Lidar Panoptic Segmentation and Tracking without Bells and Whistles

Abhinav Agarwalla[1], Xuhua Huang[1,2], Jason Ziglar[3], Francesco Ferroni[3],
Laura Leal-Taixé[5], James Hays[4], Aljoša Ošep[1] and Deva Ramanan[1]
[1]Carnegie Mellon University [2]Meta AI [3]Argo AI [4]Georgia Institute of Technology [5]TU Munich

***Abstract*— State-of-the-art lidar panoptic segmentation (LPS) methods follow "bottom-up" segmentation-centric fashion wherein they build upon semantic segmentation networks by utilizing clustering to obtain object instances. In this paper, we re-think this approach and propose a surprisingly simple yet effective detection-centric network for both LPS and tracking. Our network is modular by design and optimized for all aspects of both the panoptic segmentation and tracking task. One of the core components of our network is the object instance detection branch, which we train using point-level (modal) annotations, as available in segmentation-centric datasets. In the absence of amodal (cuboid) annotations, we regress modal centroids and object extent using trajectory-level supervision that provides information about object size, which cannot be inferred from single scans due to occlusions and the sparse nature of the lidar data. We obtain fine-grained instance segments by learning to associate lidar points with detected centroids. We evaluate our method on several 3D/4D LPS benchmarks where we significantly outperform existing methods that use comparable backbones and obtain competitive results compared to recent transformer-based architectures.**

I. Introduction

Lidar panoptic segmentation (LPS) is the task of labeling all 3D points with distinct semantic classes and instance IDs. This is directly relevant for autonomous navigation and other online streaming robot operation, as robots need to be aware of both scene semantics and surrounding dynamic objects in order to navigate safely.

While state-of-the-art 3D detection and tracking methods detect objects in top-down fashion (Fig. 1, *center*) and regress full object extent and orientation/velocity [1], [2], [3], [4], lidar instance and panoptic segmentation (Fig. 1, *left*) follow "bottom-up" segmentation-centric philosophy [5], [6], [7], [8], [9], that does not require reasoning about the full extent of 3D bounding boxes. Instead, these *segmentation-centric* first perform per-point semantic classification and then learn to group points corresponding to *thing* classes into instances in a bottom-up fashion.

In this paper, we question the established narrative that bottom-up grouping is the only design pattern for LPS and propose *MOST* (*MOdal Segmentation and Tracking*), a surprisingly simple yet effective *detection-centric* approach for *lidar panoptic segmentation* [10] and tracking [5]. Concretely, we base our method on CenterPoint [1], designed for top-down lidar-based 3D object detection. As in CenterPoint, we first encode a point cloud sequence using a sparse 3D convolutional backbone [2], [11], [1], [12] and flatten the bottleneck layer into a bird's eye view (BEV) representation of the point cloud that we use to detect objects as points. To obtain dense, per-point semantic interpretation and instance interpretation of point clouds, we add a 3D decoder head to our network that un-projects this representation and up-samples it back to the original resolution to perform per-voxel semantic classification. While 3D bounding boxes (*c.f.*, [1]) can directly be regressed from per-point bottleneck features, necessary fine-grained details needed for point-wise classification and instance segmentation are lost. Therefore, inspired by the instance segmentation branch of modern two-stage instance segmentation methods, we add a second-stage instance segmentation network that determines the *membership* of points to their respective instance centers provided by the instance recognition branch. Finally, we obtain spatio-temporal instance labels by additionally learning to regress modal offset vectors used for scan-to-scan instance association [1].

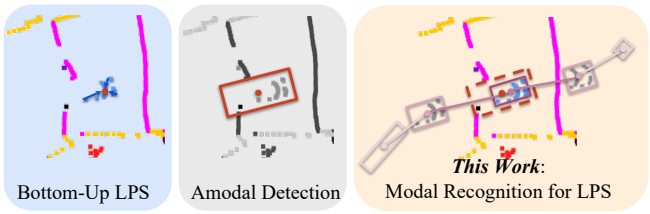


Fig. 1: State-of-the-art LPS methods (*left*) learn to group points in a bottom-up fashion, while state-of-the-art 3D object detectors (*center*) detect objects as *amodal* centers in the bird's-eye representation of the scene, followed by *amodal* 3D bounding box regression. In this paper, we re-purpose the *latter* for the *former*. Our method in parallel classifies points (*semantic segmentation*), detect *modal* instance centers (*modal instance recognition*) and their velocities (*modal instance tracking*).

The **motivation** for this design is two-fold. Firstly, LPS methods should maximize all aspects of the panoptic segmentation and tracking task, *i.e.*, (i) object recognition, (ii) instance segmentation, (iii) per-point semantic classification, and (iv) tracking, all explicitly captured via different modules in our network, supervised with corresponding loss functions. Second, this network is modular by design – while object detection, point classification, instance segmentation, and velocity regression components all share a common feature extractor, different components are disentangled. This, in principle, allows us to investigate the performance of each module separately, which is important for model interpretability that is crucial in robotics applications. Importantly, such a model can be trained in the future on multiple

datasets with different levels of supervision (as densely-labeled data is expensive to obtain), or replace different modules with stronger counterparts to boost the performance further.

Importantly, 3D detectors, such as CenterPoint, used as a base for our model, rely on *amodal* 3D bounding box supervision that enclose the full extent of the object, not only the visible portion. Such labels are not necessarily available in segmentation-centric semantic/panoptic segmentation datasets [13], [10].[1] To remedy this, we show we can leverage track-level (temporal) information to reason about the full extent of objects during the network training, and, as our experiments confirm, we alleviate the need for *amodal* labels. From this perspective, our method *MOST marries* object instance recognition and semantic segmentation in a single modular network suitable for 3D and 4D lidar panoptic segmentation and can be trained solely from temporal point-level (modal) supervision. This makes our method versatile enough for a thorough evaluation on multiple benchmarks for 3D/4D LPS on Panoptic nuScenes [14] and SemanticKITTI [13], [10] datasets.

In **summary**, (i) we propose a 3D/4D lidar segmentation network that unifies per-point semantic segmentation with modal object recognition and tracking in a single network. We (ii) detect instances via *modal* point-based temporal supervision and segment them with our novel binary instance segmentation network that determines point-to-detection membership based on BEV and per-point semantic features. Finally, we (iii) show the effectiveness of our method on various benchmarks for 3D/4D LPS. This confirms that our top-down approach based on modal recognition is highly effective for both 3D and 4D lidar panoptic segmentation and may directly impact design patterns used in future developments in this field of research. We will make code and our experimental data publicly available here.

II. Related Work

In this section, we summarize relevant work in 3D object detection, tracking, and semantic and panoptic segmentation for lidar point clouds.

Semantic segmentation. Advances in deep representation learning on unordered point sets [15] enable direct encoding of raw, unstructured point clouds to estimate fine-grained per-point semantic labels [15], [16], [17], [18]. Alternatively, several methods [19], [20], [21], [22], [23], [24] operate on a spherical projection of point cloud (*i.e.*, range images), and provide an excellent trade-off between accuracy and efficiency, important in robotics scenarios. State-of-the-art methods rely on voxel grids in conjunction with sparse convolutions [25], [26]. To efficiently encode sparse lidar point clouds, Cylinder3D [12] performs a cylindrical partition and proposes asymmetrical 3D convolution networks, followed by point refinement. We similarly adopt a sparse voxel grid-based backbone for point-based classification, a sub-task of panoptic segmentation.

Panoptic segmentation. Seminal methods for lidar panoptic segmentation follow a top-down approach inspired by the early image-based baselines [27]. These approaches train separate networks for semantic segmentation and object detection, followed by heuristic result fusion [10]. However, recent trends show that in the lidar-based domain bottom-up, proposal-free methods based on instance grouping in RGBD instance segmentation [28], [29], [30] and lidar panoptic segmentation [31], [5], [6], [7], [8], [32] obtain state-of-the-art results. Are bottom-up methods based on point grouping de-facto go-to approaches for lidar panoptic segmentation? We suggest this is not neccessarily the case.

4D lidar panoptic segmentation. Recently introduced 4D lidar panoptic segmentation [5], [14] extends lidar panoptic segmentation to the temporal domain, which requires sequence-level understanding. 4D-PLS [5] poses this task as bottom-up spatio-temporal point grouping, while MOPT [33] and CA-Net [34] segment instances in individual scans and associate them across time. Our proposed method is flexible and can generalize to utilize either single-scan or a multi-scan lidar sweep for both 3D and 4D lidar panoptic segmentation in one single unified network.

(A)modal object localization. Amodal bounding boxes (in 2D or 3D) encapsulate the full extent of the object, regardless of whether the full object is visible or not. This approach has origins in object detection [35] and requires the hallucination of bounding boxes during the annotation process. Recent works on 3D object detection ([1], [3], [11], [2], [36]) specifically utilizing amodal bounding boxes. Boxes can be hallucinated by annotators [35], obtained via linear interpolation in sequences [37] or in 3D using SLAM/structure from motion [38]. Alternatively, the recognition task can be posed as localization of the visible portion of the object (modal recognition), common in segmentation-centric tasks [39], [40], [13]. Modal annotations do not require hallucination of unobserved regions and are thus less sensitive to localization errors and less expensive in terms of annotation costs. In this paper, we demonstrate using modal annotations can achieve competitive performance compared to existing works built on amodal annotations.

III. MOST: Modal Segmentaton and Tracking

In this section, we present *MOST* (*MOdal Segmentation and Tracking*) for lidar panoptic segmentation of point clouds and point cloud sequences. Lidar panoptic segmentation methods must predict semantic class and a unique instance identity label for each point in a point cloud (sequence). This task is especially challenging in the temporal domain because objects may become occluded or may exit or (re)-enter the sensing area. We first cover an overview of *MOST*, followed by a discussion of all key components.

Overview. A visual overview of *MOST* is presented in Fig. 2. We base our network for lidar panoptic segmentation on an encoder-decoder-based U-net architecture [41]. In particular, we build on a sparse voxel grid-based backbone [11],

[1]With exception of nuScenes, which also includes *amodal* 3D object detection labels.

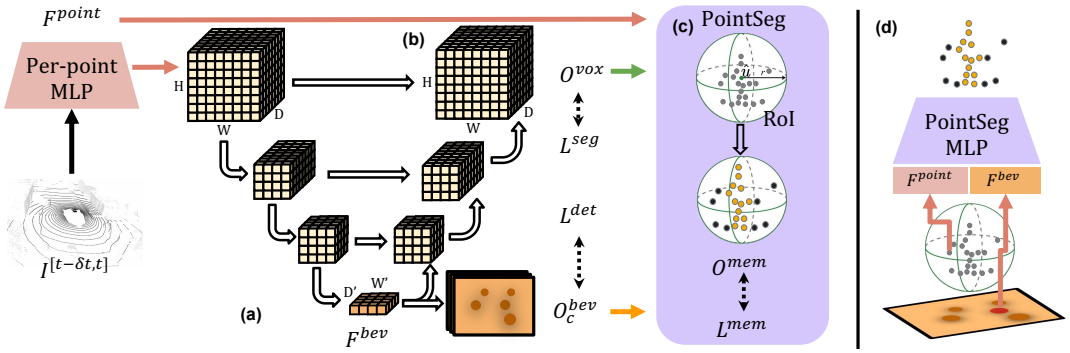


Fig. 2: *MOST* overview. We accumulate a point cloud sequence and encode it using a voxel grid-based encode-decoder backbone (*c.f.*, [11], [12]). After accumulating encoded points in a 3D voxel grid, (*a*) we down-sample the volume (via sparse 3D convolutions and pooling layers), and flatten the bottleneck layer along height-axis to obtain BEV representation (followed by 2D convolutional layers, similar to [1]). We use this representation to detect objects as (modal) points and regress (modal) offsets for temporal association. Our decoder (*b*) consists of several up-sampling layers to obtain fine-frained, voxel-level semantic predictions. Our instance segmentation network, *PointSegMLP* (*c*) performs binary classification within regions of interest (RoI) centered around predicted centers to obtain object instances. (*d*) *PointSegMLP* utilizes point and center features as input to produce panoptic segmentation results.

which encodes points with a shared multi-layer perceptron (MLP), and accumulates encoded points in voxels. We then apply several 3D sparse convolution layers to obtain a down-sampled BEV representation of the scene (Fig. 2*a*), *i.e.*, the bottleneck layer. Next, we detect object instances via the modal instance detection branch on top of the BEV representation. In parallel, our decoder upsamples the bottleneck layer back to the original voxel grid resolution via a series of 3D upsampling layers to obtain voxel-level semantic predictions (Fig. 2*b*). *Finally*, our instance segmentation network, PointSegMLP (Fig. 2*c*), determines which points belong to detected instances. To this end, PointSegMLP classifies points within instance-specific regions of interest (RoIs) centered around detected instances leading to panoptic segmentation predictions.

Network architecture. The input to our network is a point cloud $\mathscr{I}^t = \{(x,y,z,intensity),\dots\} \in N \times \mathbb{R}^4$, where N denotes the number of points. We accumulate input point clouds over a time window $[t-\delta t,t]$ to obtain *4D point cloud* $\mathscr{I}^{[t-\delta t,t]} \in N' \times \mathbb{R}^5$, wherein the last dimension encodes the relative time δt. We encode points using a MLP to obtain per-point features F^{point}, which we accumulate in a regular 4D voxel grid $\mathbb{R}^{C\times H\times W\times D}$, where H, W, D are dimensions of the bounding volume and C is the channel dimension. This 4D multichannel feature grid is processed with 3D convolutional encoders and decoders, following the sparse convolutional backbone of VoxelNet [11], which has proven successful for 3D object detection [2], [11], [1] and semantic segmentation [12].

For the *modal detection* branch, we flatten voxel features along their height to obtain a BEV feature map $F^{bev} \in \mathbb{R}^{C'\times W'\times D'}$. We then apply 2D convolutional layers to reduce the channel dimension to K output classes, followed by ReLU activation function to obtain *modal* center heatmaps $O_c^{bev} \in \mathbb{R}^{K\times W'\times D'}$, followed by non-maxima suppression to obtain a set of detected instances.

To obtain point-precise *semantic segmentation*, we up-sample the BEV feature map F^{bev} via upsampling layers back to a voxel-grid representation to obtain voxel-level logits O^{vox}. In a U-net fashion, we add skip connections from downsampling layers to capture fine-grained features. We then classify voxels via the softmax classifier.

Finally, to segment instances corresponding to predicted centers, we train an instance segmentation network (*PointSegMLP*) that predicts point-to-center *memberships*. Given a predicted center $\hat{c} \in \mathbb{R}^3$, we compute a binary membership for all points $p \in RoI(\hat{c})$. To this end, we concatenate per-point features F^{point} and predicted center BEV features F^{bev} for each point-center pair. Next, we use concatenated features to determine instance memberships $O^{mem}(p,\hat{c}) \in [0,1]$ for all pairs. We detail all components of our network in the following paragraphs.

Semantic segmentation. For voxel-level supervision, we obtain the supervisory signal from the current sweep $\mathscr{I}^t$ via majority voting to obtain $Y^{vox} \in \mathscr{K}^{H\times W\times D}$, where $\mathscr{K}$ denotes the set of all classes. Next, we apply per-voxel cross-entropy (CE) loss on-top of the voxel logits O^{vox}:

$$L_{seg} = CE(Y^{vox}, O^{vox}). \tag{1}$$

We note that even though we accumulate raw point clouds as $\mathscr{I}^{[t-\delta t,t]}$ as input to the encoder, the loss is only applied to voxels corresponding to $\mathscr{I}^t$. This is done by simply masking out loss corresponding to voxels in $\mathscr{I}^{[t-\delta t,t]}$ that do not belong to the current sweep $\mathscr{I}^t$. For point-level supervision, we utilize a point-refinement network [12]. We obtain voxel features O^{vox} at the point level, point features F^{point}, and BEV features O^{bev} and train a linear layer using cross-entropy loss. During inference, we assign point-level predictions O^{point} to all points within the voxel.

Modal object instance recognition. Assuming access to only per-point semantic class and instance IDs for objects, we represent objects via statistics computed from observed

points. More precisely, for a visible set of points $\mathscr{P}$ representing an instance, we define *modal* center $c \in \mathbb{R}^3$ as the mean of $\mathscr{P}$, and *modal* extent $r \in \mathbb{R}^3$ as the maximum distance of a point p from c. Intuitively, modal extent encodes the visible extent of an object. Following [1], we obtain BEV supervisory signal by constructing K class-wise center heatmaps $Y_c^{bev} \in \mathbb{R}^{W' \times D'}$. We project modal centers and modal extents from 3D into the 2D BEV plane. Then, we place a 2D Gaussian centered at the projected modal centers with the projected radius as the variance. Since the projection collapses the center height information, we set an additional regression target $Y_h^{bev} \in \mathbb{R}^{W' \times D'}$ to localize the object in 3D. We then apply focal loss [42] for *thing* classes as in CenterPoint [1] for modal centers heatmaps O_c^{bev}. In addition we apply an L1 regression loss on height O_h^{bev}:

$$L_{det} = FocalLoss(Y_c^{bev}, O_c^{bev}) + |Y_h^{bev} - O_h^{bev}|. \quad (2)$$

Estimating modal extent r. We first compute the extent for instances at each time step via shrink-wrapping (SW). To this end, we estimate tight axis-aligned bounding boxes that enclose observed points. We compute SW axis-aligned box at time t as: $r_t = \max\{|p-c|, p \in \mathscr{P}\}$, where $\mathscr{P}$ represents the set of observed points for this instance.

Intuitively, humans reason about the object extent by fusing information from multiple viewpoints. A sensor mounted on an autonomous vehicle similarly observes objects from different viewpoints over time. Therefore, we can derive more accurate object extent estimates r by reasoning about object size over time. We utilize unique instance IDs, available in 4D panoptic segmentation datasets [10], [14] to obtain refined object extent estimates through *temporal supervision*. To this end, we simply compute the maxima of all per-frame extent estimates for an object across time (MAX) to obtain a more precise estimate compared to naive SW: $r = \max\{r_t\}_{t=0}^{T}$. These are then used as target extents during the modal instance recognition branch training. This approach is especially beneficial for instances that contain only a few points—for example, cases where only a vehicle's front or rear bumper is visible.

PointSegMLP for instance segmentation. The modal recognition branch provides object center predictions, while the semantic decoder independently provides point-wise semantic predictions. The next step is to obtain *instance-level* segmentation, *i.e.*, modal point-precise segmentation masks for detected instances. We tackle instance segmentation by training a *point membership* function, *PointSegMLP*, that determines for each point in the scene *which* points correspond to *which* detected centroids. This is analogous to image-based two-stage instance segmentation networks [43], that segment instances via binary classification for each anchor box.

In theory, we could obtain segmentation masks by applying our membership function to each point-center pair, leading to computational complexity of $\mathscr{O}(N \times P)$, where N and P denote the number of points and This way, our method does not rely on any amodal labels, while reducing the time and memory complexity ($\mathscr{O}(N' \times P)$, where $N' << N$) to compute membership at both training time and inference. As a side effect, this also leads to faster convergence during training, as samples within the ball can be considered as the "hard samples".

Consider a detected object instance D with center $\hat{\mu}$ and class $\hat{k}$. We take all points $\mathscr{P} = \{p \in RoI(\hat{\mu}, \hat{k})\}$ as *in-range* points. Next, we obtain features for the detected center and for all in-range points p (see Fig. 2*d*). The center features are comprised of its 3D position $\hat{\mu}$, semantic class $\hat{k}$ and BEV features $F_{\hat{\mu}}^{bev}$. The BEV features $F_{\hat{\mu}}^{bev}$ are obtained by first projecting $\hat{\mu}$ onto the BEV plane, followed by linear interpolation of F^{bev} features at the projected point. Similarly, we compute point features using its 3D position $p \in \mathscr{P}$, predicted semantic label O_p^{point} and the BEV features F_p^{bev} for point p. In addition, we append F_p^{point} features obtained from the backbone. The obtained center features are concatenated with the point features to obtain a feature representation for a point-center pair. Our per-point *PointSegMLP*, shared across all points, utilizes the obtained point-center features to determine per-point-to-center instance membership $O^{mem} \in [0,1]$. The architecture boils down to a light-weight MLP comprised of fully connected layers with batch normalization and ReLU activation. For supervision, we construct a binary ground truth membership Y^{mem}, points belonging to an instance D are assigned 1, and others 0. We train *PointSegMLP* using binary cross-entropy loss (BCE):

$$L_{mem} = BCE(Y^{mem}, O^{mem}). \quad (3)$$

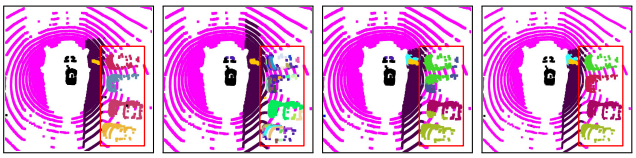
Modal panoptic tracking. We follow CenterPoint [1] and concatenate point clouds before encoding them. Such spatio-temporal representations can be used to regress offset vectors $v \in \mathbb{R}^2$ and to obtain sequence-level lidar panoptic segmentation through greedy association. The difference, however, is that we estimate offset vectors using modal labels only. We construct ground truth velocity offsets Y_v^{bev} for input point cloud $\mathscr{I}^t$ using $\mathscr{I}^{t-\delta t}$ and $\mathscr{I}^{t+\delta t}$. For each object, ground truth velocity offsets are computed through a centered difference between modal centers *i.e.*, $(\mu^{t+\delta t} - \mu^{t-\delta t})/(2\delta t)$. The velocity offset predictions O_v^{bev} when combined with per-point 3D panoptic segmentation leads to a unified, single-network top-down approach to 4D lidar panoptic segmentation. We train the velocity offset regression head using L1 loss:

$$L_{track} = |Y_v^{bev} - O_v^{bev}|. \quad (4)$$

Putting everything together. We train our network by minimizing the overall training objective, that is composed of modal detection loss L_{det}, semantic segmentation loss L_{seg}, instance segmentation loss L_{mem}, and optionally for sequences, modal velocity regression loss L_{track}:

$$L_{total} = L_{det} + L_{seg} + L_{mem} + L_{track}. \quad (5)$$

Inference. During inference, we fuse segmentation branch predictions O^{vox}, modal center heatmaps O^{bev}, and point-



(a) GT (b) DS-Net (c) NN-Baseline (d) *MOST*

Fig. 3: **Qualitative results:** We compare GT labels (Fig. 3a) to two variants of our method (Fig. 3c&3d) and DS-Net (Fig. 3b). As shown, DS-Net [7] struggles with correctly segmenting multiple cars, leading to over-segmentation. Our *modal* detection centric approach performs significantly better, however, with simple nearest neighbor heuristic (NN-baseline, Fig. 3c) we cannot correctly assign/segment *fuzzy* points. By contrast, our full method, *MOST* (Fig. 3d) correctly detects *and* segments cars. Best viewed in color.

center memberships O^{mem} to obtain 4D panoptic predictions. Intuitively, we listen to the segmentation labels predicted by the segmentation branch, and listen to instance labels predicted by the modal centroid membership branch. We summarize the approach here: using the segmentation branch predictions O^{vox}, we assign point-level predictions O^{point} to all points within the voxel. Similarly, we apply non-maximum supression (NMS) over the predicted center heatmaps O^{bev} to generate predicted modal centers $\hat{\mu}$. We then compute the membership of each point p within the RoI of each center using *PointSegMLP*, resolving overlapping RoIs by the most confident center $\hat{u}^* = \text{argmax}_{\hat{u}}(O^{mem}(p,\hat{u}))$. Next, we assign the predicted center label to all points that are its members, and assign a unique instance id. For all *stuff* points, we directly utilize the predicted semantic label. To extend our method to panoptic tracking, we associate instances across sweeps by using predicted center velocities O^{bev}_{v}, following the approach in [1]: we greedily form *tracklets* by matching previous-sweep centers to current sweep centers with subtracted velocity offsets. Finally, all the instances of an object belonging to a *tracklet* are assigned a temporally consistent unique ID.

Implementation details. We train our network in two stages. In the first stage, we optimize the modal detection and segmentation branch using L_{det}, L_{seg}, and L_{track}. Next, we freeze the first stage network and only train the second stage using a per-point L_{mem} loss. The first stage network is trained with Adam optimizer with a learning rate of $1e^{-3}$, with a batch size of 8. For the second stage training of *PointSegMLP*, we utilize SGD optimizer with a learning rate as $5e^{-4}$. The network is trained for a total of 20 epochs. Architecture: the per-point feature extraction layer and *PointSegMLP* are simple 4-layer MLPs with BatchNorm and ReLU layers. The voxel grid encoder downsamples the input point cloud by a factor of 8, while the decoder upsamples the bottleneck layer back to the original resolution. The modal detection branch comprises of two 3x3 convolution layers with ReLU activation layers. We accumulate previous 10 frames. We do not employ any test time augmentation while reporting our results.

TABLE I: **Different strategies for amodalization:** We compare different methods using the same segmentation backbone. We start with bottom-up centroid regression, as done in DS-Net. Naively creating a shrink-wrapped (SW) modal cuboid and training a top-down object-centric modal detector already improves performance, with or without sharing weights between the detection and segmentation branches. We also explore other ways of generating modal cuboids, including using a class-wise mean (CWM) cuboid dimension or taking the max dimension value across time (MAX). We see MAX performing the best when only modal annotations are available, while closing the gap with amodal annotations. Best results are **bolded**, while second best are underlined.

Method	PQ	PQ^{Th}	PQ^{St}	mIoU	$mIoU^{Th}$	$mIoU^{St}$
Offset reg. (DS-Net)	68.2	66.1	71.6	75.5	71.8	81.7
Modal Det. (SW), w/o sharing	72.2	71.7	72.9	77.2	74.5	81.7
Modal Det. (SW), w/ sharing	73.8	74.4	73.1	79.7	78.7	81.3
Modal Det. (DSB)	70.1	68.4	72.9	77.6	75.2	81.3
Modal Det. (CWM)	74.2	74.6	73.5	79.7	78.7	81.3
Modal Det. (MAX)	77.1	79.3	**73.6**	**80.3**	**79.4**	**81.7**
Amodal Det.	**78.1**	**82.7**	72.9	79.2	78.8	79.7

TABLE II: **Different membership functions:** We compare our *PointSegMLP* with a simple, yet surprisingly effective nearest neighbor heuristic (*NN-baseline*). As can be seen, our learning-based *PointSegMLP* based on 3D positions and semantic predictions already significantly outperforms the heuristic. While instance bird's-eye-view (BEV) features in isolation do not benefit our model, they further improve the performance when combined with per-point semantic features.

Method	PQ	mIoU	Membership Acc (%)
Nearest Neighbor (baseline)	70.9	77.2	76.3
Ours (semantic + geometric)	72.9	77.5	**95.4**
+ BEV feat.	72.6	77.4	95.0
+ Point feat.	**73.8**	**79.7**	**95.4**

IV. Experimental Evaluation

In this section, we first summarize our evaluation test-bed, including datasets, benchmarks and evaluation metrics used to conduct our experiments (Sec. IV-A). We next ablate various stages and design decisions of *MOST*'s network architecture for joint lidar panoptic segmentation and tracking (Sec. IV-B). Finally, we outline and discuss official benchmark results obtained on singe-scan and multi-scan (4D) lidar panoptic segmentation (Sec. IV-C).

A. Evaluation Test-Bed

Datasets. We evaluate our work using SemanticKITTI [13], [10] and Panoptic nuScenes [14] datasets, that contain per-point temporally-consistent semantic and instance labels. SemanticKITTI [13], [10] contains $1.5h$ of lidar scans, recorded using a 64-beam sensor, and labels for 28 semantic classes. Panoptic nuScenes [14] contains $1,000$ short scenes with a 32-beam sensor. It contains labels for 32 semantic classes, with a labeling frequency of 2Hz. For both datasets, we

follow the official splits for training/validation and evaluate our final models on the hidden test set.

Tasks and evaluation metrics. For *single-scan (3D) lidar panoptic segmentation*, we report well-established panoptic quality **PQ** metric [27]. In a nutshell, PQ is a soft version of the F1-score that treats *thing* and *stuff* classes in a unified manner. Following the official evaluation procedure, we set the minimum number of points on an instance to be 15 for nuScenes and 50 for SemanticKITTI. We additionally report mean intersection-over-union (**mIoU**) [35] that evaluates per-point semantic segmentation. For *multi-scan (4D) lidar panoptic segmentation i.e.*, *panoptic tracking*, we report (lidar) segmentation and tracking quality **LSTQ** [46], [5], evaluated as the geometric mean between mIoU and point association quality (**AQ**). AQ evaluates whether a point was associated with the correct instance in space and time. For Panoptic nuScenes, we additionally report the recently introduced panoptic tracking (**PAT**) metric, which combines PQ and LSTQ.

B. Ablations

Modal recognition or center-offset regression? We first study the impact of our top-down modal recognition-based approach to panoptic segmentation and compare it to a bottom-up center-offset regression approach by DS-Net [7], for which code is available. This method predicts *center offsets* followed by mean-shift clustering to obtain object instances. For comparison, we take the identical semantic segmentation network (Cylinder3D [12]), but instead of offset regression and mean-shift clustering, we train a *separate* modal instance recognition network, followed by our proposed instance segmentation network. We refer to this variant as *Modal Det. (SW) w/o sharing* in Tab. I. As the semantic segmentation networks are identical, this experiment highlights the effectiveness of our modal detection branch. In this setting, we regress tightly-fitting "shrink-wrap (SW)" bounding boxes derived from segmentation labels. With this simple approach, we improve upon the bottom-up baseline for $+4$ PQ points.

Joint training. Next, we train a single network for *joint* semantic segmentation and modal instance recognition and segmentation (as explained in Sec. III), *i.e.*, *Modal Det. (SW) w/ sharing* in Tab. I. This yields *PQ* score of 73.8 ($+0.6$), confirming the benefits of joint training of segmentation and modal recognition networks.

Amodal recognition? Next, we evaluate the impact of *amodal* training on our detection component. Entry *Amodal Det.* in Tab. I refers to a variant that trains the detection branch using such *amodal* labels. We utilize Panoptic nuScenes [14] dataset since it contains both segmentation labels and amodal boxes. We obtain PQ of 78.1, $+4.3$ points compared to our SW *modal* baseline. This is not surprising: *amodal* labels provide additional supervisory signals in the form of orientation and full extent for each instance. This suggests that *amodal* labels contain additional useful information that comes with additional annotation cost. *This begs the question, can we close the gap using only segmentation-level labels?*

Closing the gap. While regressing the full extent of the object is beneficial, we do not have access to this information (*e.g.*, in SemanticKITTI [13]). Can we do better than naïve *shrink-wrapping* baseline (73.8 PQ)? First, we observe this gap is due to the sensitivity of the modal recognition head to occlusions and decreasing sensor resolution, resulting in a large number of instances containing only a few points. An obvious remedy is to simply drop small boxes (DSB), *i.e.*, to exclude small tightly-fitting bounding boxes from training. However, this approach yields 70.1 PQ, likely due to the exclusion of a large portion of training data. A better strategy is to replace small boxes with *class-wise mean* (CWM) box sizes, which yields 74.2 PQ. Finally, we utilize the sequential information and compute tightly-fitting boxes throughout the instance trajectory. Taking max dimension over time for supervision (MAX) produces PQ of 77.1, which is reasonably close to full *amodal* supervision. This implies that once we have sequential point-level labels, we produce better estimates of object extent and use that for supervision.

PointSegMLP. Next, we justify design decisions behind our instance segmentation network, *PointSegMLP*, that predicts binary point-center memberships for each detected object instance. In addition to reporting standard *PQ* and *mIoU*, we also evaluate *membership accuracy* (*mem. acc.*), which evaluates how many points within a given RoI were correctly classified, *i.e.*, the percentage of points that have been assigned to the correct center.

We first evaluate the performance of a simple geometric baseline, denoted as *NN-baseline* in Tab. II. This method assigns each point to its nearest *semantically-compatible* detected center within the RoI of the detected center. We conclude that this simple baseline works surprisingly well, obtaining a *PQ* of 70.9. However, clearly, there is space for improvement in terms of *mem. acc.* (76.3). While most points can be unambiguously assigned to the nearest instance, a certain percentage of *fuzzy* points could be assigned to two or more instances. This motivates the usage of data-driven *PointSegMLP* to perform segmentation.

Next, we compare this baseline to our *PointSegMLP* that learns to segment points. In the first variant (denoted as *semantic + geometric*), we only utilize the 3D point coordinates of points and instance centers, along with their semantic predictions. We observe that just using geometrical features significantly outperforms the *NN-baseline* with an improvement of $+19.1$ in terms of *mem.acc*, and this translates into $+2$ improvement in *PQ*. Solely adding bird-eye-view (BEV) feature (*wrt.* detected instance) does not improve the performance, however, adding both instance *BEV* feature and fine-grained *point* features leads to a $+0.9$ increase in *PQ*. We also visualize the results in Fig. 3.

C. Benchmark results

This section compares our method to published state-of-the-art methods on standard benchmarks for 3D and 4D

TABLE III: **LPS on nuScenes panoptic:** *MOST* is 1^{st} nuScenes val set and 2^{nd} on the test set. The best results are **bolded**, while the second best are underlined.

	Method	PQ	PQ†	RQ	SQ	PQ^{Th}	RQ^{Th}	SQ^{Th}	PQ^{St}	RQ^{St}	SQ^{St}	mIoU
Validation	DS-Net [7]	51.2	-	59.0	86.1	38.4	43.8	86.7	72.3	84.2	85.0	73.5
	TORNADO-Net [44]	54.0	59.8	65.4	80.9	44.1	53.9	80.1	63.9	76.9	81.8	68.0
	GP-S3Net [8]	61.0	67.5	72.0	84.1	56.0	65.2	85.3	66.0	78.7	82.9	75.8
	Efficient-LidarPanopticSegmentation [45]	62.0	65.6	73.9	83.4	56.8	68.0	83.2	70.6	83.6	83.8	65.6
	PolarSeg-Panoptic [32]	63.4	67.2	75.3	83.9	59.2	70.3	84.1	70.4	83.5	83.6	66.9
	Panoptic-PHNet [9]	74.7	-	84.2	88.2	74.0	-	-	**75.9**	-	-	79.7
	Ours	**77.1**	**79.9**	**86.5**	**88.6**	**79.3**	**87.5**	**90.3**	73.6	**84.9**	**85.7**	**80.3**
Test	Efficient-LidarPanopticSegmentation [45]	62.4	66.0	74.1	83.7	57.2	68.2	83.6	71.1	84.0	83.8	66.7
	PolarSeg-Panoptic [32]	63.6	67.1	75.1	84.3	59.0	69.8	84.3	71.3	83.9	84.2	67.0
	Panoptic-PHNet [9]	**80.1**	**82.8**	**87.6**	**91.1**	**82.1**	**88.1**	**93.0**	**76.6**	**86.6**	**87.9**	80.2
	Ours	76.1	79.5	85.1	88.9	77.4	85.5	90.3	73.9	84.5	86.7	**80.4**

TABLE IV: **LPS on SemanticKITTI:** *MOST* is a close 2^{nd} runner. Our method builds on the same encoder-decoder backbone as DS-Net; however, it significantly improves the results in terms of *PQ* and *mIoU*, thanks to our modal recognition branch and instance segmentation network. The best results are **bolded**, while the second best are underlined.

	Method	PQ	PQ†	RQ	SQ	PQ^{Th}	RQ^{Th}	SQ^{Th}	PQ^{St}	RQ^{St}	SQ^{St}	mIoU
Validation	Panoster [6]	55.6	-	66.8	79.9	56.6	65.8	-	-	-	-	61.1
	DS-Net [7]	57.7	63.4	68.0	77.6	61.8	68.8	78.2	54.8	67.3	77.1	63.5
	PolarSeg-Panoptic [32]	59.1	64.1	70.2	78.3	65.7	74.7	**87.4**	54.3	66.9	71.6	64.5
	Efficient-LidarPanopticSegmentation [45]	59.2	65.1	69.8	75.0	58.0	68.2	78.0	**60.9**	71.0	72.8	64.9
	GP-S3Net [8]	**63.3**	**71.5**	**75.9**	**81.4**	**70.2**	**80.1**	86.2	58.3	**72.9**	**77.9**	**73.0**
	Ours	63.1	70.8	73.1	79.2	68.7	75.7	86.7	58.9	71.2	73.7	69.7
Test	Panoster [6]	52.7	59.9	64.1	80.7	49.4	58.5	83.3	55.1	68.2	78.8	59.9
	PolarSeg-Panoptic [32]	54.1	60.7	65.0	81.4	53.3	60.6	87.2	54.8	68.1	77.2	59.5
	DS-Net [7]	55.9	62.5	66.7	82.3	55.1	62.8	87.2	56.5	69.5	78.7	61.6
	Efficient-LidarPanopticSegmentation [45]	57.4	63.2	68.7	83.0	53.1	60.5	87.8	60.5	74.6	79.5	61.4
	GP-S3Net [8]	60.0	**69.0**	**72.1**	82.0	**65.0**	**74.5**	86.6	56.4	70.4	78.7	**70.8**
	Panoptic-PHNet [9]	**61.5**	67.9	**72.1**	**84.8**	63.8	70.4	**90.7**	59.9	73.3	80.5	66.0
	Ours	61.0	66.8	72.0	84.4	58.1	66.0	88.1	**63.2**	**76.3**	**81.7**	66.1

TABLE V: **4D Lidar Panoptic Segmentation Benchmarks**. Our method is 1^{st} on nuScenes and 3^{rd} on SemanticKITTI. Best results are **bolded**, while second best are underlined. MOT: tracking-by-detection [47], SFP: scene flow based propagation [48], PP: PointPillars 3D detector [3].

	Method	*LSTQ*	*PAT*	S_{assoc}	S_{cls}	*PTQ*	*PQ*
SemanticKITTI	RangeNet++ [21] + PP + MOT	35.5	-	24.1	52.4	-	-
	KPConv [17] + PP + MOT	38.0	-	25.9	55.9	-	-
	RangeNet++ [21] + PP + SFP	34.9	-	23.3	52.4	-	-
	KPConv [17] + PP + SFP	38.5	-	26.6	55.9	-	-
	4D-PLS [5]	56.9	-	56.4	57.4	-	-
	Contrastive Association [34]	63.1	-	65.7	60.6	-	-
	4D-StOP [49]	**63.9**	-	**69.5**	58.8	-	-
	Ours	60.3	-	57.8	**62.8**	-	-
nuScenes	PanopticTrackNet [33]	44.8	45.7	36.7	58.9	51.6	51.7
	4D-PLS [5]	57.8	60.5	53.6	62.3	55.6	56.6
	E-LPS [45] + Kalman	63.7	67.1	60.2	67.4	62.3	63.6
	E-LPT [45]	66.4	70.4	-	69.5	67.5	67.9
	Ours	**73.2**	**74.9**	**66.6**	**80.4**	**72.0**	**76.0**

lidar panoptic segmentation on Panoptic nuScenes [14] and SemanticKITTI [13] datasets.

Lidar panoptic segmentation. We report the results for panoptic segmentation in Tab. III (Panoptic nuScenes) and Tab. IV (SemanticKITTI). We utilize temporal supervised (MAX) for obtaining object targets for the benchmark submission. We focus this discussion on the test set results and show results on the validation set for completeness. On the nuScenes dataset, *MOST* is the second-best method with 76.1 *PQ*. Note that we only utilize standard convolution layers as opposed to transformers in state-of-the-art Panoptic-PHNet [9], so there is potential to replace our lightweight components with stronger transformer-based counterparts to achieve better performance. Moreover, Panoptic-PHNet [9] could not be easily extended for sequence-level scene understanding (i.e. tracking), while as we will show next, *MOST* can achieve competitive performance on tracking by simply appending a greedy association module. *MOST* outperforms other approaches by a large margin (+13.5 *PQ*). On Semantic-KITTI, *MOST* is a close-second obtaining 61.0 *PQ*, with state-of-the-art being 61.5 *PQ*. This highlights that *MOST* generalizes well across different datasets.

Lidar panoptic tracking. We report the results for 4D lidar panoptic tracking on SemanticKITTI and nuScenes datasets in Tab. V. Being a top-down method, *MOST* can easily extend to 4D panoptic segmentation through the greedy association of predicted velocity offsets. On nuScenes, *MOST* obtains 73.2 LSTQ and 74.9 PAT on the test set, establishing new state-of-the-art on this benchmark. *MOST* improves by +6.8 LSTQ and +4.5 PAT points over second-best approach, *Efficient-LPT* [45]. On Semantic-KITTI, *MOST* obtains competitive results (60.3 LSTQ) with a simple greedy approach. These results affirm that *MOST* is a versatile approach that performs consistently across different benchmarks, 3D and 4D panoptic segmentation on multiple datasets.

V. Conclusions

This paper presents a *top-down* approach to lidar panoptic segmentation and tracking using only *modal* annotations. Our unified network jointly detects objects as modal points and classifies voxels to obtain per-point panoptic segmentation predictions. Instances are associated across 4D spatio-temporal data using learned modal velocity offsets to obtain panoptic tracking predictions. Our method establishes a new state-of-the-art on Panoptic nuScenes 4D panoptic segmentation benchmark. We hope that this work, along with publicly release code and data will inspire future developments in recognition-centric methods for lidar panoptic segmentation and tracking.

References

[1] T. Yin, X. Zhou, and P. Krähenbühl, "Center-based 3d object detection and tracking," in *CVPR*, 2021.

[2] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[3] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *CVPR*, 2019.

[4] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud," in *CVPR*, 2019.

[5] M. Aygün, A. Ošep, M. Weber, M. Maximov, C. Stachniss, J. Behley, and L. Leal-Taixé, "4d panoptic lidar segmentation," in *CVPR*, 2021.

[6] S. Gasperini, M.-A. N. Mahani, A. Marcos-Ramiro, N. Navab, and F. Tombari, "Panoster: End-to-end panoptic segmentation of lidar point clouds," *RAL*, 2021.

[7] F. Hong, H. Zhou, X. Zhu, H. Li, and Z. Liu, "Lidar-based panoptic segmentation via dynamic shifting network," in *CVPR*, 2021.

[8] R. Razani, R. Cheng, E. Li, E. Taghavi, Y. Ren, and L. Bingbing, "Gp-s3net: Graph-based panoptic sparse semantic segmentation network," in *CVPR*, 2021.

[9] J. Li, X. He, Y. Wen, Y. Gao, X. Cheng, and D. Zhang, "Panoptic-phnet: Towards real-time and high-precision lidar panoptic segmentation via clustering pseudo heatmap," in *CVPR*, 2022.

[10] J. Behley, A. Milioto, and C. Stachniss, "A Benchmark for LiDAR-based Panoptic Segmentation based on KITTI," in *ICRA*, 2021.

[11] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *CVPR*, 2018.

[12] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin, "Cylindrical and asymmetrical 3d convolution networks for lidar segmentation," in *CVPR*, 2021.

[13] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *ICCV*, 2019.

[14] W. K. Fong, R. Mohan, J. V. Hurtado, L. Zhou, H. Caesar, O. Beijbom, and A. Valada, "Panoptic nuscenes: A large-scale benchmark for lidar panoptic segmentation and tracking," *arXiv preprint arXiv:2109.03805*, 2021.

[15] 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.

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

[17] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *ICCV*, 2019.

[18] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," in *CVPR*, 2020.

[19] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," in *ICRA*, 2018.

[20] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, "Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud," in *ICRA*, 2019.

[21] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet++: Fast and Accurate LiDAR Semantic Segmentation," in *IROS*, 2019.

[22] E. E. Aksoy, S. Baci, and S. Cavdar, "Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving," in *IVS*, 2020.

[23] R. Razani, R. Cheng, E. Taghavi, and L. Bingbing, "Lite-hdseg: Lidar semantic segmentation using lite harmonic dense convolutions," in *ICRA*, 2021.

[24] S. Li, X. Chen, Y. Liu, D. Dai, C. Stachniss, and J. Gall, "Multi-scale interaction for real-time lidar data segmentation on an embedded platform," *RAL*, vol. 7, no. 2, pp. 738–745, 2021.

[25] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal convnets: Minkowski convolutional neural networks," in *CVPR*, 2019.

[26] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han, "Searching efficient 3d architectures with sparse point-voxel convolution," in *ECCV*, 2020.

[27] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *CVPR*, 2019.

[28] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia, "Pointgroup: Dual-set point grouping for 3d instance segmentation," in *CVPR*, 2020.

[29] L. Han, T. Zheng, L. Xu, and L. Fang, "Occuseg: Occupancy-aware 3d instance segmentation," in *CVPR*, 2020.

[30] F. Engelmann, M. Bokeloh, A. Fathi, B. Leibe, and M. Nießner, "3D-MPA: Multi Proposal Aggregation for 3D Semantic Instance Segmentation," in *CVPR*, 2020.

[31] A. Milioto, J. Behley, C. McCool, and C. Stachniss, "Lidar panoptic segmentation for autonomous driving," in *IROS*, 2020.

[32] Z. Zhou, Y. Zhang, and H. Foroosh, "Panoptic-polarnet: Proposal-free lidar point cloud panoptic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 194–13 203.

[33] J. V. Hurtado, R. Mohan, and A. Valada, "Mopt: Multi-object panoptic tracking," *arXiv preprint arXiv:2004.08189*, 2020.

[34] R. Marcuzzi, L. Nunes, L. Wiesmann, I. Vizzo, J. Behley, and C. Stachniss, "Contrastive instance association for 4d panoptic segmentation using sequences of 3d lidar scans," *RAL*, 2022.

[35] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *IJCV*, vol. 88, no. 2, pp. 303–338, 2010.

[36] Z. Liu, Z. Zhang, Y. Cao, H. Hu, and X. Tong, "Group-free 3d object detection via transformers," in *ICCV*, 2021.

[37] P. Dendorfer, A. Ošep, A. Milan, K. Schindler, D. Cremers, I. Reid, and S. R. L. Leal-Taixé, "Motchallenge: A benchmark for single-camera multiple target tracking," *IJCV*, 2020.

[38] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *CVPR*, 2012.

[39] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *ECCV*, 2014.

[40] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. Sekar, A. Geiger, and B. Leibe, "MOTS: Multi-object tracking and segmentation," in *CVPR*, 2019.

[41] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[42] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *ICCV*, 2017.

[43] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *ICCV*, 2017.

[44] M. Gerdzhev, R. Razani, E. Taghavi, and L. Bingbing, "Tornado-net: multiview total variation semantic segmentation with diamond inception module," in *ICRA*, 2021.

[45] K. Sirohi, R. Mohan, D. Büscher, W. Burgard, and A. Valada, "Efficientlps: Efficient lidar panoptic segmentation," *IEEE Transactions on Robotics*, 2021.

[46] M. Weber, J. Xie, M. Collins, Y. Zhu, P. Voigtlaender, H. Adam, B. Green, A. Geiger, B. Leibe, D. Cremers, , A. Ošep, L. Leal-Taixé, and L.-C. Chen, "Step: Segmenting and tracking every pixel," *arXiv preprint arXiv:2102.11859*, 2021.

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

[48] H. Mittal, B. Okorn, and D. Held, "Just go with the flow: Self-supervised scene flow estimation," in *CVPR*, 2020.

[49] L. Kreuzberg, I. E. Zulfikar, S. Mahadevan, F. Engelmann, and B. Leibe, "4d-stop: Panoptic segmentation of 4d lidar using spatio-temporal object proposal generation and aggregation," in *ECCV AVVision Workshop*, 2022.