Zero-Shot 4D Lidar Panoptic Segmentation

Supplementary Material

Abstract

In this appendix, we provide:

- *A more detailed description of our core methodology, **SAL-4D** pseudo-label engine and model in (Appendix A);*
- *In Appendix B, we provide more detailed discussion of our baselines;*
- *Additional evaluation, including pseudo-label and model ablations and per-class results (Appendix C), and, finally,*
- *Additional qualitative results (Appendix D).*

A. Implementation Details

A.1. Pseudo-label engine

This section expands the description (Sec. 3.2) of our pseudo-label engine with a higher level of detail, including pseudo-code detailing core components of our pseudo-engine (**Track–Lift–Flatten, Algorithm 1**, **Cross-Window Association, Algorithm 2**). To ensure this section is self-contained, we start with a high-level overview.

Inputs&Notation. Our pseudo-label engine operates with a multi-modal sensory setup. We assume an input Lidar sequence $\mathcal{P} = \{P_t\}_{t=1}^{T}$ along with C unlabeled videos $\mathcal{V} = \{\mathcal{V}^c\}_{c=1}^{C}$, where each video $\mathcal{V}^c = \{I_t^c\}_{t=1}^{T}$ consists of images $I_t^c \in \mathbb{R}^{H \times W \times 3}$ of spatial dimensions $H \times W$, captured by camera c at time t. For each point cloud P_t, we produce pseudo-labels, comprising of tuples $\{\tilde{m}_{i,t}, \text{id}_i, f_i\}_{i=1}^{M_t}$, where $\tilde{m}_{i,t} \in \{0,1\}^{N_t}$ represents the binary segmentation mask for instance i at time t in the point cloud P_t, and $\text{id}_i \in \mathbb{N}$ is the unique object identifier for spatio-temporal instance i. Finally, $f_i \in \mathbb{R}^d$ represents instance semantic features aggregated over time.

Hyperparameters. We list relevant hyperparameters in Tab. 1.

A.1.1. Track–Lift–Flatten

Overview. In a nutshell, for each temporal window, we track objects in video (*track*), lift masks to 4D Lidar sequences (*lift*), and, finally, "*flatten*" overlapping masklets in the 4D volume.

Sliding windows. We proceed by sliding a temporal window of size K with a stride S over the sequence of length T. We first pseudo-label each temporal window, and then perform cross-window association to obtain pseudo-labels for sequences of arbitrary length. Our temporal windows $w_k = \{(P_t, I_t) \mid t \in T_k\}$ consist of Lidar point clouds and images over specific time frames. Here, $T_k = \{t_k, t_k + 1, \ldots, t_k + K - 1\}$ is the set of time indices for window w_k. For simplicity, we drop the camera index c unless explicitly needed. We explain our approach assuming a single-camera setup ($C = 1$) and discuss the generalization to a multi-camera setup as necessary.

Track. For each video, we use a segmentation foundation model (SAM [11]) to perform grid-prompting in the first video frame of the window I_{t_k} to localize objects as masks $\{m_{i,t_k}\}_{i=1}^{M_{t_k}}$, $m_{i,t_k} \in \{0,1\}^{H \times W}$, where M_{t_k} denotes the number of discovered instances in I_{t_k}. We then propagate masks through the entire window $\{I_t \mid t \in T_k\}$ using SAMv2 [15] to obtain masklets $\{m_{i,t} \mid t \in T_k\}_{i=1}^{M_{t_k}}$ for all instances discovered in I_{t_k}. This results in M_{t_k} overlapping masklets in a 3D video volume of dimensions $H \times W \times K$, representing objects visible in I_{t_k} across the window w_k.

Given masklets $\{m_{i,t} \mid t \in T_k\}_{i=1}^{M_{t_k}}$ and corresponding images $\{I_t \mid t \in T_k\}$, we compute semantic features $f_{i,t}$ for each mask $m_{i,t}$ using relative mask attention in the CLIP [14] feature space and obtain masklets paired with their CLIP features $\{(m_{i,t}, \text{id}_{i,k}, f_{i,t}) \mid t \in T_k\}$ for each instance i, where $\text{id}_{i,k}$ is a local instance identifier within window w_k. Detailed parameters of SAM and SAMv2 can be found in Tab. 1.

Lift. We associate 3D points $\{P_t \mid t \in T_k\}$ with image masks $m_{i,t}$ via Lidar-to-camera transformation and projection. We refine our lifted Lidar masklets to address sensor misalignment errors using density-based clustering [8]. We create an ensemble of DBSCAN clusters by varying the density parameter and replacing all lifted masks with DBSCAN masks with sufficient intersection-over-union (IoU) overlap (0.5) [13]. Due to the presence of moving objects, which makes the DBSCAN cluster prone to error, we perform this procedure separately for individual scans. Detailed ablations can be found in Appendix C.

We obtain sets $\{(\tilde{m}_{i,t}^c, \text{id}_{i,k}^c, f_{i,t}^c) \mid t \in T_k\}$ independently for each camera c, and fuse instances with sufficient IoU overlap (0.5) across cameras. We fuse their semantic features $f_{i,t}$ via mask-area-based weighted average to obtain a set of tuples $\{(\tilde{m}_{i,t}, \text{id}_{i,k}, f_{i,t}) \mid t \in T_k\}$, that represent spatio-temporal instances localized in window w_k.

Flatten. The resulting set $\{(\tilde{m}_{i,t}, \text{id}_{i,k}, f_{i,t}) \mid t \in T_k\}$ contains overlapping masklets in 4D space-time volume, leading to ambiguities in point assignments. To ensure each point is assigned to at most one instance, we perform spatio-temporal flattening as follows. We compute the spatio-temporal volume V_i of each masklet $\tilde{M}_i = \{\tilde{m}_{i,t} \mid t \in T_k\}$ by summing the number of points across all frames: $V_i =$

$\sum_{t \in T_k} |\tilde{m}_{i,t}|$, where $|\tilde{m}_{i,t}|$ denotes the number of points in mask $\tilde{m}_{i,t}$. We sort the masklets in descending order based on their volumes V_i, and incrementally suppress masklets with intersection-over-minimum larger than empirically determined threshold. For each masklet $\tilde{M}_i$ in the sorted list, we compute the Intersection-over-Minimum (IoM) with all remaining masklets $\tilde{M}_j$:

$$\text{IoM}_{ij} = \frac{\sum_{t \in T_k} |\tilde{m}_{i,t} \cap \tilde{m}_{j,t}|}{\min(V_i, V_j)}. \quad (1)$$

If $\text{IoM}_{ij} > \theta$ (a predefined threshold we set it as 0.5), we suppress $\tilde{M}_j$ by removing it from the list. The value of θ controls the aggressiveness of suppression (set to a high value to prevent overlapping masklets). With this flattening operation, we favor larger and temporally consistent instances (*i.e.*, prefer larger volumes), and ensure unique point-to-instance assignments (via IoM-based suppression) in the 4D space-time volume. However, we obtain pseudo-labels *only* for objects visible in the first video frame I_{t_k} of each window w_k. Objects appearing after t_k are not captured in this label set.

A.1.2. Labeling Arbitrary-Length Sequences

After labeling each temporal window, we obtain pseudo-labels for point clouds within overlapping windows of size K, $\{(\tilde{m}_{i,t}, \text{id}_{i,k}, f_{i,t}) \mid t \in T_k\}$, with local instance identifiers $\text{id}_{i,k}$. As mentioned before, the pseudo-label only covers objects found in the first frame of each window. To produce pseudo-labels for the full sequence of length T and account for new objects entering the scene, as detailed in Algorithm 2, we associate instances across windows in a near-online fashion (with stride S), resulting in our final pseudo-labels $\{(\tilde{m}_{i,t}, \text{id}_i, f_i) \mid t \in T\}$, where id_i is consistent across the sequence and f_i is averaged CLIP feature of the same instance across the sequence.

For each pair of overlapping windows (w_{k-1}, w_k), we perform association by solving a linear assignment problem:

$$\mathbf{A}^* = \arg\min_{\mathbf{A}} \sum_{i=1}^{M_{k-1}} \sum_{j=1}^{M_k} c_{ij} A_{ij} \quad (2)$$

Subject to:

$$\sum_{j=1}^{M_k} A_{ij} \leq 1, \quad \forall i = 1, \ldots, M_{k-1}$$

$$\sum_{i=1}^{M_{k-1}} A_{ij} \leq 1, \quad \forall j = 1, \ldots, M_k$$

$$A_{ij} \in \{0, 1\}.$$

Here, A_{ij} indicates whether instance $\text{id}_{i,k-1}$ in w_{k-1} is assigned to instance $\text{id}_{j,k}$ in w_k. We derive association costs from temporal instance overlaps (measured by 3D-IoU) in the overlapping frames $T_{k-1} \cap T_k$, defined as:

$$c_{ij} = 1 - \text{IoU}_{\text{3D}}(\tilde{m}_{i,k-1}, \tilde{m}_{j,k}), \quad (3)$$

where $\tilde{m}_{i,k-1}$ and $\tilde{m}_{j,k}$ are the aggregated Lidar masks of instances i and j over the overlapping frames. This linear assignment problem can be efficiently solved using the Hungarian algorithm. After association, we update the global instance identifiers id_i for matched instances and aggregate their semantic features f_i over time. As a final post-processing step, we remove instances that are shorter than a specified temporal threshold τ (*i.e.*, instances appearing in fewer than τ frames, τ is set to 1 in our experiments).

Parameter	Value
SAM [11]	
Model	sam_vit_h_4b8939
Inference POINTS_PER_SIDE	32
Inference POINTS_PER_BATCH	64
Inference PRED_IOU_THRESH	0.84
Inference STABILITY_SCORE_THRESH	0.86
Inference STABILITY_SCORE_OFFSET	1.0
CROP_N_LAYERS	1
Inference BOX_NMS_THRESH	0.7
Inference CROP_NMS_THRESH	0.7
Inference MIN_MASK_REGION_AREA	100
SAM2 [15]	
Model	sam2_hiera_large.pt
Config	sam2_hiera_l.yaml
Pseudo-label engine	
NMS IoU threshold	0.5
Multi-view IoU threshold	0.5
DBSCAN IoU overlap threshold	0.5
DBSCAN density thresholds	(1.2488, 0.8136, 0.6952, 0.594, 0.4353, 0.3221)
Zero-shot model	
GPUs	8 × 80GB (A100)
Batch size	24 (3 per GPU)
Learning rate (LR)	0.0002
Number of iterations	40000
LR scheduler	OneCycleLR (pct_start=0.1)
Number of queries	300
Overlap threshold	0.0
Loss weights	2.0, 5.0, 5.0, 10.0, 2.0

Table 1. **SAL-4D hyperparameters.** We list hyperparameters, including (i) segmentation foundation model parameters (SAM model [11], which we use to generate segmentation masks in images, and SAMv2 [15] that we use for the temporal mask propagation), (ii) the pseudo-label engine and (iii) 4D zero-shot segmentation model parameters.

A.2. Model

This section extends Sec. 3.3, and provides a more detailed description of our model. Our model operates on point clouds $\mathcal{P}_{super} \in \mathbb{R}^{N \times 4}$, $N = N_{t_k} + \ldots + N_{t_k+K-1}$, superimposed over fixed-size temporal windows w_k. Within these, our model directly estimates a set of spatio-temporal instances as (binary) segmentation masks, $\mathcal{M} \in \mathbb{R}^{M \times N}$. Instead of estimating a posterior over a (fixed) set of seman-

Algorithm 1 Track-Lift-Flatten (Per-Window Processing)

Input: Window index k, time indices T_k, Lidar point clouds $\{P_t \mid t \in T_k\}$, images $\{I_t^c \mid t \in T_k, c = 1, \ldots, C\}$
Output: Pseudo-labels for window w_k: $\{(\tilde{m}_{i,t}, \mathrm{id}_{i,k}, f_{i,t}) \mid t \in T_k\}$

// Track
for each camera c **do**
  $I_{t_k}^c \leftarrow$ image at time t_k from camera c
  $\{m_{i,t_k}^c\} \leftarrow \mathrm{SAM}(I_{t_k}^c)$ ▷ Generate initial masks
  $\{m_{i,t}^c\}_{t \in T_k} \leftarrow \mathrm{SAMv2}(\{I_t^c\}_{t \in T_k}, \{m_{i,t_k}^c\})$ ▷ Propagate masks
  $\{f_{i,t}^c\}_{t \in T_k} \leftarrow \mathrm{MaskCLIP}(\{I_t^c\}_{t \in T_k}, \{m_{i,t}^c\}_{t \in T_k})$ ▷ Compute semantic features
end for
// Lift
for each time $t \in T_k$ **do**
  $P_t \leftarrow$ Lidar point cloud at time t
  for each instance i **do**
    for each camera c **do**
      $\tilde{m}_{i,t}^c \leftarrow \mathrm{project\_mask}(P_t, m_{i,t}^c)$ ▷ Project image masks onto Lidar
    end for
    $\tilde{m}_{i,t} \leftarrow \mathrm{merge\_masks}(\{\tilde{m}_{i,t}^c\}_{c=1}^C)$ ▷ Merge masks from all cameras
    $\tilde{m}_{i,t} \leftarrow \mathrm{refine\_with\_DBSCAN}(\tilde{m}_{i,t}, P_t)$ ▷ Refine using DBSCAN
  end for
end for
// Flatten
Compute volumes $V_i \leftarrow \sum_{t \in T_k} |\tilde{m}_{i,t}|$ for each instance i
Sort instances $\{i\}$ in descending order of V_i
for each instance i in sorted order **do**
  for each instance $j \neq i$ not yet suppressed **do**
    Compute $\mathrm{IoM}_{ij} \leftarrow \frac{\sum_{t \in T_k} |\tilde{m}_{i,t} \cap \tilde{m}_{j,t}|}{\min(V_i, V_j)}$
    if $\mathrm{IoM}_{ij} > \theta$ **then**
      Suppress instance j
    end if
  end for
end for
Assign local instance IDs $\mathrm{id}_{i,k}$ within window w_k
return $\{(\tilde{m}_{i,t}, \mathrm{id}_{i,k}, f_{i,t}) \mid t \in T_k\}$

tic classes (as in prior work [1, 12, 18]), we regress objectness scores $\mathcal{O} \in \mathbb{R}^{M \times 2}$ that indicate how likely an instance represents an actual object. Following [13], we additionally regress for each instance a semantic (CLIP [14]) feature token $\mathcal{F} \in \mathbb{R}^{M \times d}$ that can be used for zero-shot recognition at the test-time.

Hyperparameters. We list relevant hyperparameters in Tab. 1.

Algorithm 2 Pseudo-label Engine with Cross-Window Association

Input: Lidar sequence $\mathcal{P} = \{P_t\}_{t=1}^T$, unlabeled videos $\mathcal{V} = \{\mathcal{V}^c\}_{c=1}^C$, window size K, stride S
Output: Pseudo-labels $\{(\tilde{m}_{i,t}, \mathrm{id}_i, f_i)\}$ for $t = 1$ to T

Initialize global instance ID counter: id $\leftarrow 0$
Initialize empty global instance set: $\mathcal{I} \leftarrow \emptyset$
for $k = 0$ to $\lceil \frac{T}{S} \rceil$ **do** ▷ Slide temporal window
  $t_k \leftarrow kS$
  $T_k \leftarrow \{t_k, t_k + 1, \ldots, \min(t_k + K - 1, T)\}$ ▷ Time indices for window w_k
  // Per-Window Processing
  $\{(\tilde{m}_{i,t}, \mathrm{id}_{i,k}, f_{i,t}) \mid t \in T_k\} \leftarrow$ **Track-Lift-Flatten**$(k, T_k, \{P_t\}_{t \in T_k}, \{I_t^c\}_{t \in T_k})$
  // Cross-Window Association
  if $k > 0$ **then**
    $O_k \leftarrow T_k \cap T_{k-1}$ ▷ Overlapping time frames
    For instances in w_{k-1} and w_k, compute costs:
    for each instance i in w_{k-1} **do**
      for each instance j in w_k **do**
        $\tilde{m}_{i,O} \leftarrow \mathrm{aggregate\_masks}(\{\tilde{m}_{i,t}\}_{t \in O_k})$
        $\tilde{m}_{j,O} \leftarrow \mathrm{aggregate\_masks}(\{\tilde{m}_{j,t}\}_{t \in O_k})$
        $c_{ij} \leftarrow 1 - \mathrm{IoU}_{\mathrm{3D}}(\tilde{m}_{i,O}, \tilde{m}_{j,O})$
      end for
    end for
    Solve linear assignment problem with costs c_{ij}
    for each instance i is matched **do**
      Update global instance IDs id_i for matched instances
    end for
    for each instance i is not matched **do**
      Assign new global instance IDs id_i for the unmatched new instances
    end for
  end if
  Add instances from window w_k to global set $\mathcal{I}$
  for each instance i in $\mathcal{I}$ **do**
    Aggregate semantic features f_i over time
  end for
end for
// Post-processing
for each instance i in $\mathcal{I}$ **do**
  if number of frames where instance i appears $< \tau$ **then**
    Remove instance i from $\mathcal{I}$ ▷ Discard short-lived instances
  end if
end for
return $\{(\tilde{m}_{i,t}, \mathrm{id}_i, f_i)\}$ for all i and t

Model. Our model operates on point clouds $\mathcal{P}_{super} \in \mathbb{R}^{N \times 4}$, $N = N_{t_k} + \ldots + N_{t_k+K-1}$, superimposed over fixed-size temporal windows w_k. As in [13], we encode

superimposed sequences using Minkowski U-Net [6] backbone to learn a multi-resolution representation of our input using sparse 3D convolutions. resulting in voxel features $F_v \in \mathbb{R}^{C_v \times N_v}$ and point feature $F_p \in \mathbb{R}^{C_p \times N}$. For spatio-temporal reasoning, we augment voxel features with Fourier positional embeddings [16, 18] that encode 3D spatial and temporal coordinates.

Our segmentation decoder follows the design of [3, 4, 12]. Inputs to the decoder are a set of M learnable queries that interact with voxel features, *i.e.*, our (4D) spatio-temporal representation of the input sequence. For each query, we estimate a spatio-temporal mask , an objectness score indicating how likely a query represents an object and a d-dimensional CLIP token capturing object semantics.

A.2.1. Backbone

As in [13], we encode superimposed sequences using Minkowski U-Net [6] backbone to learn a multi-resolution representation of our input using sparse 3D convolutions. resulting in voxel features $F_v \in \mathbb{R}^{C_v \times N_v}$ and point feature $F_p \in \mathbb{R}^{C_p \times N}$. For spatio-temporal reasoning, we augment voxel features with Fourier positional embeddings [16, 18] that encode 3D spatial and temporal coordinates.

A.2.2. Superimposing Point Clouds

At *test-time*, we transform point clouds to a common coordinate frame using known ego-poses, concatenate points, and voxelize them. Due to the voxelization of point clouds, such concatenation has a minor memory overhead (by contrast to point-based backbones that require more careful superposition strategies [1], which utilizes point-based backbones and performs sub-sampling). However, at the *train time*, we leave 10% of batches un-aligned to expose the network to a larger variety of non-aligned spatio-temporal instances to reduce the imbalance between spatially aligned (static) and non-aligned (dynamic) instances. This imbalance is especially visible in our zero-shot scenario, as opposed to prior works that specialize to *thing* classes, among which we observe a larger percentage of moving objects.

A.2.3. Segmentation Decoder

Our segmentation decoder follows the design of [3, 4, 12]. Inputs to the decoder are a set of M learnable queries that interact with voxel features, *i.e.*, our (4D) spatio-temporal representation of the input sequence. For each query, we estimate a spatio-temporal mask $\mathcal{M} \in \mathbb{R}^{M \times N}$, an objectness score $\mathcal{O} \in \mathbb{R}^{M \times 2}$ indicating how likely a query represents an object and a d-dimensional CLIP token $\mathcal{F} \in \mathbb{R}^{M \times d}$ capturing object semantics.

A.2.4. Training

Our network predicts a set of spatio-temporal instances, parametrized via segmentation masks over the superimposed point cloud: $\hat{m}_j \in \{0,1\}^N$, $j = 1, \ldots, M$, obtained by sigmoid activating and thresholding the spatio-temporal mask $\mathcal{M}$. To train our network, we first establish correspondences between our set of predictions $\{(\hat{m}_{i,t}, \text{id}_{i,k}, f_{i,t}) \mid t \in T_k\}$ and pseudo-labels $\{(\tilde{m}_{i,t}, \text{id}_{i,k}, f_{i,t}) \mid t \in T_k\}$ based on the mask intersection-over-union within temporal window (we perform bipartite matching using Hungarian algorithm, as commonly done by Mask transformer-based methods [12, 18]). Once matches are established, we evaluate the following loss:

$$\mathcal{L}_{SAL-4D} = \mathcal{L}_{obj} + \mathcal{L}_{mask} + \mathcal{L}_{dice} + \mathcal{L}_{token} + \mathcal{L}_{token\_aux}, \tag{4}$$

with a cross-entropy loss $\mathcal{L}_{obj}$ indicating whether a mask localizes an object, a segmentation loss consists of a binary cross-entropy $\mathcal{L}_{mask}$ and a dice loss $\mathcal{L}_{dice}$ following [12], and cosine distanc CLIP token losses $\mathcal{L}_{token}$ and $\mathcal{L}_{token\_aux}$ following [12]. As all three terms are evaluated on a sequence rather than individual frame level, our network implicitly learns to segment and associate instances over time, encouraging temporal semantic coherence.

As we are training with noisy pseudo-labels that label only a portion of the full Lidar point cloud, we use standard data augmentations (translation, scaling, rotations, *cf*., [12]), as well as FrankenFrustum [13] to train a model that can segment full Lidar point clouds. We also follow the recommendation by [13] and remove all unlabeled points (*i.e.*, those not covered by our pseudo-labels) from our training instances.

A.2.5. Inference

The mask inference is done by first multiplying the objectness score with the spatio-temporal mask $\mathcal{M} \in \mathbb{R}^{M \times N}$ and then performing argmax over each point:

$$\text{score} = \max(\mathcal{O} \in \mathbb{R}^{M \times 2}, \text{dim=-1}), \tag{5}$$

$$\text{mask} = \text{argmax}(\text{sigmoid}(\mathcal{M} \in \mathbb{R}^{M \times N}) \cdot \text{score}, \text{dim=0}). \tag{6}$$

As our model directly processes superimposed point clouds within windows of size K, we perform *near-online* inference [5] by associating Lidar masklets across time based on 3D-IoU overlap via bi-partite matching (as described in Sec. 3.2.2). For zero-shot prompting, we follow [13] and first encode prompts specified in the semantic class vocabulary using a CLIP language encoder. Then, we perform argmax over scores, computed as a dot product between encoded queries and predicted CLIP features.

B. Baselines Details

We evaluate several alternative approaches for *ZS-4D-LPS*, inspired by multi-object tracking [17] and video-instance segmentation [10] communities. In this section, we provide implementation details for these baselines.

Parameter	Value
AB3DMOT Parameters [17]	
algm	"greedy"
metric	"giou_3d"
thres	-0.4
min_hits	1
max_age	2
Ego-motion compensation	Yes

Table 2. **Multi-object tracking (MOT) baseline**. We report the key hyperparameters used in our adaptation of [17].

Stationary world (SW). As SemanticKITTI [2] is dominated by static objects, the minimal viable baseline utilizes ego-motion to propagate masks, estimated by a single-scan network [13]. To this end, we first process each point cloud individually using SAL [13]. To associate masks from $P_{t-1} \rightarrow P_t$, we perform the following: we transform all point clouds in the sequence to a common coordinate frame at time t. Then, we compute for each point $p_i \in P_t$ a nearest-neighbor $p_j \in P_{t-1}$. Then for each instance id_i that appears in the current frame P_t, find all the points $p_i \in P_t$, where $\text{id}(p_i) \in \{\text{id}_i\}$. The corresponding nearest points in the previous frame $p_j \in P_{t-1}$ have $\text{id}(p_j) \in \{\text{id}_{j_1}, \text{id}_{j_2}, \text{id}_{j_3} ...\}$. We determine for each instance $\text{id}(p_i)$ a track ID via majority voting of $\text{id}(p_j)$. The threshold of majority voting is set to 0.5.

Multi-object tracking (MOT). Model-free approaches that utilize Kalman filters in conjunction with linear or greedy association of single-scan object detections are strong baselines for Lidar-based tracking [17]. To this end, we adapt [17] to associate masks from SAL [13]. Approach by [17] parametrizes object tracks via object-oriented 3D bounding boxes (parametrized via center, bounding box size, and yaw-angle). Tracks are propagated from past point clouds to the current state via a constant-velocity Kalman filter, and associations are determined based on 3D intersection-over-union (IoU) between track predictions and detected objects (also parametrized as object-oriented bounding boxes). We adapt [17] in our work by first predicting segmentation masks for each point cloud and then fitting bounding box to each segment (the box boundary are set as the minimum and maximum 3D coordinates of the segmentation masks, $\text{Bbox} = \{x_{min}, y_{min}, z_{min}, x_{max}, y_{max}, z_{max}\}$). We report our configuration for [17] in Tab. 2.

Video Instance Segmentation (VIS) This baseline associates objects in 3D without explicit sequence-level training. Specifically, we adapt a video instance segmentation approach MinVIS [10], that utilizes object queries for associating objects at test time within Lidar data. The algorithm operates as follows. We first generate N object queries per frame using SAL [13]. Then we match queries from frame t to frame $t+1$ using cosine distance as the metric. Finally, the IDs are transferred based on established matches. As we only have a limited number of queries, which makes long-term tracking challenging. To solve this, we first do MinVIS within a temporal window of size 2 and then employ the same cross-window association as our **SAL-4D** model prediction for post-processing.

C. Additional Experimental Evaluation

Algorithm 3 Single-scan 3D SAL [13] pseudo-label engine

Input: Lidar point clouds P_t, C camera views $\mathcal{I}_t^c$, C camera calibrations K_c, timestamps $t \in 1, \ldots, T$
Output: $\{\tilde{m}_t, f_t\}, t \in 1, \ldots, T$

for each timestamp t **do**
  $P_t \leftarrow$ load_lidar(t)
  $\tilde{m}_t = \emptyset$, $f_t = \emptyset$
  $\tilde{m}_t^{DBSCAN} \leftarrow$ DBSCAN_ensamble(P_t)
  for each camera c **do**
    $\mathcal{I}_t^c \leftarrow$ load_image(t, K_c)
    $m_t^c \leftarrow$ SAM($\mathcal{I}_t^c$)
    $f_t^c \leftarrow$ MaskCLIP [7]($\mathcal{I}_t^c, m_t^c$)
    $\tilde{m}_t^c \leftarrow$ lift_to_3D (P_t^c, m_t^c, K_c)
    $\tilde{m}_t^c \leftarrow$ DBSCAN_refine($\tilde{m}_t^c, \tilde{m}_t^{DBSCAN}$)
    $\tilde{m}_t^c \leftarrow$ flatten_in_3D($\tilde{m}_t^c$)
    $\{\tilde{m}_t, f_t\} \leftarrow$ insert_or_merge($\tilde{m}_t, f_t, \tilde{m}_t^c, f_t^c$)
  end for
end for

C.1. Pseudo-label Engine Ablations

DBSCAN. We investigate how to use DBSCAN for segmentation refinement during pseudo-label generation. Tab. 3 shows the effect of doing DBSCAN on per scan separately or on all the scans within the temporal window all together. The temporal window size is set to 2. The best pseudo-label is obtained by only enabling DBSCAN per scan separately. Possibly because doing DBSCAN on all the scans will harm the segmentation performance on dynamic objects, which results in a significant drop in association score (S_{assoc}) when enabled.

C.2. Single-scan SAL pseudo-label improvements

In the process of developing **SAL-4D**, we also re-think and improve the single-scan 3D pseudo-labels proposed in [13]. Training on these labels yields the 3D SAL model results reported in the main paper (we report improved results, compared to those reported in [13]). We formalize our novel single-scan label engine in Algorithm 3 and ablate the performance boosts for class-agnostic and zero-shot *Lidar Panoptic Segmentation* (LPS) of the following improvements in Tab. 4.

Flatten in 3D. In contrast to [13], we switch the order of Flatten–Lift to Lift–Flatten, *i.e.*, perform the flattening

# frames	per-frame	all-frame	Frust. Eval.	LSTQ	S_{assoc}	S_{cls}	IoU_{st}	IoU_{th}
Class-agnostic (Semantic Oracle)								
2	✓		✓	63.5	68.0	59.3	56.1	71.0
2		✓	✓	60.8	63.9	57.8	54.9	68.9
2	✓	✓	✓	60.5	63.5	57.7	54.4	69.4
Zero-Shot								
2	✓		✓	46.3	66.4	32.3	34.1	33.9
2		✓	✓	44.6	62.6	31.8	33.6	33.3
2	✓	✓	✓	44.2	62.0	31.5	33.2	33.1

Table 3. **Pseudo-label ablations on DBSCAN settings, per-frame or all-frame:** We show the effect of doing DBSCAN per scan separately or on all the scans within the temporal window together on the KITTI validation set. The temporal window size is set to 2. The results show that doing DBSCAN per-frame gives the best result.

Single-scan 3D pseudo-labels	PQ	SQ	PQ_{th}	PQ_{st}
Class-agnostic (Semantic Oracle) LPS				
Original	48.7	73.7	53.1	45.4
+ Flatten in 3D	51.8	78.3	62.1	44.4
+ DBSCAN refine per instance	53.6	80.1	65.2	45.2
+ Flatten via coverage	55.3	79.9	66.0	47.5
Zero-Shot LPS				
Original	27.5	71.5	31.7	24.5
+ Flatten in 3D	28.6	73.4	34.0	24.7
+ DBSCAN refine per instance	29.7	75.1	36.0	25.1
+ Flatten via coverage	29.9	74.8	35.2	26.0

Table 4. **Single-scan 3D pseudo-label improvements:** We report class-agnostic and zero-shot single-scan *Lidar Panoptic Segmentation* (LPS) results with several improvements added to the original [13] pseudo-labels. Evaluation is performed in the camera frustum of the *SemanticKITTI* validation set.

# frames	LSTQ	S_{assoc}	S_{cls}	IoU_{st}	IoU_{th}
2	30.0	31.1	28.9	31.9	29.5
4	27.6	26.9	28.4	31.6	28.7

Table 5. **Pseudo-label ablations on nuScenes dataset on temporal window size:** We ablate on temporal window sizes $2-4$ frames. The quality of pseudo labels with 4 frame temporal window drops significantly. The stride is set as half the window size.

of overlapping SAM [11] masks after and not before their unprojection to 3D. To this end, we apply a non-maximum suppression (NMS) in 3D for the $flatten\_in\_3D$ step in line 11 of Algorithm 3. Lift–Flatten has the advantage of resolving potentially ambiguous or edge-case overlaps in the 2D image after their unprojection to the actual 3D geometry. Furthermore, we can run our DBSCAN refinement before the flattening. The performance boost of $+3.1$ PQ is particularly noticeable for class-agnostic segmentation.

DBSCAN refine per instance. The original DBSCAN refinement step in [13] creates an ensemble of DBSCAN segments (line 4 in Algorithm 3) by first removing the ground plane and then collecting the segments of a set of epsilon density parameters. Afterward, each SAM-based 3D mask (line 9 in Algorithm 3) with a sufficiently large IoU is replaced with a DBSCAN segment. This step refines the image-based segments and removes false positives or adds false negatives caused by wrong SAM predictions or unprojection/parallax errors. Since DBSCAN can only make statements on non-ground plane points, any ground point is added back to its original 3D instance.

Our improved DBSCAN refinement mitigates this issue and removes potential false positives even in the ground plane. To this end, we run an additional DBSCAN segmentation on each previously replaced instance. We remove all points that do not belong to the instance, keep potential ground points, and use the same epsilon density value that produced the original DBSCAN replacement mask. Using the same epsilon, we introduce an expected density prior that allows us to remove all ground points following a different distribution. The additional per-instance refinement improves class-agnostic and zero-shot *Lidar Panoptic Segmentation* performance by $+1.8$ and $+1.1$ PQ, respectively.

Flatten via coverage. Our final improvement of the single-scan label engine changes the matching metric for the 3D NMS applied during flattening (line 11 in Algorithm 3). Instead of IoU, we compute coverage (intersection-over-minimum) which removes any mask significantly covered by another mask independently of the relative mask sizes. Flattening via coverage removes many small noisy segments, for example, on large road segments. In particular, class-agnostic segmentation performance improves by $+1.7$ PQ points.

C.3. Per-Class Results

We report per-class results for Zero-Shot Lidar Panoptic Segmentation (PQ) in Tab. 9. Remarkably, not only we consistently outperform SAL [13] on (almost) all classes on both, *SemanticKITTI* and *Panoptic nuScenes* – we show we can localize and recognize even instances that the single-scan model by [13] (`motorcyclist`, `cyclist`, `barrier`) is unable to segment.

# frames	LSTQ	S_{assoc}	S_{cls}	IoU_{st}	IoU_{th}
2	46.3	66.4	32.3	34.1	33.9
4	48.0	68.9	33.5	35.3	35.1
8	49.2	70.0	34.6	36.0	36.9
16	49.9	70.0	35.6	36.4	39.0

Table 6. **Pseudo-label ablations on temporal window size without cross window association:** We ablate our approach on temporal window sizes of size $K = \{2, 4, 8, 16\}$ with stride $\frac{K}{2}$ on *SemanticKITTI* validation set. We got a similar observation as the ablation study on the cross-associated version of pseudo-labels that the association score (S_{assoc}) improves up to 8 frames, while zero-shot recognition does not saturate and continues to improve as the temporal window size increases.

Method	label	LSTQ	S_{assoc}	S_{cls}	IoU_{st}	IoU_{th}
SAL-4D	v_1	50.7	67.2	38.3	48.7	28.8
SAL-4D	v_2	53.2	77.2	36.6	47.9	25.6

Table 7. **4DSAL ablations on training on different version of labels:** We ablate our model on training on different versions of labels on *SemanticKITTI*. The temporal window size is set to $K = 8$ with stride 4. Pseudo-label v_1: the pseudo-labels are not associated cross window (*i.e.*, the semantic features are aggregated per window). Pseudo-label v_2: the pseudo-labels are associated cross window (*i.e.*, the semantic features are aggregated over the whole sequence).

Method	# frames	Franken Frustum	LSTQ	S_{assoc}	S_{cls}	IoU_{st}	IoU_{th}
pseudo-labels		×	5.8	4.0	8.4	6.9	11.7
SAL-4D	2	×	8.3	5.4	12.7	20.2	2.3
SAL-4D	2	✓	42.2	51.1	34.9	45.1	20.8

Table 8. **SAL-4D on SemanticKITTI validation set, full (360°) point cloud evaluation.** On SemanticKITTI, only 14% of all Lidar points are seen in the left RGB camera, used for pseudo-labeling. Due to low coverage, when we evaluate pseudo-labels, we obtain $LSTQ$ of 5.8 (low recall). It is critical to train the model using *FrankenFrustum* augmentation to obtain a good generalization to the whole point cloud (42.2 $LSTQ$) – only employing standard data augmentations (rotation, translation, scaling) is not sufficient (8.3 $LSTQ$).

C.4. Per-Window *vs*. Per-Sequence Labels

Tab. 6 evaluates pseudo-labels v_1 w.r.t. window size, without cross-window association (*i.e.*, the semantic features are aggregated per window). In the main paper, we report v_2 labels that additionally apply cross-window association (*i.e.*, the semantic features are aggregated over the whole sequence). We observe similar trends, that association performance (S_{assoc}) improvements saturate at window sizes of 8, while zero-shot recognition (S_{cls}) benefits from a larger temporal span. However, overall, we obtain better results with v_2 labels, as reported in the main paper.

This is also reflected in Tab. 7, where we train our model with v_1 and v_2 pseudo-labels. With v_2, we obtain overall higher $LSTQ$ (53.2), compared to v_1 (50.7). We observe that training on the cross window associated version of the pseudo-label improves significantly on association score S_{assoc} by about 15%, which demonstrates that our cross window associated pseudo label, accounting for objects entering the scene, provides precisely the supervisory signal for 4D Lidar segmentation. We note that while cross-window association significantly improves the association aspect, we observe a less severe drop in terms of zero-shot recognition (-1.7 S_{cls}).

C.5. Franken Frustum

Tab. 8 shows the generalization ability of our model and the importance of applying Franken Frustum data augmentation. The results show that if we only train on 14% of the labeled data, the model doesn't generate well when evaluated on the full point cloud (8.3 LSTQ) even with standard data augmentation. By additionally employing Franken Frustum augmentation, the model generates well outside of the camera Frustum and achieves 42.2 LSTQ.

D. Qualitative Results

Zero-Shot 4D Lidar Panoptic Segmentation. In Fig. 2 and Fig. 3, we visualize ground-truth labels (GT) (*left*), pseudo-labels (*center*), and **SAL-4D** results (*right*) on *SemanticKITTI* and *Panoptic nuScenes*, respectively. We visualize three different scenes per dataset, shown as superimposed point clouds. In the *top* row, we visualize semantics, and in the *bottom* row, we visualize (4D) instances. **Importantly, to visualize semantic classes, we prompt individual instances with test-time specified prompts that conform to class vocabularies of *SemanticKITTI* and *Panoptic nuScenes*, respectively. Neither pseudo-labels nor our model has any explicit semantic information about these object classes.** As can be seen, GT labels provide instance labels only for specific *thing* classes, whereas our pseudo-labels and model predictions densely segment point clouds consistently in space and time.

Our pseudo-labels only cover a small portion of the point cloud (14%); however, our model learns to segment *full* point clouds. Tab. 8 confirms that we can achieve such a generalization using suitable data augmentations.

Arbitrary prompts. We report additional qualitative results with arbitrary text prompts in Fig. 1. In particular, we specify single-class prompts and highlight objects in **orange**) for four different prompts. Two are canonical objects (`car` and `bicycle rider`), and two are not parts of standard class vocabularies in Lidar segmentation: `advertising stand` and `electric street box`. Nevertheless, our **SAL-4D** segments all objects correctly (three different types of advertisement stands and two electric boxes). We provide images only for reference.

Method	SemanticKITTI [2]																			
	all	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
SAL	25.3	78.8	18.2	20.3	7.5	8.7	12.6	0.0	0.0	70.3	3.2	28.7	0.0	44.6	3.2	76.5	18.8	30.0	33.6	24.6
SAL-4D	30.8	84.3	26.9	26.7	15.5	16.2	11.9	21.0	1.7	74.1	3.0	33.4	0.0	62.5	9.2	82.4	14.1	35.7	37.3	28.9

	nuScenes [9]																
	all	barrier	bicycle	bus	car	construction_vehicle	motorcycle	pedestrian	traffic_cone	trailer	truck	driveable_surface	other_flat	sidewalk	terrain	manmade	vegetation
SAL	41.2	0.6	32.8	60.3	82.9	26.4	48.8	57.3	42.5	31.3	53.1	63.1	1.6	16.3	36.6	33.3	71.7
SAL-4D	45.7	1.1	68.1	60.8	85.3	32.2	73.7	62.3	37.2	33.9	56.4	56.6	0.1	13.7	39.4	35.5	75.0

Table 9. **Per-class (zero-shot) results (PQ) for SAL-4D and SAL [13] on *SemanticKITTI* and *nuScenes-Panoptic* validation sets.** Our **SAL-4D** consistently outperforms SAL on (almost) all classes. Due to limited temporal context, SAL fails to segment smaller objects such as `motorcyclist`, `cyclist`, `barrier`. **SAL-4D** substantially improves segmentation of such objects.

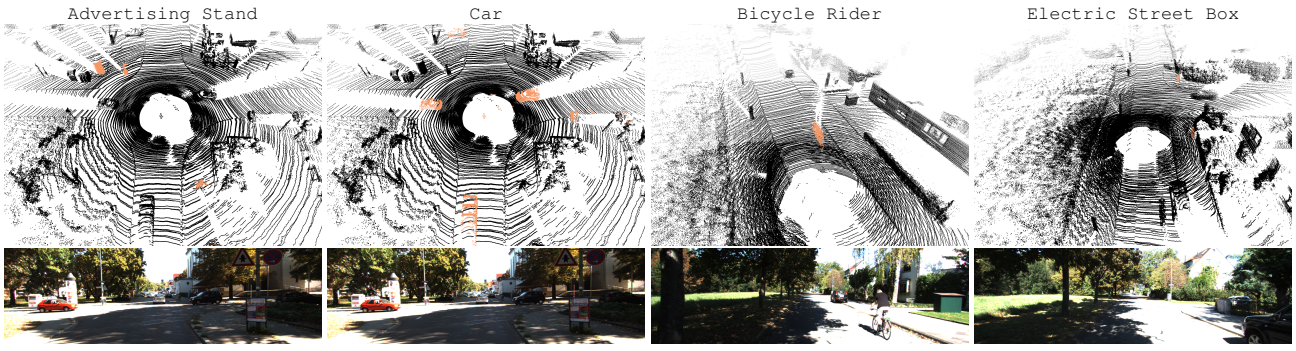

Figure 1. **Prompt examples**. We visualize the output of our model (we highlight objects in **orange**) for four different prompts: two canonical `car` and `bicycle rider`, and two "arbitrary" object, `advertising stand` and `electric street box`. As can be seen, all are segmented correctly, including stationary and moving instances. **Remarkably, all three different types of `advertising stand`, and both instances of `electric street box` are correctly segmented.** We provide images for reference; images are *not* used as input to our model. *Best seen in color, zoomed.*

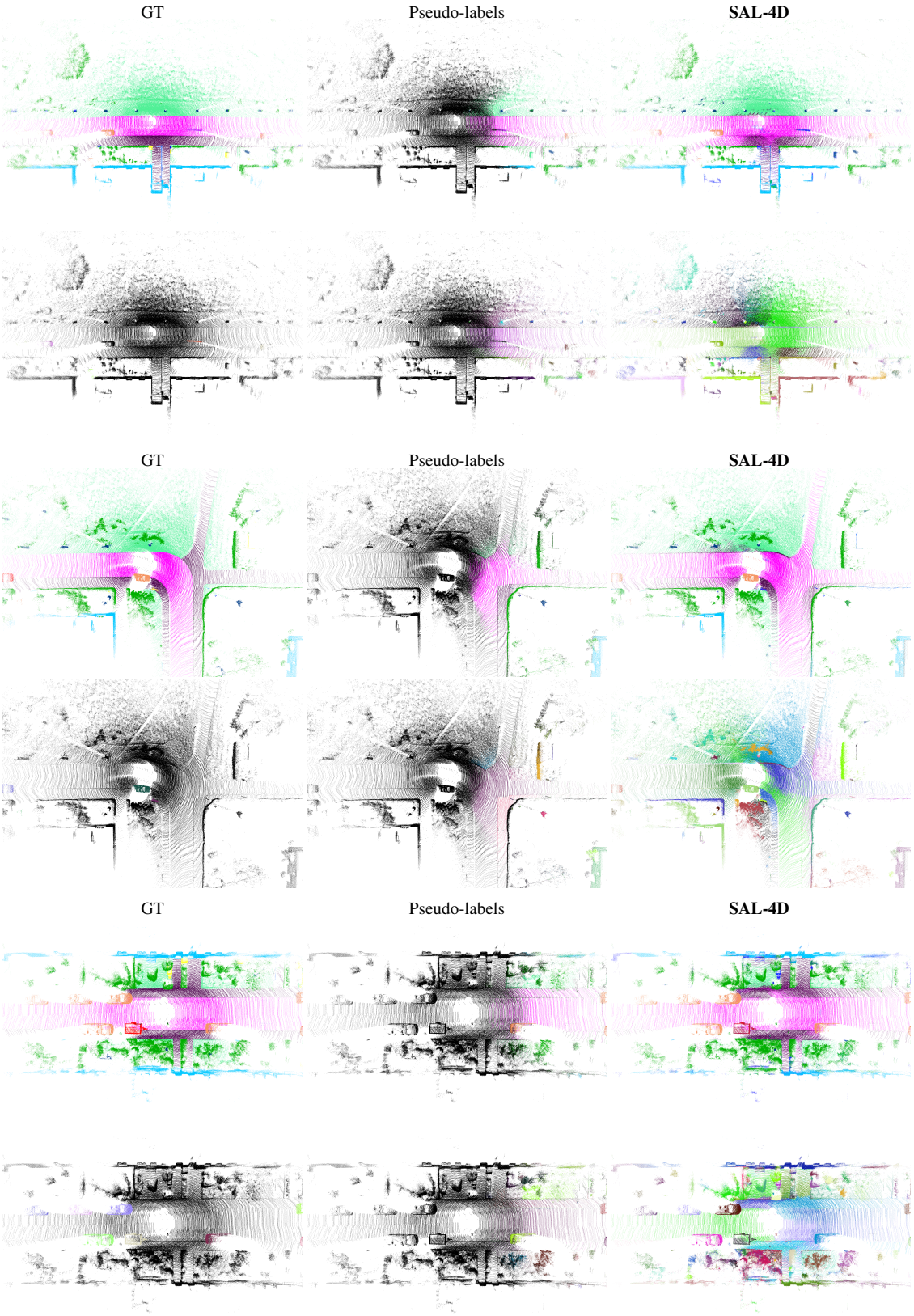

Figure 2. **Qualitative results on SemanticKITTI**. We show ground-truth (GT) labels (*first column*), our pseudo-labels (*middle column*), and **SAL-4D** results (*right column*). We show three scenes (we superimpose point clouds). For each, we show semantic predictions in the *first row* and instances predictions in the *second row*. **Importantly, we visualize semantics for pseudo-labels via zero-shot prompting whereas pseudo-labels do not provide explicit semantic labels, only CLIP tokens.**

Figure 3. **Qualitative results on Panoptic nuScenes**. We show ground-truth (GT) labels (*first column*), our pseudo-labels (*middle column*), and **SAL-4D** results (*right column*). We show three scenes (we superimpose point clouds). For each, we show semantics predictions in the *first row* and instances predictions in the *second row*. **Importantly, we visualize semantics for pseudo-labels via zero-shot prompting; pseudo-labels do not provide explicit semantic labels, only CLIP tokens.** In nuScenes, points also reflect from the ego-vehicle (seen as a car-shaped object in the center, replicated along the trajectory when the vehicle is moving; see 2^{nd} and 3^{rd} scene examples).

References

[1] Mehmet Aygün, Aljoša Ošep, Mark Weber, Maxim Maximov, Cyrill Stachniss, Jens Behley, and Laura Leal-Taixé. 4d panoptic lidar segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 3, 4

[2] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *ICCV*, 2019. 5, 8

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Eur. Conf. Comput. Vis.*, 2020. 4

[4] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 4

[5] Wongun Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *Int. Conf. Comput. Vis.*, 2015. 4

[6] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 4

[7] Zheng Ding, Jieke Wang, and Zhuowen Tu. Open-vocabulary universal image segmentation with maskclip. In *Int. Conf. Mach. Learn.*, 2023. 5

[8] 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. 1

[9] 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. *RAL*, 2021. 8

[10] De-An Huang, Zhiding Yu, and Anima Anandkumar. Minvis: A minimal video instance segmentation framework without video-based training. In *Adv. Neural Inform. Process. Syst.*, 2022. 4, 5

[11] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Int. Conf. Comput. Vis.*, 2023. 1, 2, 6

[12] Rodrigo Marcuzzi, Lucas Nunes, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. Mask-based panoptic lidar segmentation for autonomous driving. *IEEE Rob. Automat. Letters*, 2023. 3, 4

[13] Aljosa Osep, Tim Meinhardt, Francesco Ferroni, Neehar Peri, Deva Ramanan, and Laura Leal-Taixe. Better call sal: Towards learning to segment anything in lidar. In *Eur. Conf. Comput. Vis.*, 2024. 1, 3, 4, 5, 6, 8

[14] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Int. Conf. Mach. Learn.*, 2021. 1, 3

[15] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 1, 2

[16] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Adv. Neural Inform. Process. Syst.*, 2020. 4

[17] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3D Multi-Object Tracking: A Baseline and New Evaluation Metrics. In *Int. Conf. Intel. Rob. Sys.*, 2020. 4, 5

[18] Kadir Yilmaz, Jonas Schult, Alexey Nekrasov, and Bastian Leibe. Mask4former: Mask transformer for 4d panoptic segmentation. In *Int. Conf. Rob. Automat.*, 2024. 3, 4