis was conducted. Moreover, it is unclear *how* [24] reports mAP in a class-agnostic setting.

Cross-dataset performance. Finally, in Tab. 7 we report results we obtain by training PP detector on Argoverse2 dataset. We compare the supervised detector (*labeled*) to SeMoLi, trained via pseudo-labels, generated on Argoverse2. Importantly, we generate pseudo-labels using SeMoLi trained on Waymo dataset, thus truly assessing cross-dataset generalization. When reporting results on *all* objects, we obtain 35.5 AP with the supervised model, and 22.9 with ours. For moving objects, we obtain 82.9 with the supervised model, and 57.6 with our approach, confirming our approach is indeed general and transferable across datasets. We detail the training in appendix.

5. Conclusions

We introduced SeMoLi, a data-driven, class-agnostic approach for pseudo-labeling moving objects in Lidar. We devised SeMoLi by revisiting correlation clustering in the context of message passing networks training our model to *learn* to decompose graphs constructed from point clouds. We utilized SeMoLi to pseudo-label data, as needed to train object detectors, and demonstrated that our data-driven approach performs favorably compared to prior art, and, more importantly, generalizes across datasets. By making our code, experimental data ad baselines publicly available we hope to inspire future efforts in this field of research.

References

[1] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine learning*, 56:89–113, 2004. 3, 4

[2] Jens Behley, Volker Steinhage, and Armin B Cremers. Laser-based segment classification using a mixture of bag-of-words. In *Int. Conf. Intel. Rob. Sys.*, 2013. 2

[3] 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 *CVPR*, 2020. 1

[4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 2

[5] C. Choy, J.Y. Gwak, and S. Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 2

[6] MMDetection3D Contributors. MMDetection3D: Open-MMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020. 5

[7] Ayush Dewan, Tim Caselitz, Gian Diego Tipaldi, and Wolfram Burgard. Motion-based detection and tracking in 3d lidar scans. In *Int. Conf. Rob. Automat.*, 2015. 4

[8] Bertrand Douillard, James Underwood, Noah Kuntz, Vsevolod Vlaskine, Alastair Quadros, Peter Morton, and Alon Frenkel. On the segmentation of 3d lidar point clouds. In *Int. Conf. Rob. Automat.*, 2011. 2

[9] 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 *Rob. Sci. Sys.*, 1996. 2, 3

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012. 1, 2

[11] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. 2017. 2, 3

[12] David Held, Jesse Levinson, Sebastian Thrun, and Silvio Savarese. Combining 3d shape, color, and motion for robust anytime tracking. In *Rob. Sci. Sys.*, 2014. 1

[13] David Held, Devin Guillory, Brice Rebsamen, Sebastian Thrun, and Silvio Savarese. A probabilistic framework for real-time 3d segmentation using spatial, temporal, and semantic cues. In *Rob. Sci. Sys.*, 2016. 1, 3

[14] Peiyun Hu, David Held, and Deva Ramanan. Learning to optimally segment point clouds. *IEEE Rob. Automat. Letters*, 5(2):875–882, 2020. 2

[15] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *CVPR*, 2019. 5

[16] Klaas Klasing, Dirk Wollherr, and Martin Buss. A clustering method for efficient segmentation of 3d laser data. In *Int. Conf. Rob. Automat.*, 2008. 2

[17] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 2

[18] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 5

[19] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 7

[20] Ze Liu, Zheng Zhang, Yue Cao, Han Hu, and Xin Tong. Group-free 3d object detection via transformers. 2021. 2

[21] Leland McInnes, John Healy, and Steve Astels. HDBSCAN: Hierarchical density based clustering. *J. Open Source Softw.*, 2(11):205, 2017. 2

[22] Frank Moosmann and Christoph Stiller. Joint self-localization and tracking of generic objects in 3d range data. In *Int. Conf. Rob. Automat.*, 2013. 1

[23] Frank Moosmann, Oliver Pink, and Christoph Stiller. Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion. In *Intel. Veh. Symp.*, 2009. 1

[24] Mahyar Najibi, Jingwei Ji, Yin Zhou, Charles R Qi, Xinchen Yan, Scott Ettinger, and Dragomir Anguelov. Motion inspired unsupervised perception and prediction in autonomous driving. In *ECCV*, 2022. 1, 2, 3, 4, 5, 6, 7, 8

[25] Lucas Nunes, Xieyuanli Chen, Rodrigo Marcuzzi, Aljosa Osep, Laura Leal-Taixé, Cyrill Stachniss, and Jens Behley. Unsupervised class-agnostic instance segmentation of 3d lidar data for autonomous vehicles. *IEEE Rob. Automat. Letters*, 7(4):8713–8720, 2022. 2

[26] Aljoša Ošep, Paul Voigtlaender, Jonathon Luiten, Stefan Breuers, and Bastian Leibe. Large-scale object mining for object discovery from unlabeled video. In *Int. Conf. Rob. Automat.*, 2019. 2

[27] A. Petrovskaya and S. Thrun. Model based vehicle detection and tracking for autonomous urban driving. 26:123–139, 2009. 2

[28] C.R. Qi, H. Su, K. Mo, and L.J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 1, 2

[29] C.R. Qi, L. Yi, H. Su, and L.J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv*, 2017. 1, 2

[30] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, 2017. 2

[31] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *CVPR*, 2019.

[32] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointR-CNN: 3D Object Proposal Generation and Detection From Point Cloud. In *CVPR*, 2019. 2

[33] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 1, 2, 5

[34] Alex Teichman and Sebastian Thrun. Tracking-based semi-supervised learning. *Int. J. Rob. Research*, 31(7):804–818, 2012. 1, 3

[35] Alex Teichman, Jesse Levinson, and Sebastian Thrun. Towards 3D object recognition via classification of arbitrary object tracks. In *Int. Conf. Rob. Automat.*, 2011. 1, 2

[36] Charles Thorpe, Martial Herbert, Takeo Kanade, and Steven Shafer. Toward autonomous driving: the cmu navlab. i. perception. *IEEE expert*, 6(4):31–42, 1991. 2

[37] Chaoyang Wang, Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural prior for trajectory estimation. In *CVPR*, 2022. 4, 6

[38] Dominic Zeng Wang, Ingmar Posner, and Paul Newman. What could move? Finding cars, pedestrians and bicyclists in 3D laser data. In *Int. Conf. Rob. Automat.*, 2012. 2, 3

[39] Xudong Wang, Zhaowei Cai, Dashan Gao, and Nuno Vasconcelos. Towards universal object detection by domain attention. In *CVPR*, pages 7289–7298, 2019. 3

[40] Max Wertheimer. Untersuchungen zur lehre von der gestalt: I. prinzipielle bemerkungen. *Psychologische forschung*, 1 (1):47–58, 1922. 1, 2, 3

[41] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Adv. Neural Inform. Process. Syst.*, 2021. 5

[42] Kelvin Wong, Shenlong Wang, Mengye Ren, Ming Liang, and Raquel Urtasun. Identifying unknown instances for autonomous driving. In *Conference on Robot Learning*, pages 384–393. PMLR, 2020. 2

[43] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 2

[44] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *CVPR*, 2018. 2

[45] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *CVPR*, 2020. 2

[46] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. In *CVPR*, 2021. 2

[47] Hao Zhang, Christopher Reardon, and Lynne E Parker. Real-time multiple human perception with color-depth cameras on a mobile robot. *IEEE Transactions on Cybernetics*, 43(5): 1429–1441, 2013. 1

[48] Lunjun Zhang, Anqi Joyce Yang, Yuwen Xiong, Sergio Casas, Bin Yang, Mengye Ren, and Raquel Urtasun. Towards unsupervised object detection from lidar point clouds. In *CVPR*, 2023. 2, 3, 5

[49] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018. 2