UNIT: Unsupervised Online Instance Segmentation through Time

Corentin Sautier[1,2] Gilles Puy[2] Alexandre Boulch[2] Renaud Marlet[1,2] Vincent Lepetit[1]
[1]LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, France [2]Valeo.ai, France

Abstract

Online object segmentation and tracking in Lidar point clouds enables autonomous agents to understand their surroundings and make safe decisions. Unfortunately, manual annotations for these tasks are prohibitively costly. We tackle this problem with the task of class-agnostic unsupervised online instance segmentation and tracking. To that end, we leverage an instance segmentation backbone and propose a new training recipe that enables the online tracking of objects. Our network is trained on pseudo-labels, eliminating the need for manual annotations. We conduct an evaluation using metrics adapted for temporal instance segmentation. Computing these metrics requires temporally-consistent instance labels. When unavailable, we construct these labels using the available 3D bounding boxes and semantic labels in the dataset. We compare our method against strong baselines and demonstrate its superiority across two different outdoor Lidar datasets. Project page: csautier.github.io/unit

1. Introduction

Being able to segment, classify, and track all object instances in sequences of point clouds acquired by Lidars is critical for many applications, including autonomous driving. This task, called (Multi-Object) Panoptic Tracking [11, 13, 17], can be seen as a temporal version of panoptic segmentation.

Unfortunately, the cost of annotating point cloud data for this task can be enormous [3]. Moreover, to ensure high performance and robustness, a dataset should contain enough examples of rare objects (e.g., one-wheelers, electric scooters) or hard settings (e.g., adverserial weather), increasing the cost of annotation further. For systems with an international scope, such as autonomous driving, there is also a need to address domain shifts and cover different object distributions (e.g., shape variations [33]) and local specificities (e.g., rickshaws). One way to reduce the need for annotations is to rely on self-supervised [4, 22, 23, 26, 28, 29] or unsupervised learning techniques [2, 21, 31, 38].

In this work, we introduce a new fully unsupervised method, which we call UNIT, to segment class-agnostic object instances and track them over time, regardless of their nature or semantics. More exactly, our goal is to densely segment an input point cloud so that each input point is assigned to one and only one instance ID.

As illustrated in Fig. 1, given unlabeled Lidar sequences, we first obtain pseudo 4D segments of objects by using a spatio-temporal clustering. Our clustering is inspired by SegContrast [22] and TARL [23], but we improve these methods to obtain clusters over a longer time window. This gives us 4D segments of objects which we then use as pseudo-labels to train a novel auto-regressive architecture, which, at inference, identifies and track objects in new Lidar sequences by being applied to consecutive scans.

To evaluate our method, we test it on two main datasets, SemanticKITTI [3] and PandaSet-GT [35], which exhibit different point densities, scan patterns and environments. We use several metrics to analyze both the segmentation and temporal tracking quality of our method. We compare our results against strong baselines as well as temporally-extended versions of 3D instance segmentation techniques. Finally, we also show the potential of UNIT as a self-supervised pretraining technique for semantic segmentation.

2. Related work

Object detection/segmentation with transformers. Our network architecture relies on a transformer decoder similar to the one used in DETR [6] for object detection, or in [7, 16, 30] for object segmentation. A set of learnable object queries are provided at the input of a transformer decoder, which is in charge of detecting/segmenting objects in the image or point cloud of interest thanks to a series of cross-attention layers between the object queries and pixel or point features. The transformer is trained so that each query attends to at most one object instance and each object instance is matched to a single query. By default, these methods do not allow objects to be tracked over time.

Object detection and tracking with transformers. To detect and track objects in videos, [17, 19, 32, 37] extend DETR [6] using two sets of object queries: a first set of queries is in charge of tracking discovered objects in past scans, while a second one is in charge of discovering new objects. In contrast, we propose an architecture that detects and tracks objects with a single set of queries.

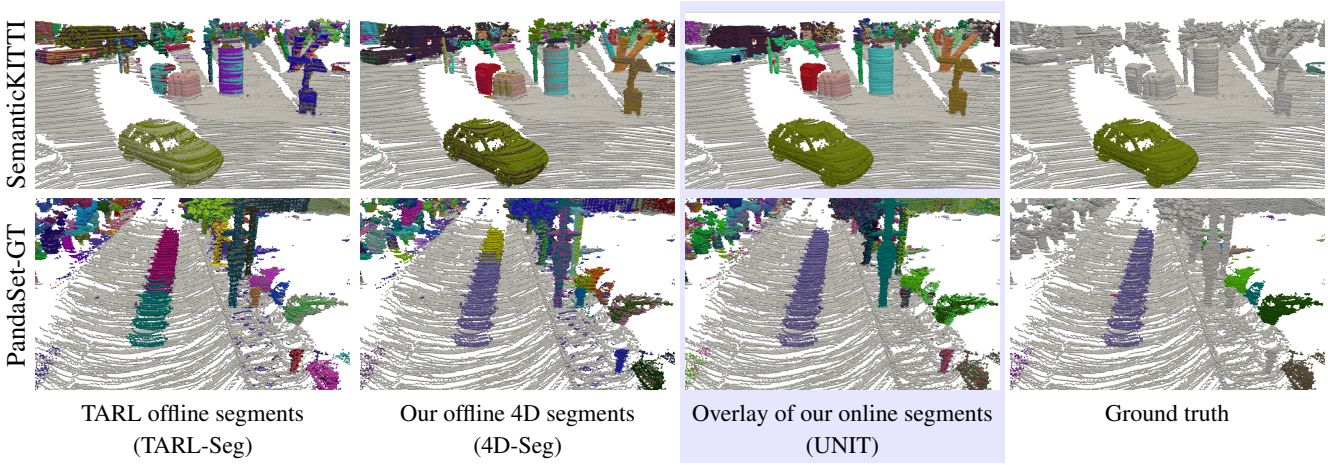

Figure 1. We generate pseudo-labels of instances across time in Lidar scans (4D-Seg), which we use to train an online segmenter (UNIT) that assigns to each 3D point an instance ID consistent over time. UNIT is the only online method displayed here (our input is *not* an aggregated scan). We show aggregations of scans over time. The sample scene from Semantic-KITTI, with the car in the foreground, is mostly static, while the scene from PandaSet-GT is dynamic, with the trace of a moving vehicle.

Note that we obtain more labels than in the ground truth as our class-agnostic segmentation include all objects and stuff, such as trees or buildings. Also, while our architecture performs online, it outperforms the offline spatio-temporal clustering that we use for training.

Unsupervised instance segmentation. UnScene3D [27] performs unsupervised instance segmentation in dense point clouds acquired in static indoor scenes. A Mask3D model [30] is trained using pseudo-masks obtained by leveraging self-supervised features and a graph-cut algorithm. Because of the static setting considered, UnScene3D does not require any temporal consistency in predictions.

SegContrast [22] and TARL [23] are self-supervised methods whose pretext tasks rely on pseudo-masks of objects. SegContrast uses a RANSAC [10] algorithm to remove the road plan of Lidar scans, and then a DBSCAN clustering [9] to obtain pseudo-segments of instances. TARL extends SegContrast by working on accumulated scans, replacing RANSAC with Patchwork [15], a better-performing ground segmentation method, and using HDBSCAN instead of DBSCAN [18], which gives better segments especially in low-density regions of the point cloud. We compute our pseudo-segments by building upon TARL (see Sec. 3.1).

3DUIS [21] uses a self-supervised model trained with SegContrast to define affinity in a graph between points, and a graph-cut algorithm to get object instances. [21] also propose the $\mathrm{S}_{\mathrm{assoc}}$ metric for evaluating unsupervised segments.

AutoInst [25], like 3DUIS, obtains object segments using a graph-cut algorithm. This graph is built by exploiting TARL [23] point features (possibly complemented with DINOv2 [24] features) instead of SegContrast [22] features. A notable difference between AutoInst and 3DUIS is that AutoInst operates on accumulated scans instead of single scans. These segments are then used as pseudo-labels to train a MaskPLS [16] model, which improves the quality of predicted segments. This MaskPLS model, like the graph-cut algorithm, operates on accumulated scans. Unlike AutoInst, UNIT operates on single scans instead of accumulated scans, in a online fashion. At inference time, contrary to AutoInst, UNIT does not need odometry information for accumulation. UNIT also does not need to leverage pretrained models to obtain pseudo-labeled objects segments. Besides, AutoInst, only evaluates on SemanticKITTI where sequences 01 and 04 are removed from the training set because they predominantly consist of dynamic instances. In contrast, with UNIT, there are no requirements limiting the number of dynamic instances for training. Last, the code of AutoInst is not publicly available, which prevents full-fledged comparisons.

OYSTER [38], SeMoLi [31] and LISO [2] use motion cues to find moving objects and regress boxes around them, which are used to train an object detector unsupervisedly. While some aspects are similar to our work, they focus on moving objects and object detection, while we target all objects and instance segmentation. Besides, they do not use a temporal architecture.

3. Method

Fig. 2 provides an overview of our approach: given a sequence of Lidar scans, we train an online instance segmenter that labels each 3D point with an ID for the object to which it belongs, while maintaining consistent IDs over time: two points from different scans sharing the same ID belongs to the same instance. The online, temporal processing is performed by a novel auto-regressive architecture that successively takes individual Lidar scans as input and labels points as scans come in (without knowing the future).

We train this network without any human supervision. To that end, we first create pseudo-labels on unannotated Lidar sequences using a spatio-temporal clustering. As it

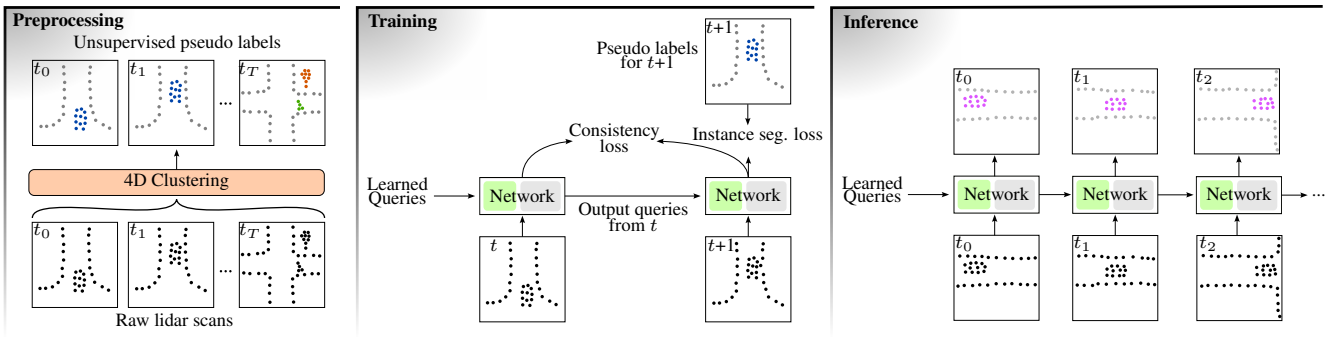

Figure 2. **Method overview.** Given unlabeled Lidar scans, we create offline pseudo-labels by spatio-temporal clustering (preprocessing). We then use these pseudo-labels to train an auto-regressive network. At inference, we apply this network to successive scans as they come.

is unsupervised and incorporates very few priors beyond 4D consistency, our method makes no distinction between "thing" objects (e.g., cars, pedestrians) and "stuff" objects (e.g., vegetation, buildings), whose separation may anyway vary from one dataset to another (e.g., for traffic signs).

We detail each step of our approach below.

3.1. Pseudo-labeling via spatio-temporal clustering

Road segmentation. As experimentally verified in prior work [22, 23, 38], a good strategy for segmenting objects in automotive Lidar scans is to first remove the ground, as it makes the objects stand out as isolated clusters that are more easily separable. As the ground can often be assimilated to a large, mostly horizontal plane, RANSAC [10] can be used to separate this plane from the rest of the scan. However, as they take into account the specific patterns of Lidar point clouds, recent alternative methods [14, 15] obtain better performance on ground segmentation, while remaining unsupervised.

We rely on these methods to segment the ground. To find the ground points, we used Patchwork [15] on SemanticKITTI (as in TARL [23]), and Patchwork++ [14] on PandaSet-GT. Both methods need to be applied scan-wise on each Lidar scan of the training set.

Spatio-temporal segmentation. The authors of TARL [23] demonstrate that applying HDBSCAN [18] on temporally accumulated scans enables the extraction of spatially and temporally coherent object instances, even for dynamic objects. Yet, the complexity of HDBSCAN restricts the application of this clustering to a short time window. In practice, TARL [23] aggregates 12 successive scans of SemanticKITTI, and thus provides pseudo-labels over a temporal window of 1.2 s as the Lidar operates at 10 Hz. To increase the temporal context window, we apply a voxel grid sampling to heavily reduce the number of points to cluster after temporal aggregation. More details are given in the supplementary material. With this process, we are able to obtain time-coherent clusters over 40 aggregated scans. We will see in Sec. 4 (Tab. 1) that our clusters have the same quality as the clusters obtained in [23] when evaluated on a per-scan basis, but that their temporal consistency is of a much better quality.

3.2. Online instance segmentation network

Our segmenter architecture is based on Mask3D [30], which itself is an adaptation of MaskFormer [7] for 3D point clouds. We recall below how these methods work (using object-based queries) and explain what modifications we bring to make them work in a class-agnostic and temporal (online) setting. More details are provided in appendix.

Class-agnostic object segmentation. Mask3D uses a transformer decoder to segment objects. This decoder attends to point features and produces one embedding for each input query, with a fixed number of queries $|Q|$. Each query $q \in Q$ is in charge of segmenting one object via a mask module: the scalar product between each query embedding and all point features provide an instance heatmap, which can be transformed into a object-wise binary mask, e.g., via thresholding. The modifications we bring to Mask3D are threefold.

First, we have to take into account that, in our unsupervised setting, we have no access to the object classes. We therefore do not use any semantic classification head on top of the query embeddings.

Second, to densely segment the whole input point cloud, each input point must be assigned to a mask ID, and thus to a query. To do so, at inference, after computing all scalar products $\langle \text{feat}(p), q \rangle$ between the point features and the query embeddings, each point p is assigned the ID of the query that responds the most to the point features:

$$\text{ID}(p) = \text{ID}(\text{query}(p)) \tag{1}$$

$$\text{query}(p) = \arg\max_{q \in Q} \langle \text{feat}(p), q \rangle \,. \tag{2}$$

(There is no thresholding here as it could leave points without any assigned ID.) Queries to which at least one point is assigned are called active queries. During training, this hard assignment is not applied and the loss is directly computed on the results of the scalar products (cf. Sec. 3.3).

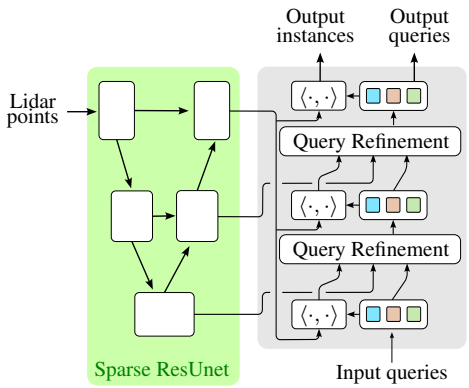

Figure 3. Mask3D [30] architecture. The queries are trained to favor attending to at most one object in the Lidar point cloud (Sec. 3.3). We adapt this architecture for online segmentation and tracking of objects in Lidar sequences by forwarding output queries to another instance of the network, in an auto-regressive manner (see Fig. 2).

Finally, we wish to adapt the queries to follow moving objects. To that end, we consider an auto-regressive architecture where the queries at network output are used again as input to the next network inference (see below).

Auto-regressive extension. Mask3D's transformer decoder refines the object queries progressively by attending at point features output at different resolutions (cf. Fig. 3). After each refinement step, each query increasingly specializes to a specific object.

To allow the tracking of each instance through time, we inject the query embedding at time t (output of the transformer decoder) as input to the transformer decoder at time t+1. As the encoder and decoder share the same weights at each time step, each query at time t+1 should therefore continue to attend to the same object that it was already attending to at time t. At least in principle, this architecture enables the tracking of objects over an infinite time window. This recursion is illustrated in Fig. 2 (training, inference).

As input to the first network application of this auto-regressive scheme (see input queries at bottom of Fig. 3), we use fully-learnable initial queries. This flexibility, which is in line with [6–8] but unlike the non-parametric queries of [20, 30], prepare for the recursive use of the queries.

3.3. Training loss

Our training loss is made of two main terms. The first term is a scan-wise training loss, similar to the one used in Mask3D [30]. We detail its construction in our class-agnostic setup below. The second loss term favors a time-consistent segmentation.

We denote the number of points, queries and pseudo ground-truth instances by N_p, N_q and N_o, respectively. The loss is computed based on an affinity score A_{ij}, which is the scalar product between the query-embeddings $j = 1, \ldots, N_q$ and the point features $i = 1, \ldots, N_p$, followed by a sigmoid. We also denote by $G \in [0,1]^{N_p \times N_o}$ the assignment matrix between the points and the objects as computed from the pseudo-labels: $G_{io} = 1$ if point i belongs to object o according to the clustering, and 0 otherwise.

Scan-wise training loss. To compute the scan-wise loss, we need to find correspondences between queries and object instances. These correspondences are computed by minimizing a cost of assignment between them. The global cost matrix $C \in \mathbb{R}^{N_q \times N_o}$ between the queries and the object instances is defined as:

$$C = \lambda_{\text{dice}}\, C^{\text{dice}} + \lambda_{\text{BCE}}\, C^{\text{BCE}}\,, \tag{3}$$

where C^{dice} is based on the Dice loss,

$$C^{\text{dice}}_{jo} = \frac{2 \sum_{i=1}^{N_p} A_{ij} G_{io}}{\sum_{i=1}^{N_p} {A_{ij}}^2 + \sum_{i=1}^{N_p} {G_{io}}^2}\,, \tag{4}$$

and C^{BCE} is based on the binary cross entropy,

$$C^{\text{BCE}}_{jo} = \sum_{i=1}^{N_p} G_{io} \log(A_{ij}) + (1 - G_{io}) \log(1 - A_{ij})\,. \tag{5}$$

We use $\lambda_{\text{dice}} = 2$ and $\lambda_{\text{BCE}} = 5$, as in [30]. Unlike in [30], we do not have any information about the semantic class of the objects in our unsupervised setting. We therefore do not use any classification loss related to semantic categories. The actual matching between object instances o and queries j is obtained using the Kuhn–Munkres algorithm (aka Hungarian matching) with C as cost matrix. Once this matching $m(\cdot)$ is obtained, we minimize

$$\sum_{o=1}^{\min(N_q, N_o)} \lambda_{\text{dice}}\, C^{\text{dice}}_{m(o)o} + \lambda_{\text{BCE}}\, C^{\text{BCE}}_{m(o)o}\,, \tag{6}$$

where $m(o)$ is the index of the matched query for object o. Finally, this process is repeated identically at 12 different layers and scales of the ResUNet, and the final loss is a sum of all corresponding loss computations, as in [30].

Time consistency loss. We aim at being able to segment and track object instances for a time period much longer than the time window seen during training. In fact, we can only train our network on two consecutive scans on our resources. To enforce time-coherent segmentation over longer periods of time, we propose the consistency loss described below.

Let us consider an object instance o segmented at time t and t+1 with our pseudo-labeling strategy. For instance o and for each query q, we compute the average similarity score (scalar product between query embedding and point features) with the points belonging to o at time t and t+1. We then transform these scores as a probability of assignment over

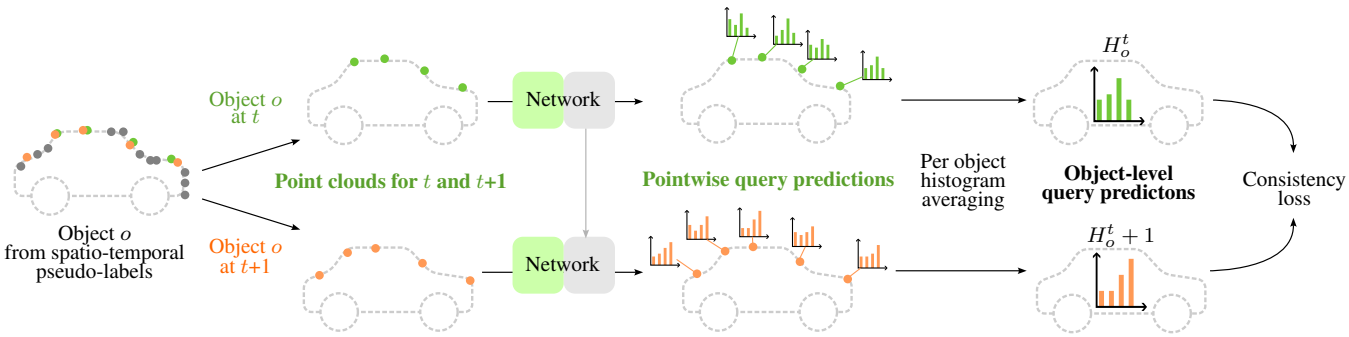

Figure 4. **Computing the consistency loss**. Given an object, we compute the averages H_o^t and H_o^{t+1} of its pointwise query predictions over two time steps. The consistency loss is the cross-entropy between H_o^t and H_o^{t+1}.

the queries using a softmax layer (over the dimensions of the queries). We obtain two probability distributions for each object o: $H_o^t \in [0,1]^{N_q}$ and $H_o^{t+1} \in [0,1]^{N_q}$. To encourage temporal continuity, we minimize the relative entropy, i.e., the Kullback-Leibler divergence from H_o^{t+1} to H_o^t

$$\mathcal{L}_o^{\text{cons}} = D_{\text{KL}}(H_o^t || H_o^{t+1}) \,. \tag{7}$$

We consider H_o^t to be the target distribution and do not back-propagate through it (thanks to a stop-gradient operation). However, note that back-propagation goes through the two timesteps thanks to the auto-regressive architecture. In practice, an equivalent gradient is obtained with a cross-entropy loss between H_o^{t+1} and H_o^t, while keeping the stop-gradient operation on H_o^t.

Global loss. We first pretrain our network in a single scan setting, using the scan-wise loss function:

$$\frac{1}{N} \sum_{o=1}^{N} \left[\lambda_{\text{dice}} \, C_{m(o)o}^{\text{dice}} + \lambda_{\text{BCE}} \, C_{m(o)o}^{\text{BCE}} \right] \,, \tag{8}$$

with $N = \min(N_q, N_o)$. We then train on two consecutive scans by adding the time-consistency loss between the first and second scans:

$$\frac{1}{N} \sum_{o=1}^{N} \left[\lambda_{\text{dice}} \, C_{m(o)o}^{\text{dice}} + \lambda_{\text{BCE}} \, C_{m(o)o}^{\text{BCE}} + \lambda_{\text{cons}} \mathcal{L}_o^{\text{cons}} \right] \,. \tag{9}$$

In this second phase, costs C^{dice} and C^{BCE} are computed only on the second scan. Thus, the Hungarian matching used to compute mapping $m(\cdot)$ uses only the query embedding at the second scan. We used $\lambda_{\text{cons}} = 1$ in our experiments.

Further details about the architecture and training procedure are provided in appendix.

3.4. Query recycling mechanism

The number of queries in our architecture is fixed. In our experiments, it is set to $N_q = 300$, which off-the-shelf allows the segmentation and tracking of at most 300 different instances over an entire sequence. This is far lower than the number of objects in a sequence of thousands of scans. We need a process to handle the appearance of new objects and their disappearance (when they go out of sight) with a query recycling mechanism.

Our recycling process is based on a simple heuristics. After each scan at a given time t is processed, we compute the barycenter of the points associated to every active query. Then for each active query q, we retrieve its previous barycenter, at t_{past}. At this stage, we need a simple rule to determine if query q is still tracking the same object or tracking a new object. We do so by checking the distance between the two barycenters for q at t and t_{past}. If this distance is smaller than 10 m, then we assume the query is still tracking the same object; otherwise it is tracking a new object, which was never seen in the past. In this case, we assign a new object ID to the query.

Note that the barycenters are computed in the coordinate system of the ego vehicle. We compute the distance between barycenters *without* compensating for the motion of the ego-vehicle between time t_{past} and t, which means we do not need odometry information nor a side method to register the scans. Our criterion simply checks if, relative to the ego-vehicle, the segmented object appears in a completely different area than where it was before. This criterion is imperfect. For instance, a new object could be detected by the same query as an old object if it were appearing at a location close to the position of the old object relative to the ego-vehicle. While this strategy still works reasonably well in practice, it leaves room for improvement.

4. Experiments

4.1. Datasets

SemanticKITTI [3] is a Lidar dataset acquired in Germany, with a 64-beam Lidar capturing scans at 10 Hz. We use the usual split where sequences 00 to 10 but 08 are used for (unsupervised) training, and sequence 08 is used for validation. On the validation sequence, our metric are computed

on object instances of the following classes: car, bicycle, motorcycle, truck, other-vehicle, person, bicyclist, and motorcyclist. This dataset contains the longest sequences, with the validation sequence consisting of 4071 consecutive scans.

PandaSet [35] contains data captured with two different Lidar sensors, a rotating Pandar64 and a solid-state PandarGT, at 10 Hz. Following [26], we use GPS coordinates to create two spatially separated training and validation sets, using scans collected in San Francisco for training and the rest as the validation set. PandaSet does not contain instance ground truth. We create them by combining semantic segmentation and object detection labels in a process described in the supplementary material. In this work, we only experiment with the dense point clouds captured with PandarGT, as Pandar64 has similar characteristics to the Velodyne HDL-64E used in SemanticKITTI. We denote this set by PandaSet-GT.

4.2. Baselines

We consider six different baselines. The first three baselines are 3DUIS [21], the unsupervised spatio-temporal segmentation used in TARL [23] (TARL-Seg), and our technique for generating pseudo-labels (dubbed 4D-Seg). All of these baselines are able to segment and track over a limited time window. We thus propose an improved version of them by post-processing the results to find correspondences between instances obtain over two successive time windows. Note that, unlike UNIT, none of these baselines are online, except 3DUIS, which does not provide spatio-temporal segments.

3DUIS [21] is, to the best of our knowledge, the only published method performing unsupervised instance segmentation on automotive Lidar point clouds. This method is working scan-wise and obtains instances by solving a graph-cut problem. As solving this graph-cut problem already is computationally intensive in the scan-wise setting, we do not apply graph-cuts directly in the spatio-temporal setting. Instead, we compare this method to ours in a scan-wise setting but also use the post-processing described below to turn the discovered 3D segments into 4D segments.

TARL's segments and our 4D segments are obtained using classical clustering (HDBSCAN [18]) on scans accumulated after rigid registration (see Sec. 3.1). Both methods provide spatio-temporal segments over a predefined time-window, unlike our method UNIT which is online. Note as well that we do not need to register the scans in UNIT. The clustering used in TARL was tuned for SemanticKITTI, with a time window corresponding to 12 consecutive scans. Our 4D segments provides object instances over 40 consecutive scans on both SemanticKITTI and PandaSet-GT.

Baseline improvement. All of the above methods provide 4D segments over a fixed time window. By construction, they are unable to track an object over a temporal horizon longer than this time window, which impacts negatively their performance relatively to the metrics defined in Sec. 4.3.

We improve the performance of these baselines by matching instances discovered in two successive time-windows. Given the last scan ℓ in a reference time window and the first scan f in the next time window, we register these scans in the same coordinate system and compute the convex hull of all instances in ℓ and f. Whenever the convex hulls of two instances from different scans overlap with an IoU larger than 0.5, the object instance in f is assigned the object ID of the instance in ℓ. Note that we use the ground-truth poses of the ego-vehicle only for these baselines—UNIT does not need access to the vehicle poses. More details are provided in the supplementary material. We denote the improved baselines by 3DUIS++, TARL-Seg++, and 4D-Seg++.

4.3. Metrics

Association score. Our main metric is the association score used in [1] for 4D panoptic segmentation in Lidar data, and used in [21] for unsupervised class-agnostic instance segmentation. We recall its definition below for completeness.

Let $\mathcal{G}$ be the set of manually-annotated objects (sets of points) serving as ground truth for evaluation but unused during training, and $\mathcal{S}$ be the set of segments (objects) predicted by our network. Note that any object in $\mathcal{G}$ (similarly for $\mathcal{S}$) is a *4D segments* containing the list of points belonging to that object at every timestep of the sequence. The temporal association score $\mathrm{S}_{\mathrm{assoc}}^{\mathrm{temp}}$ satisfies:

$$\mathrm{S}_{\mathrm{assoc}}^{\mathrm{temp}} = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \frac{1}{|g|} \sum_{\substack{s \in S \\ s \cap g \neq 0}} \mathrm{TPA}(s,g) \mathrm{IoU}(s,g) \,, \quad (10)$$

where $\mathrm{TPA}(s,g) = |s \cap g|$ is the number of true positive associations between the 4D segments s and g, and IoU is the intersection-over-union.

A non-temporal version of the association score can be computed by considering that objects and predictions in different scans come from different instances. This version of the association metric, denoted $\mathrm{S}_{\mathrm{assoc}}$, is the metric used in [21] as this method works in a scan-wise setting.

Best IoU. In complement to the above score, we introduce another metric where, for each object in $\mathcal{G}$, we first find the best overlapping segments in $\mathcal{S}$ in terms of IoU, and then compute the average IoU with these best matches:

$$\mathrm{IoU}^* = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \max_{s \in S} \mathrm{IoU}(s,g) \,. \quad (11)$$

Note that this metric is computed using 4D segments: the higher the score, the better the temporal consistency. Note as well that, in this metric, a predicted segment s can be a best match for two different ground-truth objects g and

Sem.KITTI	Online	Unfiltered			Filtered	
		S^{temp}_{assoc}	IoU*	S_{assoc}	S^{temp}_{assoc}	S_{assoc}
3DUIS w/o time	✓	-	-	0.550	-	0.768
UNIT w/o time	✓	-	-	**0.715**	-	**0.811**
3DUIS++	✓	0.116	0.214	0.550	0.148	0.769
TARL-Seg	✗	0.231	0.353	0.668	0.264	0.735
TARL-Seg++	✗	0.317	0.446	0.617	0.370	0.678
4D-Seg	✗	0.421	0.529	0.667	0.486	0.784
4D-Seg++	✗	0.447	0.513	0.647	0.512	0.762
UNIT	✓	**0.482**	**0.568**	0.696	**0.563**	0.790

Table 1. **Results on SemanticKITTI.** All scores are computed on the validation set of SemanticKITTI. The association scores are computed using the code of [1], which, by default, is only applied on segments of more than 50 points for any given scan; we report the corresponding scores with and without filtering.

PandaSet-GT	Online	Unfiltered			Filtered	
		S^{temp}_{assoc}	IoU*	S_{assoc}	S^{temp}_{assoc}	S_{assoc}
UNIT w/o time	✓	-	-	**0.562**	-	0.719
TARL-Seg	✗	0.206	0.286	0.369	0.390	0.757
4D-Seg	✗	**0.332**	**0.399**	0.492	**0.503**	**0.740**
UNIT	✓	0.209	0.310	0.531	0.351	0.688

Table 2. **Results on PandaSet-GT** (validation set, see Sec. 4.1).

g' ($g \neq g'$). It thus provides a measure of the best IoU we can reach with the available set predictions.

Limitations. The above metrics require 4D manual segments of each object. In existing datasets, such annotations are available only for *thing* classes (cars, pedestrians, etc.). Thus, while our method is able to segment and track all objects (both *stuff* and *things*), these metrics are only used to measure performance on a subset of the objects.

Filtered metrics. The metrics used by [21] and [1] include a scan-wise filter removing all 3D ground-truth segments with fewer than 50 points. We include these "filtered" metrics for comparability purposes. We would like to recall nevertheless that this filtering removes small or distant objects from the metrics. We thus also compute the metrics without any filtering to provide a more complete assessment of the performance of the methods.

4.4. Results of UNIT

On SemanticKITTI. We compare our method to all the considered baselines on SemanticKITTI. The results are presented in Tab. 1. In the single-scan setting, UNIT is better than 3DUIS. This shows the advantage of training end-to-end an instance segmenter rather than relying on a graph-cut method. In the online setting, we notice that post-processing

Training			Inference		Metric	
Scan-wise	Autoreg.	$\mathcal{L}^{cons}_{o}$ (7)	Scan-wise	Autoreg.	S^{temp}_{assoc}	S_{assoc}
4D-Seg					0.420	0.667
✓	-	✗	✓	-	-	0.715
✓	-	✗	-	✓	0.330	0.667
-	✓	✗	-	✓	0.446	0.727
-	✓	✓	-	✓	0.482	0.696

Table 3. **Ablation study on the validation set of SemanticKITTI.** Influence of the training and inference procedure (scan-wise or auto-regessive) as well as the effect of the loss (7) on the performance.

the segments obtained by 3DUIS is not enough to compete with the segments obtained by TARL-Seg and 4D-Seg on both temporal metrics S^{temp}_{assoc} and IoU*. We also remark that 4D-Seg performs better than TARL-Seg. This is due to the longer temporal window used in 4D-Seg as both methods give similar scores on the scan-wise metric S_{assoc}. We also observe that our post-processing over the segments of TARL-Seg and 4D-Seg improve the spatio-temporal quality of the segments. The best scores are obtained with our method.

On PandaSet-GT. We notice in Tab. 2 that UNIT surpasses TARL-Seg, except when filtering out all small ground-truth segments. Unlike on SemanticKITTI, UNIT does not outperforme its pseudo-labels (4D-Seg). The large context window (4 s) for 4D-Seg with respect to sequence duration (8 s) on PandaSet-GT provides an advantage to 4D-Seg which has access to nearly the whole sequence for segmentating, while UNIT works online. We also recall that UNIT does not require precise registration of the scans, unlike the baselines.

4.5. Ablation study

We conduct an ablation study on the components of UNIT on SemanticKITTI. The results are reported in Tab. 3.

First, we remark that, surprisingly, training our network in a scan-wise setting but doing the inference in an auto-regressive manner (second row of the table) already provides reasonable segmentation of instances in 4D, with scores not so far from those of 4D-Seg. This indicates that using as input the query embedding of the past scan already allows to track some object instances without explicit supervision. This justifies the auto-regressive use of the queries.

Second, as seen from the third row of Tab. 3, training the network in an auto-regressive manner permits us to surpass the 4D-Seg baseline (our pseudo-labels). Note that, in this case, there is no explicit constraint or regularization that favors the discovery of spatially and temporally consistent instances. This behavior is only achieved implicitly via the auto-regressive use of the query embeddings and the use of spatio-temporal pseudo-labels. Yet, these two ingredients are enough to surpass 4D-Seg.

nuScenes	Online	Unfiltered			Filtered	
		S^{temp}_{assoc}	IoU*	S_{assoc}	S^{temp}_{assoc}	S_{assoc}
UNIT w/o time	✓	-	-	**0.390**	-	**0.570**
TARL-Seg	✗	0.085	0.156	0.189	0.253	0.475
4D-Seg	✗	**0.163**	**0.246**	0.287	**0.350**	0.546
UNIT	✓	0.126	0.221	0.356	0.270	0.561

Table 4. **Results on nuScenes** (validation set).

	Pre-training	Inputs	Percentage of training data			
			1%	10%	50%	100%
nuScenes	No pre-training	xyz + i	35.0	57.3	69.0	71.2
	PointContrast [36]	xyz + i	37.0 +2.0	58.9 +1.6	69.4 +0.4	71.1 -0.1
	ALSO [4]	xyz + i	37.4 +2.4	**59.0 +2.7**	69.8 +0.8	71.8 +0.6
	BEVContrast [29]	xyz + i	**37.9 +2.9**	**59.0 +2.7**	**70.5 +1.5**	**72.2 +1.0**
	No pretraining	depth + i	35.4	60.2	71.3	71.3
	UNIT w/o time	depth + i	35.3 -0.1	60.9 +0.7	**72.2 +0.9**	**74.0 +2.7**
	UNIT	depth + i	35.4	**61.9 +1.7**	72.0 +0.7	73.6 +2.3
SemanticKITTI	No pre-training	xyz + i	46.2	57.6	61.8	62.7
	PointContrast [36]	xyz + i	47.9 +1.7	59.7 +2.1	62.7 +0.9	63.4 +0.7
	SegContrast [22]	xyz + i	48.9 +2.7	58.7 +1.1	62.1 +0.3	62.3 -0.4
	STSSL [34]	xyz + i	49.4 +3.2	60.0 +2.4	62.9 +1.1	63.3 +0.6
	ALSO [4]	xyz + i	50.0 +3.8	60.5 +2.9	**63.4 +1.6**	63.6 +0.9
	TARL [23]	xyz + i	52.5 +6.3	61.2 +3.6	**63.4 +1.6**	63.7 +1.0
	BEVContrast [29]	xyz + i	**53.8 +7.6**	**61.4 +3.8**	**63.4 +1.6**	**64.1 +1.4**
	No pretraining	depth + i	46.6	59.8	62.2	63.4
	UNIT w/o time	depth + i	49.0 +2.4	61.8 +2.0	**64.9 +2.7**	65.1 +1.7
	UNIT	depth + i	**50.2 +3.6**	**62.4 +2.6**	**64.9 +2.7**	**65.9 +2.5**

Table 5. **Fine-tuning for semantic segmentation.** We show the quality (mIoU% after fine-tuning) of the features we learn self-supervisedly. Other methods, in grey, have a similar training setup but a different network architecture and inputs. 'i' is for intensity.

Finally, adding the time consistency loss further boosts the performance on the spatio-temporal metrics.

4.6. Case of low density point clouds

Our method, like [21–23], relies on the extraction of accurate segments for pseudo-labeling. Such a segmentation is far easier when starting from dense point clouds, as available in SemanticKITTI and PandaSet-GT, than when starting from low-density point clouds. This probably explains why related methods do not conduct experiments on datasets like nuScenes, where point clouds are much sparser than in SemanticKITTI. To test the limits of our method and baselines on low-density point clouds, we apply them on nuScenes [5] (see description in Appendix).

As on the other datasets, we notice (cf. Tab. 4) that 4D-Seg surpasses TARL-Seg thanks to its longer temporal window. UNIT performs better than 4D-Seg on the non-temporal metric S_{assoc} but remains below on the temporal metrics: UNIT does not surpass its pseudo-labels. We recall nevertheless that UNIT work online and without accumulation and registration of scans, unlike 4D-Seg. Let us also mention that we noticed a slight drop of performance for UNIT when using the consistency loss on nuScenes. We hypothesize that the small point density leads to worse pseudo-labels for small or distant objects, which in turn creates more noise on the segments used in the consistency loss.

4.7. Fine-tuning the backbone for self-supervision

As an additional experiment, we study the potential of UNIT as a self-supervised method for semantic segmentation. We use the sparse U-Net [30] backbone pre-trained by UNIT as a point-based segmenter. Following the protocol of [4, 29] on nuScenes and SemanticKITTI, we finetune the pretrained backbone with different amount of annotated data (1%, 10%, 50% and 100%). The results are presented in Tab. 5. For comparison, we also report the scores from the literature [29]. Let us highlight nevertheless that these methods use different variants of sparse U-Net, with different input features. Yet all sparse U-Nets attain comparable baseline results when using 100% of labeled data and no pretraining.

We notice that our pretrained sparse backbone underperforms at very low data regime, but reaches better performance than state-of-the-art self-supervised techniques at high annotation percentages. It could be due to the size of the self-training head, i.e., the layers on top of the sparse U-Net. Indeed our head is a complete multi-scale transformer, as opposed to previous methods that are at most 3-layered MLPs. In the latter case, the output feature maps are more likely to be linearly separable, thus requiring less finetuning and performing better in low-data regime. A similar behavior is observed for MAE [12] where the pretraining head is an 8-block transformer and the best performance are reached for very long finetuning. We also observe that going from single-scan to multi-scan models improves the performance.

5. Conclusion

We showed that we could segment and track objects in Lidar data without requiring any manual annotations. This is made thanks to an offline method generating pseudo-labels, and to a novel auto-regressive architecture that performs robustly online once trained on these noisy pseudo-labels only. We hope that our approach will inspire other authors to consider other tracking problems where annotations are not available.

Acknowledgements: This work was performed using HPC resources from GENCI–IDRIS (Grants 2023-AD011013765R1). This work was supported in part by the French Agence Nationale de la Recherche (ANR) grant MultiTrans (ANR21-CE23-0032).

References

[1] Mehmet Aygun, Aljosa Osep, Mark Weber, Maxim Maximov, Cyrill Stachniss, Jens Behley, and Laura Leal-Taixé. 4D Panoptic LiDAR Segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2021. 6, 7

[2] Stefan Baur, Frank Moosmann, and Andreas Geiger. LISO: LiDAR-Only Self-Supervised 3D Object Detection. In *arXiv Preprint*, 2024. 1, 2

[3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *International Conference on Computer Vision*, 2019. 1, 5

[4] Alexandre Boulch, Corentin Sautier, Bjoern Michele, Gilles Puy, and Renaud Marlet. ALSO: Automotive LiDAR Self-Supervision by Occupancy Estimation. In *Conference on Computer Vision and Pattern Recognition*, 2023. 1, 8

[5] 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 *Conference on Computer Vision and Pattern Recognition*, 2020. 8

[6] 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*, 2020. 1, 4

[7] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-Pixel Classification Is Not All You Need for Semantic Segmentation. In *Advances in Neural Information Processing Systems*, 2021. 1, 3

[8] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2022. 4

[9] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, and Others. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996. 2

[10] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981. 2, 3

[11] Whye Kit Fong, Rohit Mohan, Juana Valeria Hurtado, Lubing Zhou, Holger Caesar, Oscar Beijbom, and Abhinav Valada. Panoptic nuScenes: A Large-Scale Benchmark for LiDAR Panoptic Segmentation and Tracking. *IEEE Robotics and Automation Letters (RA-L)*, 2022. 1

[12] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Conference on Computer Vision and Pattern Recognition*, 2022. 8

[13] Juana Valeria Hurtado, Rohit Mohan, Wolfram Burgard, and Abhinav Valada. MOPT: Multi-Object Panoptic Tracking. In *CVPR Workshop on Scalability in Autonomous Driving*, 2020. 1

[14] Seungjae Lee, Hyungtae Lim, and Hyun Myung. Patchwork++: Fast and Robust Ground Segmentation Solving Partial Under-Segmentation Using 3D Point Cloud. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022. 3

[15] Hyungtae Lim, Oh Minho, and Hyun Myung. Patchwork: Concentric Zone-Based Region-Wise Ground Segmentation with Ground Likelihood Estimation Using a 3D LiDAR Sensor. *IEEE Robotics and Automation Letters (RA-L)*, 2021. 2, 3

[16] R. Marcuzzi, L. Nunes, L. Wiesmann, J. Behley, and C. Stachniss. Mask-Based Panoptic LiDAR Segmentation for Autonomous Driving. *IEEE Robotics and Automation Letters (RA-L)*, 2023. 1, 2

[17] R. Marcuzzi, L. Nunes, L. Wiesmann, E. Marks, J. Behley, and C. Stachniss. Mask4D: End-to-End Mask-Based 4D Panoptic Segmentation for LiDAR Sequences. *IEEE RA-L*, 8 (11):7487–7494, 2023. 1

[18] Leland Mcinnes, John Healy, and Steve Astels. HDBSCAN: Hierarchical Density Based Clustering. *The Journal of Open Source Software*, 2(11):205, 2017. 2, 3, 6

[19] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-Object Tracking with Transformers. In *Conference on Computer Vision and Pattern Recognition*, 2022. 1

[20] Ishan Misra, Rohit Girdhar, and Armand Joulin. An End-to-End Transformer Model for 3D Object Detection. In *International Conference on Computer Vision*, 2021. 4

[21] 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 Robotics and Automation Letters (RA-L)*, 2022. 1, 2, 6, 7, 8

[22] Lucas Nunes, Rodrigo Marcuzzi, Xieyuanli Chen, Jens Behley, and Cyrill Stachniss. SegContrast: 3D Point Cloud Feature Representation Learning through Self-Supervised Segment Discrimination. *IEEE RA-L*, 2022. 1, 2, 3, 8

[23] Lucas Nunes, Louis Wiesmann, Rodrigo Marcuzzi, Xieyuanli Chen, Jens Behley, and Cyrill Stachniss. Temporal Consistent 3D LiDAR Representation Learning for Semantic Perception in Autonomous Driving. In *Conference on Computer Vision and Pattern Recognition*, 2023. 1, 2, 3, 6, 8

[24] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *TMLR*, 2024. 2

[25] Cedric Perauer, Laurenz Adrian Heidrich, Haifan Zhang, Matthias Niessner, Anastasiia Kornilova, and Alexey Artemov. AutoInst: Automatic instance-based segmentation of lidar 3d scans. *arxiv:2403.16318*, 2024. 2

[26] Gilles Puy, Spyros Gidaris, Alexandre Boulch, Oriane Siméoni, Corentin Sautier, Patrick Pérez, Andrei Bursuc, and Renaud Marlet. Three pillars improving vision foundation

model distillation for lidar. In *Conference on Computer Vision and Pattern Recognition*, 2024. 1, 6

[27] David Rozenberszki, Or Litany, and Angela Dai. UnScene3D: Unsupervised 3D Instance Segmentation for Indoor Scenes. In *Conference on Computer Vision and Pattern Recognition*, 2024. 2

[28] Corentin Sautier, Gilles Puy, Spyros Gidaris, Alexandre Boulch, Andrei Bursuc, and Renaud Marlet. Image-To-LiDAR Self-Supervised Distillation for Autonomous Driving Data. In *Conference on Computer Vision and Pattern Recognition*, 2022. 1

[29] Corentin Sautier, Gilles Puy, Alexandre Boulch, Renaud Marlet, and Vincent Lepetit. BEVContrast: Self-Supervision In BEV Space for Automotive LiDAR Point Clouds. In *3DV*, 2024. 1, 8

[30] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3D: Mask Transformer for 3D Semantic Instance Segmentation. In *International Conference on Robotics and Automation*, 2023. 1, 2, 3, 4, 8

[31] Jenny Seidenschwarz, Aljosa Osep, Francesco Ferroni, Simon Lucey, and Laura Leal-Taixé. SeMoLi: What Moves Together Belongs Together. In *Conference on Computer Vision and Pattern Recognition*, 2024. 1, 2

[32] Peize Sun, Jinkun Cao, Yi Jiang, Rufeng Zhang, Enze Xie, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple Object Tracking with Transformer. In *arXiv Preprint*, 2020. 1

[33] Yan Wang, Xiangyu Chen, Yurong You, Li Erran, Bharath Hariharan, Mark Campbell, Kilian Q. Weinberger, and Wei-Lun Chao. Train in Germany, test in the USA: Making 3D object detectors generalize. In *Conference on Computer Vision and Pattern Recognition*, 2020. 1

[34] Yanhao Wu, Tong Zhang, Wei Ke, Sabine Süsstrunk, and Mathieu Salzmann. Spatiotemporal Self-Supervised Learning for Point Clouds in the Wild. In *Conference on Computer Vision and Pattern Recognition*, 2023. 8

[35] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, and Others. Pandaset: Advanced Sensor Suite Dataset for Autonomous Driving. In *ITSC*, 2021. 1, 6

[36] Saining Xie, Jiatao Gu, Demi Guo, Charles R. Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised Pre-Training for 3D Point Cloud Understanding. In *European Conference on Computer Vision*, 2020. 8

[37] Fangao Zeng, Bin Dong, Yuang Zhang, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. MOTR: End-To-End Multiple-Object Tracking with Transformer. In *European Conference on Computer Vision*, 2022. 1

[38] 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 *Conference on Computer Vision and Pattern Recognition*, 2023. 1, 2, 3