Mask-Based Panoptic LiDAR Segmentation for Autonomous Driving

Rodrigo Marcuzzi Lucas Nunes Louis Wiesmann Jens Behley Cyrill Stachniss

***Abstract*—Autonomous vehicles need to understand their surroundings geometrically and semantically to plan and act appropriately in the real world. Panoptic segmentation of LiDAR scans provides a description of the surroundings by unifying semantic and instance segmentation. It is usually solved in a bottom-up manner, consisting of two steps. Predicting the semantic class for each 3D point, using this information to filter out "stuff" points, and cluster "thing" points to obtain instance segmentation. This clustering is a post-processing step with associated hyperparameters, which usually do not adapt to instances of different sizes or different datasets. To this end, we propose MaskPLS, an approach to perform panoptic segmentation of LiDAR scans in an end-to-end manner by predicting a set of non-overlapping binary masks and semantic classes, fully avoiding the clustering step. As a result, each mask represents a single instance belonging to a "thing" class or a "stuff" class. Experiments on SemanticKITTI show that the end-to-end learnable mask generation leads to superior performance compared to state-of-the-art heuristic approaches. The implementation of our approach is available at https://github.com/PRBonn/MaskPLS.**

***Index Terms*—Semantic Scene Understanding, Deep Learning Methods**

I. Introduction

SEMANTIC scene understanding is a key requirement for autonomous vehicles navigating in dynamic environments. Panoptic segmentation [24] seeks to jointly achieve semantic segmentation to semantically interpret the surrounding and instance segmentation to identify object instances. On LiDAR point clouds, panoptic segmentation is usually solved in a bottom-up way by predicting different kind of information, such as offset vectors [20], embedding vectors [33], and/or object centers [2] to enable and guide the clustering of instances using off-the-shelf clustering approaches [13], [8]. The clustering algorithms need hyperparameter tuning that depends on the data and does not allow end-to-end training of the model. For images, recent work [11], [10], [43] solves this problem by directly predicting a set of binary masks and semantic classes but this paradigm has not yet been applied for 3D LiDAR point clouds, where the distance-dependent sparsity might pose a challenge for the aforementioned approaches.

In this work, we propose to remove the clustering step and perform panoptic segmentation on 3D LiDAR point clouds by directly predicting a set of binary masks and semantic classes. The method is end-to-end trainable and does not require hyperparameter tuning to work on different data.

Manuscript received: Aug 24, 2022; Revised: Nov 2, 2022; Accepted: Dec 24, 2022. This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and Reviewers' comments.

All authors are with the University of Bonn, Germany. Cyrill Stachniss is additionally with the Department of Engineering Science at the University of Oxford, UK, and with the Lamarr Institute for Machine Learning and Artificial Intelligence, Germany. Corresponding author: rmarcuzzi@uni-bonn.de

Digital Object Identifier (DOI): see top of this page.

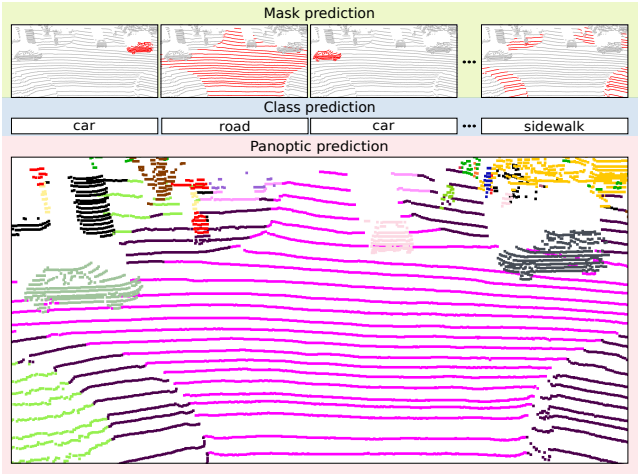


Fig. 1: Panoptic segmentation by predicting binary masks and their corresponding semantic class. Each mask represents a single instance of a "thing" class, i.e., car or a complete "stuff" class, i.e., road, sidewalk. Top: predicted binary masks. Middle: predicted mask class. Bottom: panoptic prediction by merging the masks.

Inspired by recent work on images [9], [11], we adopt a backbone network for feature extraction, a set of learnable feature vectors as mask proposals and a transformer decoder to predict non-overlapping binary masks and their semantic classes like depicted in Fig. 1. Each mask represents a single instance of a "thing" class or a complete "stuff" class. We base our approach on the work by Cheng et al. [10]. Compared to images, we have to deal with the sparsity of the LiDAR data while avoiding the high memory requirements of 3D convolutions. Therefore, we use a backbone based on 3D sparse convolutions to extract multi-scale features. A modified transformer decoder combines learnable queries and features from different resolutions through masked attention [10]. We generate the final 3D masks by combining the output queries of the decoder and the full resolution point features. To deal with the voxelization effects, we compute point-wise features through interpolation instead of directly feeding voxel features to the decoder. To improve the quality of 3D masks, we merge semantic and spatial information by adding fixed positional encodings to full-resolution point features.

The main contribution of this paper is a method to perform panoptic segmentation by predicting a set of binary masks and semantic classes that can be trained end-to-end without any heuristic post-processing steps, like clustering. To the best of our knowledge, we present the first approach for mask-based panoptic segmentation in 3D point clouds. Our experiments suggest that our approach (i) performs better than common approaches relying on clustering algorithms that might need hyperparameter tuning for different datasets; and (ii) achieves competitive performance even with a simple feature extractor.

II. Related Work

Semantic Segmentation. Early approaches [23], [35], [36] process the point cloud directly, which is usually time-consuming and computationally expensive for large point clouds. Follow-up works build point kernels [41] while still operating at point level. Several works [34], [29] project the LiDAR point cloud into a range image and use lightweight 2D convolutional networks, leveraging well-known architectures [19]. Other approaches [12], [49] voxelize the 3D space to keep topological relations and use sparse convolutions [17] to reduce the memory usage. There are also works [40], [1], [44], [45] that combine different representations to extract features and get a better performance.

Instance Segmentation. Zhang et al. [47] propose a dense feature encoding technique and Yin et al. [46] represent objects as points for detection and tracking. Hu et al. [21] remove the "stuff" points and use hierarchical clustering to get class-agnostic instances. Based on transformers, DETR [9] performs object detection in images by predicting bounding boxes with learnable queries and solving an end-to-end set prediction problem. They add a mask head like Mask R-CNN [18] to decode masks and obtain instance segmentation.

Panoptic Segmentation [24] requires to fuse semantic and instance segmentation. Several approaches project into 2D images [39], [33], [22], [27] while others [20], [15], [37], [30] keep the 3D structure either using sparse convolutions [17] or kernel point convolutions [41]. Some top-down methods [39], [22] follow image-based works and add a semantic head to Mask R-CNN [18] and perform panoptic segmentation on range images. Bottom-up methods [33], [20], [37], [28], [27] use semantic segmentation predictions to filter the "stuff" points and apply a clustering method as post-processing to get the final instance segmentation, which does not allow training these methods end-to-end. Gasperini et al. [15] propose a learning-based clustering method to train the model end-to-end. Still, they require several post-processing steps to fix the clustering errors and achieve competitive performance. Li et al. [26] propose a cluster-free instance segmentation head by pillarizing foreground point embeddings and finding the connected pillars with pairwise embedding comparison. Other approaches [2], [32] extended the task to the temporal domain by generating instance segmentations that are consistent over time resulting in so-called 4D panoptic segmentation.

Mask Segmentation. Different from other segmentation paradigms, mask-based approaches [18], [7], [9] predict a set of binary masks and corresponding semantic classes. They perform instance segmentation by first estimating bounding boxes and then generating the masks from them. Recently, Cheng et al. [11] proposed to remove the bounding boxes and use a transformer [42] to perform semantic segmentation by using a set of learnable queries as object proposals and directly output the masks. Further works [43], [10] apply this concept to achieve panoptic segmentation in a top-down way without manually-designed post-processing steps like non-maximum suppression to remove duplicated predictions. These approaches have been validated in the 2D image domain but have not been applied to 3D point clouds. We extend these ideas by facing different challenges such as a larger number of points, sparsity of the data, and voxelization artifacts.

In this paper, we propose to perform panoptic segmentation on 3D LiDAR point clouds by directly predicting a set of binary masks and their corresponding semantic classes, removing this way the need for a clustering step.

III. Mask-based Panoptic LiDAR Segmentation

A. Task Definition

Given an input point cloud $\boldsymbol{X} \in \mathbb{R}^{N\times 3}$, we want to segment it into a set $\mathcal{Y} = \{(\boldsymbol{m}_i, c_i)\}_{i=1}^{K}$ of K non-overlapping binary masks $\boldsymbol{m}_i \in \{0,1\}^N$ and their semantic classes $c_i \in \{1,\dots,C\}$ for C classes. Each mask represents either a single instance of a "thing" class or a complete "stuff" class.

Our model predicts a set $\mathcal{Z} = \{(\hat{\boldsymbol{m}}_i, \hat{p}_i)\}_{i=1}^{M}$ of M binary masks predictions $\hat{\boldsymbol{m}}_i \in [0,1]^N$ with values between 0 and 1; and probability distributions $\hat{p}_i$ over the classes $\{1,\dots,C\} \cup \{\emptyset\}$, where $\emptyset$ is an additional "no object" class. Note that we assume $M > K$ and include a "no object" class that is assigned to the extra $M-K$ masks.

We follow prior work [9], [10], [11] and use an architecture consisting of a feature extractor to obtain multi-scale features and a transformer decoder combining learnable queries and backbone features to obtain the segmented output. Fig. 2a shows an overview of our method. We voxelize the input point cloud and extract voxel features at different resolutions via the backbone. We convert voxel features into point features and feed these to the transformer decoder along with M learnable queries (learnable feature vectors). The final M mask predictions are obtained by combining the mask embeddings and the output query embeddings while the mask semantic classes are decoded from the output queries.

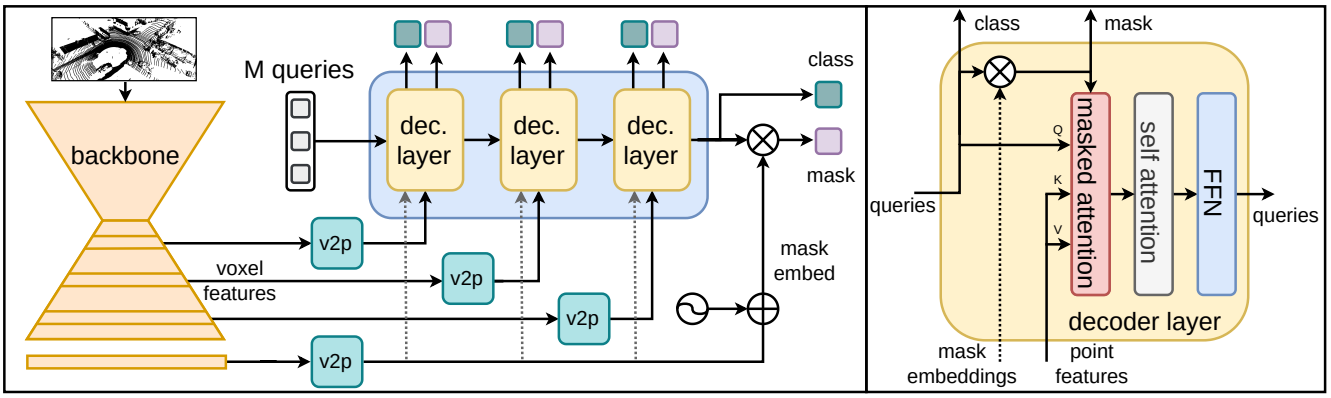
B. Feature Extractor

We use a feature extractor to obtain multi-scale features as input to the decoder. Given the sparse nature of the LiDAR data, we use a MinkowskiNet [12] backbone composed of 3D sparse convolutions to preserve spatial knowledge and obtain meaningful features with a low memory footprint. To show that our approach can be applied to other backbones, we also evaluate it with CylinderNet [49] using cylindrical voxels, and SPVCNN [40] using sparse point-voxel convolutions.

The backbone extracts voxel features $\boldsymbol{F}_v \in \mathbb{R}^{C_v\times N_v}$ at different resolutions, where C_v is the feature dimension and N_v is the number of voxels, which depend on feature stride, and full-resolution point features $\boldsymbol{F}_p \in \mathbb{R}^{C_p\times N}$, where C_p is the feature dimension. We adopt multi-scale features to help segment small instances. To use these features as decoder input, we convert the voxel features into point features with the voxel-to-point function (v2p) $f : \mathbb{R}^{C_v\times N_v} \to \mathbb{R}^{C_v\times N}$ that we discuss in Sec. III-F

C. Transformer Decoder

The decoder takes queries as mask proposals and multi-scale features and outputs the final queries used for the mask predictions. Following the work by Cheng et al. [10],



(a) Description of the approach (b) Description of the decoder layers

Fig. 2: Overview of our method. (a) The 3D feature extractor processes the input point cloud to obtain multi-resolution features. The voxel-to-point (v2p) transforms voxel features into point features. The learnable queries and point features are inputs to the decoder layers. The predicted masks are obtained via dot product between mask embeddings and output queries. The class prediction is decoded from the output query embeddings. (b) we obtain intermediate binary masks and use them in the masked attention, followed by self-attention and a feedforward network (FFN). The intermediate predictions are only used for training and are supervised with the same loss function.

we use a modified transformer decoder layer to leverage optimization and training efficiency improvements by using masked attention. Compared to the standard cross-attention, where the queries attend to all the keys, masked attention restricts the attention to a subset of keys localized inside a mask. Each decoder layer is depicted in Fig. 2b and consists of masked attention between queries and backbone features followed by self-attention and a feedforward network (FFN). Each masked attention combines the M learnable queries with features from different resolutions to enhance the training and help the model segment small objects. We group three decoder layers into a transformer decoder block that we repeat L times. After the decoder blocks, we obtain the output query embeddings $\boldsymbol{Q} \in \mathbb{R}^{M \times C_P}$. We keep spatial reasoning during the decoding step, adding fixed point-wise positional encodings [42] $\boldsymbol{P}_e \in \mathbb{R}^{N \times C_p}$ to the point features input to each cross-attention module and to the full resolution features to obtain mask embeddings $\boldsymbol{E} = \boldsymbol{F}_p + \boldsymbol{P}_e$.

D. Mask Prediction

The final step is to compute the mask predictions using output queries and point features. We obtain the probability distributions $\hat{p}_i$ over the semantic classes by applying a linear layer over the output queries. For the masks predictions, we want to obtain the mask scores $\boldsymbol{M} \in \mathbb{R}^{N \times M}$, i.e., the score of each point for each of the M predicted masks $\hat{\boldsymbol{m}}_i \in [0,1]^N$, where $\hat{\boldsymbol{m}}_i$ is represented by the i^{th} column of $\boldsymbol{M}$. We compute the mask scores via the dot product between mask embeddings and output queries and apply a sigmoid activation function $\boldsymbol{M} = \text{sigmoid}(\boldsymbol{E}\boldsymbol{Q}^\top)$. We can obtain the final prediction by performing argmax twice: over the semantic class for each mask and over the mask scores for each point. In practice, we filter out masks with a large part of their binary mask occluded by other predictions to reduce false positives as done in previous works [9], [11].

Apart from the final predictions, we generate intermediate predictions to enhance the training by supervising them with the same loss function as for the final predictions. Before each decoder layer, we use the mask embeddings $\boldsymbol{E}$ and the queries of the previous step to get the intermediate predictions as shown in Fig. 2b. The masked attention function requires a binary mask to allow the queries to attend only to the point features inside it. We obtain them by applying a threshold $\tau = 0.5$ over the intermediate mask predictions.

E. Loss Function

To train our approach, we follow MaskFormer [11] and perform a bipartite matching between the M predictions $\mathcal{Z}$ and the K ground-truth masks $\mathcal{Y}$ to supervise the training. As mentioned before, the number of predictions M is larger than the number of ground-truth masks K. We complete with "no object" class $\varnothing$ to obtain a one-to-one matching. We define an assignment cost matrix $\mathbf{C}$ for all pairs of prediction-ground-truth taking into account the mask and the semantic class. Each element of the matrix takes the form:

$$\mathbf{C}_{i,j} = -\hat{p}_j(c_i) + \mathcal{L}_{mask}, \tag{1}$$

where $\hat{p}_j(c_i)$ is the probability of class c_i and $\mathcal{L}_{mask}$ is the combination of binary cross entropy and dice loss:

$$\mathcal{L}_{mask} = \lambda_{\text{dice}}\text{Dice}(m_i, \hat{m}_j) + \lambda_{\text{ce}}\text{BCE}(m_i, \hat{m}_j). \tag{2}$$

We obtain the optimal matching given the fixed cost function using the Hungarian method [38]. After the matching between ground-truth masks and predictions, we compute the loss function $\mathcal{L}_m$ over the K matched pairs. It consists of the mask loss $\mathcal{L}_{mask}$ for mask segmentation, and cross-entropy for the semantic class, as follows:

$$\mathcal{L}_m = \sum_{j=1}^{K} -\lambda_{\text{cls}} \log \hat{p}_j(c_i) + \mathcal{L}_{mask}. \tag{3}$$

For the $M - K$ unmatched masks, we want the model to predict the "no object" class. We only use the cross entropy loss that we down-weight by a factor $\alpha < 1$ to compensate

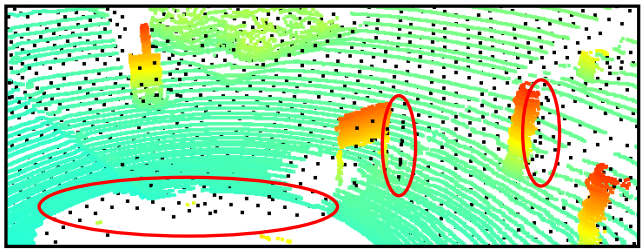


Fig. 3: Downsamplig and voxelization effects. The original point cloud is colored with the height. Black points show the result of voxelization and downsampling. We show artificial points of the downsampled version in regions with no points from the original point cloud.

for class imbalance since there is usually a majority of non-matched masks. The loss takes the form:

$$\mathcal{L}_n = \sum_{j=K+1}^{M} -\alpha \log \hat{p}_j(\varnothing), \tag{4}$$

and the final loss is the sum of both: $\mathcal{L} = \mathcal{L}_m + \mathcal{L}_n$.

Inspired by previous works [10], [25], we randomly sample a fixed number of S points to calculate the mask loss instead of using all the N points in the LiDAR scan to reduce computation. To improve the backbone features, we add an auxiliary loss to directly supervise them. We use a linear layer to convert from full-resolution point features to class probabilities for each point. We supervise the logits with cross-entropy at the point level.

F. Point-Level Multi-Scale Features

While using downsampled feature maps as input to transformer blocks works in the image domain, the voxelization of the real-world point clouds and the downsampling operations can lead to an undesired behavior. The voxels in lower scales cover a bigger volume but are represented just as a point located in the voxel center. Its coordinates represent artificial points which do not exist in the original point cloud. This can be seen in the Fig. 3, where we show the original full point cloud colored with the point height and in black, the same point cloud but voxelized and downsampled by a factor of 4. The signaled points are located in regions in which there are no points in the original point cloud. Using the voxel features $\boldsymbol{F}_v$ in the masked attention harms the performance since we are using non-existing points. To account for this, we project the voxel features to the full point cloud with the voxel-to-point operation (v2p) and feed the resulting point features to the transformer decoder. The v2p function could be to copy the same feature to all points inside the same voxel. To avoid having repeated features for all points inside each voxel, we perform a k-nearest-neighbors interpolation between voxel centers and the points of the full-resolution scan. We discuss the performances of these two v2p functions in Sec. IV-G.

IV. Experimental Evaluation

The main focus of this work is to perform panoptic segmentation in an end-to-end manner by removing the classic clustering step and directly predicting binary masks with their corresponding semantic classes. We present our experiments to show the capabilities of our method and support our key claims, which are: (i) our approach performs better than common approaches relying on clustering algorithms that might need hyperparameter tuning for different datasets, and (ii) it achieves competitive performance even if using a simple feature extractor without any modification.

A. Implementation Details

We use $L = 3$ decoder blocks (9 decoder layers). In the loss function, we adopt $\lambda_{\text{dice}} = \lambda_{\text{ce}} = 5, \lambda_{\text{cls}} = 2$ and the number of randomly sampled points $S = 50000$. We set the number of learnable queries $M = 100$. We train the models for 100 epochs using AdamW [31] optimizer and step learning rate schedule. The initial learning rate is 0.0001 and we decay the learning rate by a factor of 10 at epoch 80. As augmentations, we apply random rotations, flips along the X or Y axis, random scaling, and random transformations.

B. Experimental Setup

To evaluate our method, we use SemanticKITTI [4], [3], which provides point-wise semantic and instance annotations [5] for 22 sequences of the KITTI odometry dataset [16]. Sequences 00 to 10 are used for training except for sequence 08, which serves as the validation set and sequences 11 to 21 for the test set. We run further experiments on nuScenes [6], which consists of 1000 sequences with 20 seconds of duration recorded with a 32-beam LiDAR sensor. The annotations are created every 0.5 s and include 16 semantic classes including 10 "thing" classes.

We use panoptic quality (PQ), segmentation quality (SQ) and recognition quality (RQ) as metrics to evaluate panoptic segmentation following the task definition [24].

C. Comparison with Clustering as Post-Processing

This experiments supports claim (i) and shows that our proposed method performs better than approaches relying on clustering algorithms. For that, we compare the performance of three backbones with different post-processing algorithms with our proposed end-to-end model. We build our models on top of three widely used 3D semantic segmentation networks as feature extractors without making any change, namely MinkNet [12], CylinderNet [49], and SPVCNN [40]. We will see that our method works with potentially any given backbone and using a better semantic segmentation network would help to achieve better panoptic segmentation results. We name our models MaskPLS-M (MinkNet), MaskPLS-C (CylinderNet), and MaskPLS-S (SPVCNN).

For the clustering approaches, we follow common practices and add an instance head composed of three convolutions and two linear layers to predict, for each point of a "thing" class, an offset vector to the instance center. During training, we filter out the "stuff" points using the labels and semantic predictions during inference. As post-processing, we shift the points using the predicted offsets and use a clustering algorithm to group points into instances. We use HDBSCAN [8], mean shift [13] and the dynamic shifting module [20]. The results on the

Method	PQ	SQ	RQ	mIoU
MinkNet + HDBSCAN	57.9	79.8	67.6	63.4
MinkNet + MeanShift	58.3	79.8	68.0	63.4
MinkNet + DS	58.2	**80.4**	67.9	**63.5**
MaskPLS-M (ours)	**59.8**	76.3	**69.0**	61.9
CylinderNet + HDBSCAN	55.3	77.0	65.5	63.2
CylinderNet + MeanShift	56.4	76.5	67.1	**63.5**
CylinderNet + DS	57.7	**77.6**	68.0	**63.5**
MaskPLS-C (ours)	**58.9**	75.7	**68.4**	63.4
SPVCNN + HDBSCAN	52.5	**77.5**	62.9	60.0
SPVCNN + MeanShift	53.0	**77.5**	65.0	60.0
SPVCNN + DS	52.8	77.3	63.3	59.9
MaskPLS-S (ours)	**55.5**	75.0	**65.1**	**60.6**

TABLE I: Panoptic segmentation results on SemanticKITTI validation set.

validation set of SemanticKITTI are presented in Tab. I and show that our method consistently improves RQ and PQ for all the different backbones and outperforms all clustering methods in terms of PQ while being end-to-end trainable. MaskPLS-M achieves a better PQ but a lower mIoU with respect to the baselines. The cause might be that, for the baselines, the backbone is trained only for semantic segmentation while MaskPLS is trained for a different objective, which comprises both semantic and instance segmentation. Therefore, the balance between both tasks might harm the performance of a single one. In our experiments, we saw that even though the mIoU decreases, the SQ increases for most of the "thing" classes, making it easier to separate points into instances.

D. Performance

The next experiment shows that our approach achieves competitive performance by just using a simple feature extractor without any modification and thus supports claim (ii). We compare the performance of our approach MaskPLS-M and previous works on the SemanticKITTI test set and show the results in Tab. II. We achieve competitive performance using as feature extractor, a simple ResNet-like model without any modification. Furthermore, we surpass previous end-to-end approaches Panoster [15] and CPSeg [26]. We rank first on the leaderboard among open-source projects. It is important to note that the semantic segmentation performance plays an important role in panoptic segmentation and in the panoptic quality. The state-of-the-art approaches achieve better panoptic quality while relying on a backbone with a much better semantic segmentation performance. Our approach achieves a comparably good SQ, particularly for "things". This shows that it can better differentiate the detected segments and this results in more accurate instance predictions. In our approach, the semantic segmentation performance heavily relies on the backbone performance. Potentially, a better backbone would improve our panoptic segmentation results.

E. Different Dataset

With this experiment, we show that we do not need to tune any hyperparameter to use our model in a different dataset and provide support for claim (i). We conduct a similar experiment to the one in Sec. IV-C but on nuScenes.

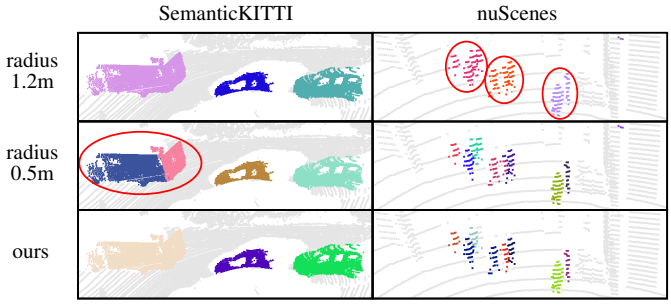


Fig. 4: Comparison of instance segmentation using mean shift with different clustering radius and our approach. In the top row, we select a radius for SemanticKITTI and when evaluating on nuScenes, several pedestrians are grouped as a single instance. In the middle row, we select a radius for nuScenes. The pedestrians are correctly separated but a vehicle is split into two instances. In the bottom row, our approach correctly segments instances in both datasets without changing any hyperparameter.

Tab. III shows a comparison between the performances of our approach and different clustering methods as post-processing step with MinkNet and CylinderNet as backbones on the nuScenes validation set. For a fair comparison, we retrain DS-Net [20] with the annotations provided by Fong et al. [14]. Our approach outperforms the baselines in terms of PQ also in this dataset recorded with a different LiDAR sensor. Furthermore, our approach adapts to this different dataset without tuning any hyperparameter while in the case of the clustering algorithms it is usually necessary to specify at least a bandwidth or the minimum cluster size. The performance of our approach does not match state-of-the-art methods because we do not change any hyperparameter to re-train it on this different dataset. To improve the performance, some parameters should be changed to account, for example, for the different LiDAR density in this new dataset.

In Fig. 4, we depict examples of the instance segmentation performances with different clustering radius applied to the semantic predictions obtained by MaskPLS. We show that the clustering algorithms usually need a different radius for different classes and different datasets. We use MinkNet and mean shift and compare it with our approach. First, we select a radius of 1.2 m for SemanticKITTI and the instances are correctly segmented. However, on nuScenes several pedestrians are clustered together. Second, we select a radius of 0.5 m for nuScenes, where the pedestrians are separated but on SemanticKITTI a vehicle is split into two instances. Our method, in contrast, correctly segments instances of different classes in both datasets despite the different sensors and without hyperparameter tuning.

F. Mask Embeddings

We generate the mask predictions via dot product between point features and query embeddings. Thus, the quality of the masks relies on the quality of the point features, which should contain both semantic and spatial information. This can be seen in Fig. 5.a, where we show the cosine similarity between the mean feature of a car that we want to segment and the rest of the points. All the car points have a higher similarity than the background points, showing the semantic

Method	PQ	PQ+	RQ	SQ	PQ^{Th}	RQ^{Th}	SQ^{Th}	PQ^{St}	RQ^{St}	SQ^{St}	mIoU
Panoster [15]	52.7	59.9	64.1	80.7	49.4	58.5	83.3	55.1	68.2	78.8	59.9
CPSeg [26]	57.0	63.5	68.8	82.2	55.1	64.1	86.1	58.4	72.3	79.3	62.7
EfficientLPS [39]	57.4	63.2	68.7	83.0	53.1	60.5	87.8	60.5	**74.6**	79.5	61.4
PVCL [30]	59.1	65.7	69.6	84.0	59.8	66.7	89.2	58.6	71.6	80.3	64.0
GP-S3Net [37]	60.0	**69.0**	**72.1**	82.0	**65.0**	**74.5**	86.6	56.4	70.4	78.7	**70.8**
SCAN [45]	**61.5**	67.5	**72.1**	84.5	61.4	69.3	88.1	**61.5**	74.1	**81.8**	67.7
Panoptic-PHNet [28]	**61.5**	67.9	**72.1**	**84.8**	63.8	70.4	**90.7**	59.9	73.3	80.5	66.0
MaskPLS-M (ours)	58.2	63.3	68.6	83.9	55.7	61.7	89.2	60.0	73.7	80.0	62.5

TABLE II: Panoptic segmentation results on SemanticKITTI test set.

Method	PQ	PQ+	RQ	SQ	PQ^{Th}	RQ^{Th}	SQ^{Th}	PQ^{St}	RQ^{St}	SQ^{St}	mIoU
EfficientLPS [39]	62.0	65.6	73.9	83.4	56.8	68.0	83.2	70.6	83.6	83.8	65.6
Panoptic-PolarNet [48]	63.4	67.2	75.3	83.9	59.2	70.3	84.1	70.4	83.5	83.6	66.9
Panoptic-PHNet [28]	**74.7**	**77.7**	**84.2**	**88.2**	**74.0**	**82.5**	**89.0**	**75.9**	**86.9**	**86.8**	**79.7**
MinkNet + HDBSCAN	52.6	57.4	63.2	69.0	59.1	71.2	81.2	47.9	57.4	60.2	61.6
MinkNet + MeanShift	54.4	58.9	64.8	69.5	62.6	74.0	82.6	48.3	58.1	60.0	61.6
MaskPLS-M (ours)	57.7	60.2	66.0	71.8	64.4	73.3	84.8	52.2	60.7	62.4	62.5
CylinderNet + DS	51.4	56.1	62.4	68.3	58.8	70.9	81.0	46.0	56.2	59.0	56.7
MaskPLS-C (ours)	57.4	59.8	66.2	71.8	63.6	72.7	84.6	52.9	61.4	62.5	60.2

TABLE III: Panoptic segmentation results on nuScenes validation set.

information in the point-wise features. Furthermore, the closer the points are to the desired car, the higher the similarity, depicting the spatial information. However, the obtained masks are not always accurate enough to segment a single instance. In our example in Fig. 5.b, the predicted mask also includes a part of a second car.

We argue that the problem is caused by using the backbone features $\boldsymbol{F}_p$ to generate the masks, i.e., as mask embeddings $\boldsymbol{E}$. In this configuration, the point features have to account for spatial and semantic information at the same time. We propose to divide the mask embeddings into spatial and semantic embeddings to relax the dependency of the point-wise features with respect to the spatial information. We use point-wise fixed positional encodings $\boldsymbol{P}_e$ [42] to capture the spatial information. For the semantic information, we use the output features from the backbone $\boldsymbol{F}_p$. This way, the backbone features do not have to account for spatial information and only focus on accurate semantic information to better describe the scene. The final mask embeddings are the sum of both: $\boldsymbol{E} = \boldsymbol{F}_p + \boldsymbol{P}_e$. In Fig. 5.c, we show the feature similarity between the mean feature of the desired car and all the other points after making this change. The similarity is now higher for almost all car points, showing that the features are mainly focusing on semantic information. We add the positional encodings to obtain our proposed mask embedding and the generated masks can more accurately capture single instances, as depicted in Fig. 5.d.

G. Ablation Studies

We perform experiments to show the influence of each design choice and the number of decoder blocks. To reduce the total wall-clock training time, we perform the experiments in a smaller model by reducing the feature dimensions of the backbone blocks and the decoder blocks by a factor of 4. We use the same training schedule as in Sec. IV-D except that we train the models with a fixed learning rate for 50 epochs. We show the results on the SemanticKITTI validation set.

#	sem loss	point feat.	interp.	pos enc	PQ	runtime [ms]
A					51.9	390
B	✓				52.3	390
C	✓	✓			52.7	283
D	✓	✓	✓		53.4	313
E	✓	✓	✓	✓	54.9	315

TABLE IV: Influence of the design choices in panoptic quality and inference time in the validation set of SemanticKITTI.

Design choices: In Tab. IV, we show the influence of each design choice in the final performance and runtime. In experiment (A) we follow Mask2Former [10] but using a feature extractor with 3D sparse convolutions and voxel features, achieving a panoptic quality of 51.9%. In (B) we add the auxiliary semantic segmentation loss on the full resolution point features. We seek to get more distinctive features, they are used to generate the mask predictions and the overall performance highly relies on their quality. We boost the performance by 0.4 percent points without affecting runtime. In (C), we use point features instead of voxel features by copying voxel features to all the points within that voxel as v2p function and obtain a gain of 0.4 percent points and the runtime decreases to 283 ms. The reason is that we do not need to interpolate the masks for the masked-attention since we are using full resolution masks and features. In (D) we perform KNN interpolation between voxel features and point features. Each point feature is the weighted sum of the k nearest voxels. We use $k = 3$ neighbors and improve 0.7 percent points while increasing runtime to 313 ms. Finally, in experiment (E) we add the positional encoding to the full-scale point features to

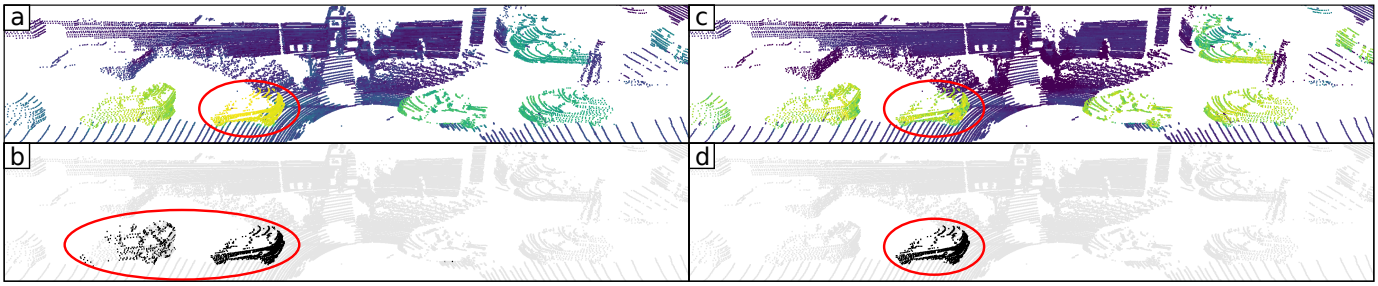


Fig. 5: Influence of the proposed mask embeddings: (a) is the cosine similarity between the mean feature of the circled car and all the other point features. The point features provide semantic and spatial information: all car points have high similarity and nearby points have more similar features. The generated mask (b) includes part of a second car. Splitting the mask embeddings, the features from the backbone focus mainly on semantic information, giving all cars similar features (c) while the spatial information is given by the positional encodings. As a result, the mask can accurately segment a single car (d).

obtain the so-called mask embeddings and generate the mask predictions with them. This boosts the performance by 1.5 percent points to the final PQ of 54.9% and increases the inference time to 315 ms.

Number of decoder blocks: We show the influence of the number of decoder blocks in Tab. V. Using one block, we achieve a panoptic quality of 53%. With two blocks, the performance increases by 0.8 percent points and the addition of a third block boosts the performance by 1.1 percent points, reaching a PQ of 54.9%. The mIoU remains approximately the same but extra blocks improve instance segmentation.

# decoder blocks	PQ	RQ	SQ	mIoU
1	53.0	63.2	73.7	57.7
2	53.8	64.2	78.7	58.5
3	54.9	64.9	79.7	58.1

TABLE V: Influence of the number of decoder blocks in panoptic quality on the validation set of SemanticKITTI.

H. Learnable Queries

In this experiment, we show that the learnable queries act as mask proposals. We visualize the proposals by generating the masks before the first decoder layer. In Fig. 6 we compute the mask scores for each point via the dot product and visualize the proposals. For "stuff" classes, the learnable query covers all the points belonging to that class without differentiating the point coordinates. In other words, the queries that generate "stuff" masks are position invariant. On the other hand, for "thing" classes, the queries must separate between instances and account for the spatial information and the mask proposals focus on individual instances.

V. Conclusion

In this paper, we presented an approach to perform panoptic segmentation of 3D LiDAR point clouds by directly predicting a set of binary masks along with their semantic classes. Our method allows for optimizing the model end-to-end and does not need any clustering algorithm as post-processing. This enables us to successfully use our method on different datasets without changing any hyperparameter. We implemented and evaluated our approach on different datasets and compared it against baselines and techniques to achieve panoptic segmentation. We performed experiments to support all our claims and showed that our approach achieves competitive performance even with a semantic segmentation network without any modification as feature extractor.

References

[1] I. Alonso, L. Riazuelo, L. Montesano, and A. Murillo. 3D-MiniNet Learning a 2D Representation from Point Clouds for Fast and Efficient 3D LIDAR Semantic Segmentation. *IEEE Robotics and Automation Letters (RA-L)*, 5(4):5432–5439, 2020.

[2] M. Aygun, A. Osep, M. Weber, M. Maximov, C. Stachniss, J. Behley, and L. Leal-Taixe. 4D Panoptic LiDAR Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[3] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, J. Gall, and C. Stachniss. Towards 3d lidar-based semantic scene understanding of 3d point cloud sequences: The semantickitti dataset. *Intl. Journal of Robotics Research (IJRR)*, 40(8-9):959–967, 2021.

[4] 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 *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[5] J. Behley, A. Milioto, and C. Stachniss. A Benchmark for LiDAR-Based Panoptic Segmentation Based on KITTI. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[6] H. Caesar, V. Bankiti, A.H. Lang, S. Vora, V.E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[7] Z. Cai and N. Vasconcelos. Cascade R-CNN: Delving Into High Quality Object Detection. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[8] R.J. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *Proc. of Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, 2013.

[9] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 213–229, 2020.

[10] B. Cheng, I. Misra, A.G. Schwing, A. Kirillov, and R. Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[11] B. Cheng, A.G. Schwing, and A. Kirillov. Per-pixel classification is not all you need for semantic segmentation. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2021.

[12] C. Choy, J. Gwak, and S. Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[13] D. Comaniciu and P. Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 24:603–619, 2002.

[14] W. Fong, R. Mohan, H. Valeria, L. ZHOU, H. Caesar, O. Beijbom, and A. Valada. Panoptic nuScenes A Large-Scale Benchmark for LiDAR Panoptic Segmentation and Tracking. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[15] S. Gasperini, M.N. mahani, A. Marcos-Ramiro, N. Navab, and F. Tombari. Panoster: End-To-End Panoptic Segmentation of LiDAR Point Clouds. *IEEE Robotics and Automation Letters (RA-L)*, 6(2):3216–3223, 2021.

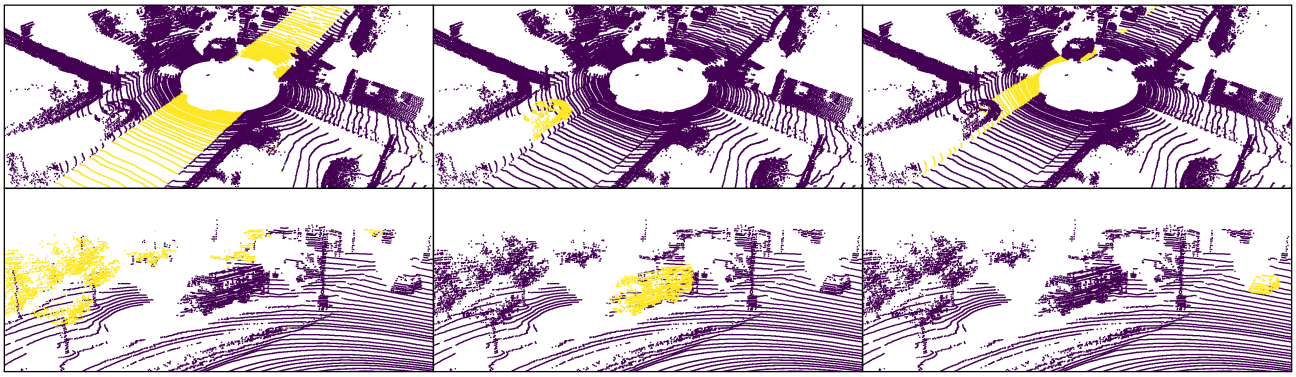


Fig. 6: Mask proposals from some learnable queries showing "thing" instances and "stuff" classes. We obtain the mask scores via dot product between point features and queries before the transformer decoder.

[16] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[17] B. Graham, M. Engelcke, and L. van der Maaten. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[18] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[20] F. Hong, H. Zhou, X. Zhu, H. Li, and Z. Liu. LiDAR-Based Panoptic Segmentation via Dynamic Shifting Network. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[21] P. Hu, D. Held, and D. Ramanan. Learning to Optimally Segment Point Clouds. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):875–882, 2020.

[22] J.V. Hurtado, R. Mohan, W. Burgard, and A. Valada. Mopt: Multi-object panoptic tracking. *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[23] J. Jung, C. Stachniss, and C. Kim. Automatic room segmentation of 3d laser data using morphological processing. *Intl. Journal of Geo-Information*, 6(7):206, 2017.

[24] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[25] A. Kirillov, Y. Wu, K. He, and R. Girshick. PointRend: Image Segmentation As Rendering. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[26] E. Li, R. Razani, Y. Xu, and B. Liu. Cpseg: Cluster-free panoptic segmentation of 3d lidar point clouds. *arXiv preprint*, 2021.

[27] E. Li, R. Razani, Y. Xu, and B. Liu. Smac-seg: Lidar panoptic segmentation via sparse multi-directional attention clustering. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 9207–9213, 2022.

[28] 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 *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 11809–11818, 2022.

[29] 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. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):738–745, 2022.

[30] M. Liu, Z. Qiang, H. Zhao, J. Li, Y. Du, K. Keutzer, L. DU, and S. Zhang. Prototype-Voxel Contrastive Learning for LiDAR Point Cloud Panoptic Segmentation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[31] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2019.

[32] R. Marcuzzi, L. Nunes, L. Wiesmann, I. Vizzo, J. Behley, and C. Stachniss. Contrastive Instance Association for 4D Panoptic Segmentation for Sequences of 3D LiDAR Scans. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[33] A. Milioto, J. Behley, C. McCool, and C. Stachniss. LiDAR Panoptic Segmentation for Autonomous Driving. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[34] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[35] C.R. Qi, H. Su, K. Mo, and L.J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[36] C. Qi, K. Yi, H. Su, and L.J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2017.

[37] R. Razani, R. Cheng, E. Li, E. Taghavi, Y. Ren, and L. Bingbing. GP-S3Net: Graph-Based Panoptic Sparse Semantic Segmentation Network. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

[38] Reid, Donald B. An algorithm for tracking multiple targets. *IEEE Trans. on Automatic Control*, 24(6):843–854, 1979.

[39] K. Sirohi, R. Mohan, D. Büscher, W. Burgard, and A. Valada. EfficientLPS Efficient LiDAR Panoptic Segmentation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[40] 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 *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[41] H. Thomas, C. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L.u. Kaiser, and I. Polosukhin. Attention is all you need. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2017.

[43] H. Wang, Y. Zhu, H. Adam, A. Yuille, and L.C. Chen. MaX-DeepLab: End-to-end panoptic segmentation with mask transformers. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[44] J. Xu, R. Zhang, J. Dou, Y. Zhu, J. Sun, and S. Pu. RPVNet: A Deep and Efficient Range-Point-Voxel Fusion Network for LiDAR Point Cloud Segmentation. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

[45] S. Xu, R. Wan, M. Ye, X. Zou, and T. Cao. Sparse cross-scale attention network for efficient lidar panoptic segmentation. In *Proc. of the Conf. on Advancements of Artificial Intelligence (AAAI)*, pages 2920–2928, 2022.

[46] T. Yin, X. Zhou, and P. Krahenbuhl. Center-Based 3D Object Detection and Tracking. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[47] F. Zhang, C. Guan, J. Fang, S. Bai, R. Yang, P. Torr, and V. Prisacariu. Instance Segmentation of LiDAR Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[48] Z. Zhou, Y. Zhang, and H. Foroosh. Panoptic-PolarNet: Proposal-Free LiDAR Point Cloud Panoptic Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[49] 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 *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.