2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

LiDAR Panoptic Segmentation for Autonomous Driving

Andres Milioto Jens Behley Chris McCool Cyrill Stachniss

Abstract— **Truly autonomous driving without the need for human intervention can only be attained when self-driving cars fully understand their surroundings. Most of these vehicles rely on a suite of active and passive sensors. LiDAR sensors are a cornerstone in most of these hardware stacks, and leveraging them as a complement to other passive sensors such as RGB cameras is an enticing goal. Understanding the semantic class of each point in a LiDAR sweep is important, as well as knowing to which instance of that class it belongs to. To this end, we present a novel, single-stage, and real-time capable panoptic segmentation approach using a shared encoder with a semantic and instance decoder. We leverage the geometric information of the LiDAR scan to perform a novel, distance-aware tri-linear upsampling, which allows our approach to use larger output strides than using transpose convolutions leading to substantial savings in computation time. Our experimental evaluation and ablation studies for each module show that combining our geometric and semantic embeddings with our learned, variable instance thresholds, a category-specific loss, and the novel trilinear upsampling module leads to higher panoptic quality. We will release the code of our approach in our LiDAR processing library LiDAR-Bonnetal [27].**

I. Introduction

Perception and scene understanding are key components for building fully autonomous cars that can drive safely even in unknown parts of the world. A multitude of sensors such as cameras, LiDARs, and radars offering redundant views of the world are part of the hardware stack of these vehicles. Image-based perception has been steadily becoming more capable due to advances in deep learning [21] and convolutional neural networks (CNN). LiDAR sensors are often used because they produce accurate distance measurements, even in scenarios where other sensors fail, like at night. However, challenges caused by the characteristics of the LiDAR data, such as its distance-dependent sparsity, consequently call for different solutions.

Panoptic segmentation [16] is a recent task unifying semantic segmentation of so-called *stuff* classes and instance-specific *thing* classes jointly. Specifically, *stuff* refers to uncountable classes, such as *vegetation*, or *road*, but also countable classes that are not critical to distinguish individually when performing a specific task, such as is the case for *buildings* while driving. Opposite to this, *things* represent interesting countable classes for the task the robot performs, such as driving participants (i.e. *cars*, *pedestrians*, etc). Therefore, this task provides a unified understanding of the scene components, leading towards scene understanding capturing the complete picture.

All authors are with the University of Bonn, Germany. This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germanys Excellence Strategy EXC 2070 390732324, as well as grant number BE 5996/1-1.

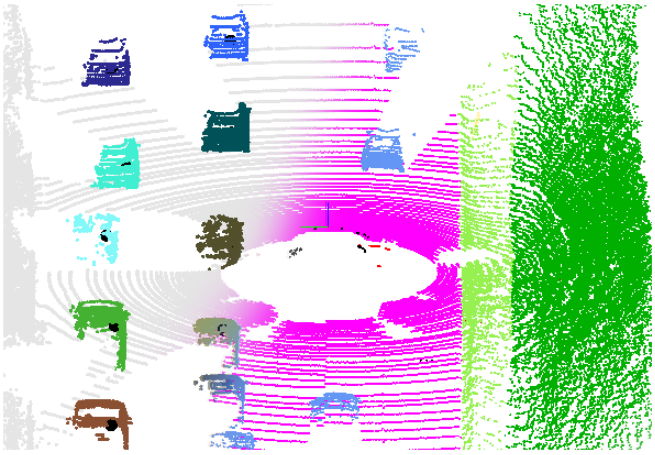

Fig. 1: Our approach provides a panoptic segmentation for point clouds from a rotating automotive LiDAR. Thus, we assign each point a semantic label (right part), and instance IDs (left part).

Recently, many approaches to this task using images were proposed [16], [22], [24], [31], [47]. In this paper, we investigate the task of panoptic segmentation using solely LiDAR scans and present an approach for solving this task, as shown in Fig. 1. We propose a single-stage approach that jointly solves semantic and instance segmentation. This architecture learns both instance and semantic embeddings using a shared encoder. The instance decoder provides an offset prediction for each point that points towards the object center, which allows us to segment individual instances. In turn, the semantic embeddings are used to extract point-wise semantic labels and aid the clustering of object centers. Furthermore, we propose a novel upsampling method that allows us to use large output strides, enabling better runtime performance. Combined with a category-based loss, we achieve high panoptic quality for the panoptic task [2] of SemanticKITTI [1] than an approach combining state-of-the-art projective semantic segmentation [27] and state-of-the-art object detection [20]. Lastly, our experiments show that our single-stage approach runs at a fraction of the time compared to two two-stage approaches, which is paramount for moving vehicles. In summary, our contributions are: (i) a single-stage approach for LiDAR panoptic segmentation that achieves state-of-the-art performance at a fraction of the processing time of two-stage approaches, (ii) a novel upsampling strategy exploiting the distance information provided by the LiDAR point clouds leading to better panoptic quality and increased runtime efficiency, and (iii) the novel combination of semantic and geometric embeddings with learned point-wise radii for metric learning-based instance clustering.

978-1-7281-6211-9/20/$31.00 ©2020 IEEE

II. Related Work

We aim to provide a broad overview of closely related instance and semantic segmentation approaches for point clouds, but we also discuss closely related RGBD and image-based approaches.

Semantic Segmentation. For semantic segmentation of point clouds, a variety of approaches have been proposed. Voxel-based methods transform the point cloud into a voxel-grid and apply convolutional neural networks with 3D convolutions for object classification [25] and semantic segmentation [14]. Both approaches were among the first investigating such models and allow for directly exploiting architectures and insights known from image-based methods.

To overcome the limitations of the voxel-based representation, such as the inherent higher memory consumption with increasing voxel grid resolution, approaches either upsample voxel-predictions [39] using a conditional random field (CRF) or use a different representation, such as more efficient spatial subdivisions [9], [17], [37], [44], [50], graphs [19], [40], splats [38], or points directly [7], [10], [13], [15], [32], [34], [36], [41]. Opposite to these spatial partition approaches, methods exploiting the organization of the generated measurements by a rotating automotive LiDAR sensor [45], [46] or approaches using a cylindrical or spherical projection [27] of the point cloud showed promising results on the KITTI Vision Benchmark and its extension SemanticKITTI [1]. Compared to the aforementioned point cloud-based approaches, these techniques can use larger backbones [35] and realize a more efficient neighbor search by exploiting the organization of the data from the sensor directly [27].

We base our single-stage approach on the latter style, and we present a novel range-image-based tri-linear upsampling method in our decoders that exploits the image representation of neighbor information but uses the actual distances between points from the point cloud pyramid to upsample features spatially. Furthermore, we exploit a category loss that exploits the knowledge of a useful dataset ontology to improve the accuracy of the results.

Instance Segmentation. For image-based instance segmentation, there are mainly two types of approaches: detection-based [11], [12], [48] and clustering-based [5], [26], [29], [3]. Detection-based approaches, pioneered by Mask R-CNN [11], first locate objects using an object detector and then segment the object inside the bounding box. Clustering-based approaches, pioneered by Brabandere *et al*. [3], use metric learning to find an embedding, which facilitates the clustering of pixels from an instance. Often, this also involves the prediction of a seed pixel or point, like the center of an object [5], [26], [29], from which the clustering is seeded.

Also, point cloud instance segmentation has been explored. The approach of Wang *et al*. [42] extracts point-wise features using a PointNet, which are used to generate a similarity matrix, a confidence map, and a semantic segmentation then used to cluster instances by virtue of the similarity scores. The two-stage approach of Hou *et al*. [12] regresses bounding boxes for objects and uses then information from point clouds, but also image information to generate an instance mask for each bounding box. In contrast, the single-stage approach of Yang *et al*. [48] directly estimates a fixed number of bounding boxes and associates each bounding box with a point-wise mask separating the object from the background. Yi *et al*. [49] use object-like proposals instead of bounding boxes, which are then used to generate bounding boxes, segmentation masks, and classification into object classes.

Panoptic Segmentation. Recently, the task of panoptic segmentation [16], i.e., jointly predicting a semantic segmentation of *stuff* classes and instance segmentations for *things* gained significant interest using images [5] or RGBD data [12], [30], [43]. Panoptic segmentation metrics were also adopted by several of the major image datasets [6], [23], [28].

The approach of Pham *et al*. [30] uses a PointNet-based network to provide semantic class probabilities, but also instance embeddings. These are then used by a conditional random field [18] to predict instance labels and semantic labels for an RGBD scan. Similarly, Wang *et al*. [43] use an encoder with two decoders to generate semantic and instance features using PointNets for RGBD data, combining these with an associative segmentation module that uses semantics to generate instance IDs and vice versa.

In contrast to prior work, we propose a single-stage, end-to-end trainable, and real-time capable approach using point clouds generated by a rotating automotive LiDAR. Our approach combines a suite of practices that improve panoptic quality. This includes the combination of semantics and geometry for instance clustering, a learned point-wise threshold for said clustering, a new trilinear upsampling for range images in the decoders, and a joint class plus category loss.

III. Our Approach

Fig. 2 illustrates our network architecture. First, the point cloud obtained by the LiDAR scanner is projected to a range-image-like representation containing the range, (x, y, z) point coordinates, and remission of each point by virtue of a spherical projection of the de-skewed scans caused by the ego-motion of the vehicle. Then, we extract features at different resolutions, or output strides (OS), using a shared backbone, which is trained with all the losses through back-propagation. At the same time, we construct a point cloud pyramid which samples points from the image representation of the latter at exactly the location where the downsampling, stride 2 convolutions are applied in the backbone. This helps us recover the features with higher accuracy during the upsampling process.

Following the backbone and image pyramid, we use two separate decoders, bringing back the backbone features to the original image resolution, which also contain convolutional layers that learn task-specific functions. The first decoder extracts a semantic embedding $\hat{\mathbf{e}}_p$ that allows us to predict classes and categories, as well as an error estimate for the embeddings, used in the clustering. The second decoder

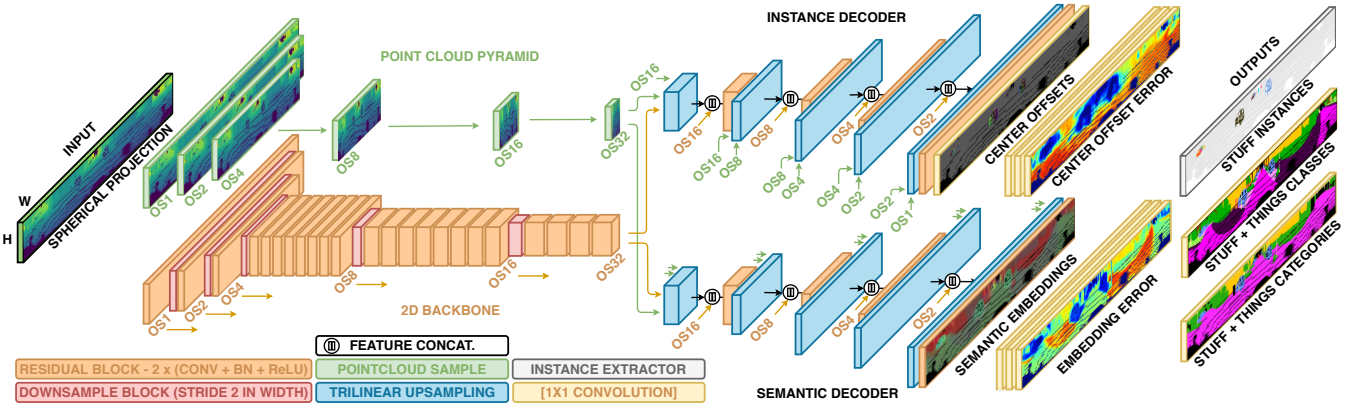

Fig. 2: Architecture layout for the single-stage, projective panoptic segmentation approach. For detailed samples of inputs, intermediate representations, and outputs see Fig. 3

extracts, for each point, an offset to the center of the instance. It also predicts an error estimate for the offsets, used later in the clustering (Sec. III-D). Both decoders use a novel range-image-based trilinear upsampling, which we explain in detail in Sec. III-C. Finally, the instance extractor uses the center offsets and the semantic embeddings to assign an instance id to each point in the *thing* classes and categories, before unprojecting the points to 3D. Fig. 3 shows an example of the input range image, the semantic embedding $\hat{\mathbf{e}}_p$ as a random projection from 32 dimensions to 3 displayed as RGB, the center offsets $\hat{\mathbf{o}}_p$ showing each one of the offsets in x, y, and z as RGB colors, and the final outputs based on semantic and geometric embeddings.

A. Projection

The first step in the projective panoptic segmentation pipeline is to project the points using a spherical projection (Fig. 3, top). To this end, we transform all (x, y, z) 3D points into a set of (u, v) 2D image coordinates using the formula:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y, x)\,\pi^{-1}\right] & W \\ \left[1 - \left(\arcsin(z\,r^{-1}) + \mathrm{f}_{\mathrm{up}}\right)\mathrm{f}^{-1}\right] & H \end{pmatrix} \quad (1)$$

where $r = \sqrt{x^2 + y^2 + z^2}$ is the distance from the point to the sensor, and $f = f_{\mathrm{up}} + f_{\mathrm{down}}$ is the vertical sensor field of view. This generates a $(5, H, W)$ volume which represents the point cloud as an image with channels $(\mathrm{range}, x, y, z, \mathrm{remission})$. For a point cloud of size N, we generate an index matrix of size $(N, 2)$ containing all the (u, v) image coordinate pairs, which we use later when transferring back the predictions to 3D (see Sec. III-E).

B. Backbone and Point Cloud Pyramid

Relevant works for projection-based LiDAR semantic segmentation are SqueezeSeg [45], [46] and RangeNet++ [27], which is the first approach of this type for SemanticKITTI. These approaches exploit the way the sensor acquires the points using a rotating array of laser beams and use a 2D segmentation CNN on a spherical projection of the input point cloud.

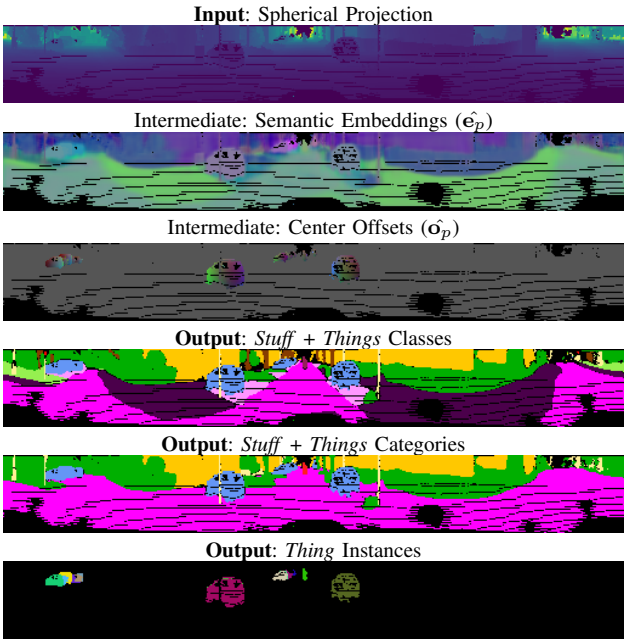

Fig. 3: Example input, intermediate results, and outputs of our panoptic segmentation.

Both of these approaches downsample periodically on the width dimension by using strided convolutions. This generates feature volumes of OS 1, 2, 4, 8, 16, and 32, which are later skipped to the decoder to recover high-frequency signals lost in downsampling, aiding the upsampling process, which uses either bilinear upsampling or transposed convolutions. However, neither of these approaches exploit the fact that the inputs contain useful metric information as they downsample, to aid the upsampling. In our work, each time we apply strided convolutions, we store a downsampled version of the point cloud which contains the points at the centers of the locations where the kernels were applied. This can be used to recover useful geometric information during the upsampling in the decoder that allows us to improve the accuracy of the final output. The foundation of our architecture and shared feature extractor for both the instance head and the semantic head branch is DarkNet53 [27], [35].

C. Decoders

After the point cloud range image is processed by the backbone and the sampled pyramid is generated, we pass the features through two different decoders that complement each other to solve the task of panoptic segmentation. One decoder predicts the instance centers, and the other one the semantics of the scene. Both decoders follow the same structure, which is illustrated in Fig. 2.

In contrast to prior works [27], [45], [46], which upsample the backbone features using transposed convolutions or bilinear upsampling, exploiting closeness in the image, we implement a differentiable trilinear upsampling layer, which exploits the fact that our inputs are 3D point clouds and not camera images. As in bilinear upsampling, for each point in the resolution that is currently upsampled, we find the 4 corresponding points in the coarser grid using the point cloud pyramid. However, since the real information of vicinity between the points and its coarser corresponding points is known to us through their real (x, y, z) coordinates, besides their image (u, v) coordinate. This allows us to approximate a trilinear upsampling by using the real 3D Euclidean distances rather than the 2D image ones. Even though this is technically not a strict trilinear upsampling due to the absence of a cubic lattice, our approach uses the real 3D distances. Thus, it more closely resembles the trilinear upsampling than bilinear interpolation, as shown in Fig. 4. After upsampling the feature volume, we concatenate the upsampled features with the matching resolution from the backbone as a skip connection followed by a convolutional block. These blocks also learn the task-dependent weights that separate the tasks. This is done recursively until the input resolution is met, and the task-dependent heads are applied. In the remainder, we describe each decoder head individually.

1) Instance decoder: This decoder has the responsibility to predict a 3D offset $\hat{\mathbf{o}}_p = [\Delta x, \Delta y, \Delta z]^T$ for each point $\mathbf{x}_p = [x_p, y_p, z_p]^T$ on the range image belonging to one of the *things*, from its center $\mathbf{c} = [x_c, y_c, z_c]^T$. This is similar to [29], which predicts a 2D offset to the center for instance segmentation, and [33], which predicts 3D offsets for point cloud object detection. Since we are using range image representations of 3D point clouds, our approach sits in the middle, predicting 3D offsets given an image representation. After the offsets are predicted, each point in the *thing* mask (from the semantic segmentation) predicts a center coordinate, which is used to cluster the instances. This is done using the region-based clustering method described in Sec. III-D.

The offsets $\hat{\mathbf{o}}_p$ are learnt through an L_2 loss of the form:

$$\mathcal{L}_{CENTER} = \frac{1}{I}\sum_{i=1}^{I}\frac{1}{P_i}\sum_{p=1}^{P_i}[\hat{\mathbf{o}}_p - (\mathbf{x}_p - \mathbf{c}_i)]^2, \quad (2)$$

where I is the number of instances in the batch, and P_i is the number of points in the instance i. During inference, the predicted center for each point can then be calculated as $\hat{\mathbf{c}}_p = \hat{\mathbf{x}}_p - \hat{\mathbf{o}}_p$.

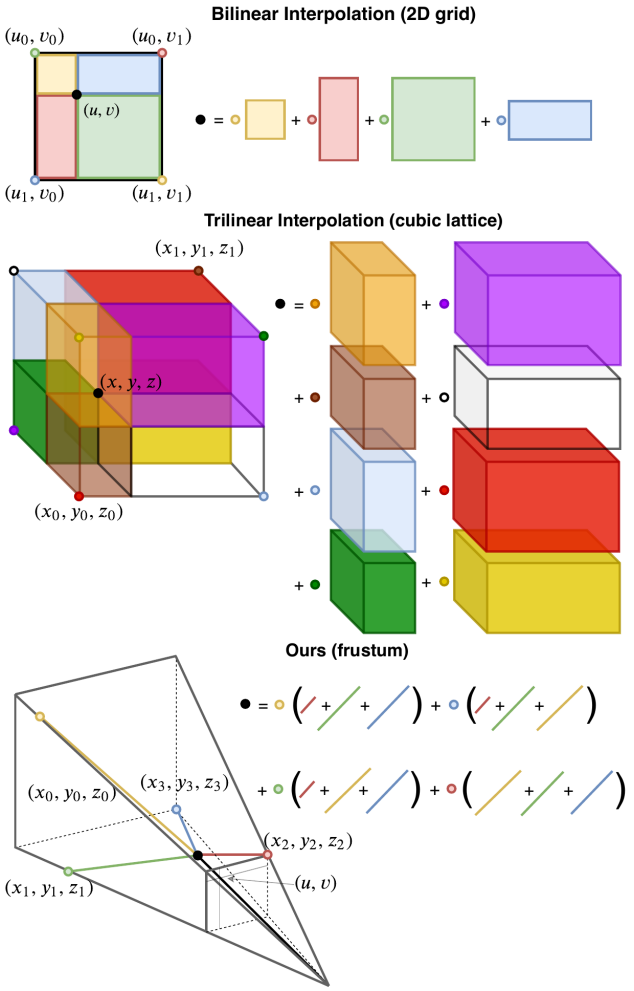

Fig. 4: Upsampling methods graphically. The black dot corresponds to the desired interpolated feature value, while the colored dots represent the values to interpolate from. Top: Bilinear upsampling. Center: Trilinear Interpolation. Bottom: Ours. The vertex values are obtained in the range image domain, but their distances to the query point are calculated in 3D as their real euclidean distance. Then the interpolated feature value is a linear combination of four the closest feature values in the lower resolution grid (which define the frustum), with the coefficients calculated as an inverse of the distance to the point, normalized by the total distance.

To use a region-based clustering method, an intra-cluster radius needs to be defined. For the center embedding $\hat{\mathbf{c}}_p$, the radius is the maximum Euclidean distance that we allow two points in the same instance to predict. In theory, this radius could be fixed, and chosen by cross-validation, but research shows that different points have different levels of accuracy [29], resulting in either over- or under-segmentation. Using a learned, adaptive radius for each point instead can help solve this problem, as shown in Fig. 5. Therefore, we add three convolutional blocks on top of the offset encoder that predict this radius $\hat{\epsilon}_p$ for each point, estimating the radius as the Euclidean distance between the offset prediction and its ground truth, for each point, during the training, i.e,

$$\mathcal{L}_{CENT\ ERR} = \frac{1}{I}\sum_{i=1}^{I}\frac{1}{P_i}\sum_{p=1}^{P_i}\left[\hat{\epsilon}_p - \|\hat{\mathbf{o}}_p - (\mathbf{x}_p - \mathbf{c}_i)\|\right]^2 \quad (3)$$

The intuition behind using the training offset error as the ground truth for this variable radii is not to estimate the uncertainty of each prediction, which would not be possible, since the training error is usually significantly lower than the test time error. Instead, we are interested in learning the *relative difficulty* of different points belonging to the same instance. Fig. 5 shows an example where the point in the fender is significantly more inaccurate than the point in the center of the car, requiring a larger radius to be properly assigned to this car's cluster. This is fixable by simply using a larger radius for all points, but this leads to under-segmentation in many cases in our dataset, preventing us from a proper understanding of the scene. Therefore, at inference time we use this learned tolerances as relative thresholds for each point, but adjust the overall scale by a constant factor to account for the difference between training and validation error. This factor is the ratio between the latter.

2) Semantic decoder: This decoder predicts a 32-dimensional semantic embedding for each point in the range image representation. From this embedding, two $[1 \times 1]$ convolutional heads are used to predict classes and categories. In order to train the network to predict both, we use a cross-entropy loss for the classes and an analogous one for the categories, of the form:

$$\mathcal{L}_{SEM} = -\sum_{c=1}^{C} w_c \, y_c \, \log\left(\hat{y}_c\right), \tag{4}$$

where $w_c = \frac{1}{\log\left(f_c + \epsilon\right)}$ is a class-wise weight calculated as a function of the inverse class-frequency f_c, and ϵ limits the largest possible weight for a class. In practice, the category prediction does not need its own head or loss. However, our experiments show that adding a separate head, along with a category loss, improves the category segmentation, effectively mapping semantically similar classes together in embedding space.

To aid the instance segmentation head center predictions, we also add an auxiliary loss that applies directly to the embeddings from the semantic head. Fig. 3 shows a random projection from the 32-dimensional embeddings to 3 dimensions, plotted as RGB values. It is possible to see that not only the embeddings are different from class to class, but also between different instances of the same class. This loss uses metric learning to cluster all pixel embeddings $\hat{\mathbf{e}}_p$ of the same instance close to each other, and pushing all mean embeddings of different instances away from each other. This is done through an attraction and a repulsion loss, of the form:

$$L_{ATTRACT} = \frac{1}{I}\sum_{i=1}^{I}\frac{1}{P_i}\sum_{p=1}^{P_i}\left(\hat{\mathbf{e}}_{i,p} - \hat{\bar{\mathbf{e}}}_i\right)^2 \tag{5}$$

$$L_{REPEL} = \frac{1}{I(I-1)}\sum_{\substack{i_A=1 \\ i_A \neq i_B}}^{I}\sum_{i_B=1}^{I}\frac{1}{\left(\hat{\bar{\mathbf{e}}}_{i_A} - \hat{\bar{\mathbf{e}}}_{i_B}\right)^2}, \tag{6}$$

where $\hat{\mathbf{e}}_{i,p}$ is the embedding for pixel p of instance i, and $\hat{\bar{\mathbf{e}}}_i$ is the mean embedding of all points in instance i.

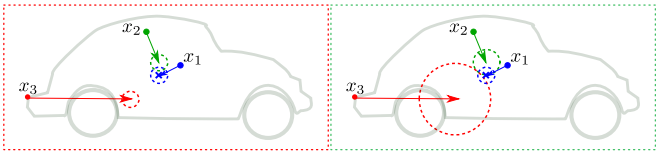

Fig. 5: Influence of clustering radius. Left: Wrong clustering due to fixed radii. Right: Correct clustering using the learned radii depending on point difficulty.

Analogously to the center offsets, we also predict an error radius for each pixel embedding that is later used for the adaptive clustering, following an analogous loss to the center offset error loss:

$$\mathcal{L}_{EMBED\ ERR} = \frac{1}{I}\sum_{i=1}^{I}\frac{1}{P_i}\sum_{p=1}^{P_i}\left[\hat{\epsilon}_p - \left\|\hat{\mathbf{e}}_{i,p} - \hat{\bar{\mathbf{e}}}_i\right\|\right]^2, \tag{7}$$

D. Instance Extractor

To obtain all individual instance IDs, we employ an iterative procedure. First, we sample a *thing* point from the semantic prediction and obtain its distance in feature space to all other points in the same class. We propose two alternative features to do this through the instance and the semantic decoders. Using the instance decoder, the features represent the prediction of the center of the instance $\hat{\mathbf{c}}_i = \mathbf{x}_p - \hat{\mathbf{o}}_p$. Using the semantic decoders, the features used are the embeddings of each point $\hat{\mathbf{e}}_{i,p}$. In both cases, we then use the predictions of the error for each pixel as the clustering radius to consider points as belonging to the same instance as the initially sampled point. We then assign the instance ID to all points within the radius, remove these points from the pool, and start over with a new sampled point, until all points are consumed. In our ablation study, we compare all features and the usage of the learned radius vs. a statically defined threshold chosen by cross-validation.

E. Point Cloud Extraction and Post-Processing

After the segmentation of the point cloud as a range image, recovering all labels for the N original points in the point cloud is desired. However, if $N > HW$, unprojecting the image using the intrinsic calibration of the sensor does not reconstruct all points. This is why the reprojection of the labeled points to the 3D world is, instead, performed by keeping an $(N, 2)$ shaped list of (u, v) image indexes that can use the label image as a lookup table to recover the labels and IDs for all N points.

IV. Experimental Evaluation

In the first part of the experiments, we compare our panoptic segmentation approach with state-of-the-art approaches on SemanticKITTI [1], [2], a large-scale LiDAR dataset providing instance and semantic segmentation annotations, as well as a panoptic segmentation benchmark. We then provide ablations studies to show the importance of different design decisions of our approach, such as the use of semantic and center embeddings jointly, the inclusion of the novel trilinear upsampling module, and the category loss.

Implementation details. In all the following experiments, we use the following parameters, when not otherwise stated. All networks in the single-stage approach were trained following the same training schedule, using Adam optimizer with a learning-rate of $1 \cdot 10^{-4}$, a warm-up ramp of 1 epoch, momentums $(0.9, 0.99)$, a learning-rate decay of 0.99 per epoch, training for 200 epochs. To integrate all losses, we tested GradNorm [4], but yielded no significant improvement over the simple addition of all losses.

Dataset. We evaluate our approach on SemanticKITTI [1], which provides point-wise semantic annotations for all scans of the KITTI odometry split [8]. Recently, we extended the dataset by providing temporally consistent instance annotations for all scans in the dataset [2]. The dataset provides 23,201 scans for training and the remaining 20,351 scans are used for evaluation on a benchmark server. We use sequence 08 from the training data comprised of 4,071 scans for validation purposes. The dataset contains overall 28 classes from which the vehicle classes and classes representing humans have point-wise instance annotations.

Evaluation Metrics. In order to compare the semantic segmentation branch with other approaches in the semantic segmentation benchmark, we calculate the mean intersection over union (mIoU) over all classes, defined as:

$$\text{mIoU} = \frac{1}{|\mathcal{Y}|} \sum_{c \in \mathcal{Y}} \frac{|\{i \mid y_i = c\} \cap \{j \mid \hat{y}_j = c\}|}{|\{i \mid y_i = c\} \cup \{j \mid \hat{y}_j = c\}|}, \quad (8)$$

where y_i corresponds to the ground truth label of point $\vec{p}_i$ and $\hat{y}_i$ to the prediction.

To measure the quality of the joint semantic and instance segmentation for each class, we use the recently proposed panoptic quality (PQ) [16], [2]:

$$\text{PQ} = \frac{1}{|\mathcal{Y}|} \sum_{c \in \mathcal{Y}} \frac{\sum_{(\mathcal{S},\hat{\mathcal{S}}) \in \text{TP}_c} \text{IoU}(\mathcal{S}, \hat{\mathcal{S}})}{|\text{TP}_c| + \frac{1}{2}|\text{FP}_c| + \frac{1}{2}|\text{FN}_c|}, \quad (9)$$

where $\text{IoU}(\mathcal{S}, \hat{\mathcal{S}})$ is the intersection over union between the prediction segment $\hat{\mathcal{S}}$ and the ground truth segment $\mathcal{S}$, TP_c are the pairs of predicted segments $\hat{\mathcal{S}}$ that present over 50% intersection over union (IoU) with a ground truth segment $\mathcal{S}$, FP_c the set of unmatched predicted segments $\hat{\mathcal{S}}$, and FN_c the set of unmatched ground truth segments $\mathcal{S}$. The overall panoptic quality (PQ) metric averaged over all classes, which makes the metric insensitive to class imbalance.

For *stuff* classes, Porzi *et al*. [31] use an alternative metric that considers the specific case of only one segment per image, where the segment- and IoU-based criterion often leads to unmatched predictions (FP):

$$\text{PQ}_c^{\dagger} = \begin{cases} \text{IoU}(\mathcal{S}, \hat{\mathcal{S}}) & \text{, if c is a } \textit{stuff} \text{ class} \\ \text{PQ}_c & \text{, otherwise.} \end{cases} \quad (10)$$

Lastly, SemanticKITTI defines an ontology assigning each class to a category, e.g., *truck*, *car*, *other-vehicle* belong to the category *vehicle* [2]. These category definitions are useful for autonomous driving, e.g. identifying *humans* as an alternative to the more fine-grained *person* or *bicyclist* classes. Therefore, alternatively to the class-wise metrics, we also evaluate all approaches with respect to categories.

A. Comparison with the State of the Art

The first experiment evaluates the performance of our single-stage approach in comparison with the two-stage approaches proposed as baselines for the panoptic task in [2].

Tab. I shows the results in terms of class-wise performance on the test set. Likewise, Tab. II shows the performance in respect to the categories. Our proposed single-stage approach gets superior performance in comparison with the two-stage approach using RangeNet++ [27]. However, it is worse than the best performing KPConv [41], which achieves 44.5 panoptic quality and can be mainly attributed to better semantic segmentation, albeit at a higher computational cost.

We also show in the results the difference between using the category loss and head, vs a lookup table between class prediction and corresponding category. Interestingly, these results show that the category loss helps to improve the panoptic quality performance for *thing* classes, but leads to worse results with respect to the semantic segmentation quality as shown by a drop in mIoU.

Nevertheless, the main motivation for our single-stage approach is the improved computational efficiency in comparison to the aforementioned two-stage approaches. Instead of using multiple different networks, we can use a single multi-task network, which also profits from sharing the encoder between different tasks. Fig. 6 shows the runtime performance in relation to the panoptic quality. Our single-stage approach is considerably faster than the two-stage approaches. Here, we assume that the separate object detectors runs in parallel (314 ms for *pedestrian/cyclist* and 105 ms for *car*) after the semantic segmentation (200 ms for KPConv and 95 ms for RangeNet++) resulting in 514 ms and 409 ms respectively. Our single-stage approach with trilinear upsampling takes 85 ms on average.

B. Ablation Studies

We also validate that our contributions lead to an increase in performance with respect to panoptic quality through ablation studies. Note that here we evaluate all approaches on the validation set.

The first experiment, cf. Tab. III, shows the influence of the upsampling method to regain spatial resolution in the decoder after backbone downsampling. We can see that using our trilinear evaluation leads to considerable gains in class and computational performance, allowing for more downsampling without sacrificing in accuracy or speed.

The next experiment, cf. Tab. IV, shows the influence of features used for clustering instances. We can see that the best method using a single head is through the prediction of the center instances, rather than the semantic embeddings. However, combining both yields an increase in the performance of the clustering, which is appreciated by an increase in the recognition quality of *things*. Furthermore, we compare the clustering using each feature with the learned radii vs the best static threshold, found by cross-validation, and we show that learning a point-wise radius helps the performance of the approach.

Method	FPS	mIoU	PQ	PQ†	RQ	SQ	PQ^{Th}	RQ^{Th}	SQ^{Th}	PQ^{St}	RQ^{St}	SQ^{St}
KPConv [41] + PointPillars [20]	1.9	58.8	44.5	52.5	54.4	80.0	32.7	38.7	81.5	53.1	65.9	79.0
RangeNet++ [27] + PointPillars [20]	2.4	52.4	37.1	45.9	47.0	75.9	20.2	25.2	75.2	49.3	62.8	76.5
Ours (**without** category loss)	11.8	51.0	35.3	44.3	45.0	76.5	19.1	24.1	76.7	47.2	60.2	76.4
Ours (**with** category loss)	11.8	50.9	38.0	47.0	48.2	76.5	25.6	31.8	76.8	47.1	60.1	76.2

TABLE I: Comparison of test set results on SemanticKITTI using *stuff*(St) and *thing*(Th) *classes*. All results in [%].

Method	FPS	mIoU	PQ	PQ†	RQ	SQ	PQ^{Th}	RQ^{Th}	SQ^{Th}	PQ^{St}	RQ^{St}	SQ^{St}
KPConv [41] + PointPillars [20]	1.9	84.8	70.0	71.5	79.2	87.3	54.3	62.1	86.7	77.8	87.8	87.7
RangeNet++ [27] + PointPillars [20]	2.4	78.8	63.6	65.9	74.3	83.8	43.8	52.3	82.2	73.5	85.3	84.6
Ours (**without** category loss)	11.8	77.4	59.9	62.6	71.4	81.7	37.7	47.2	78.0	71.0	83.4	83.5
Ours (**with** category loss)	11.8	77.8	65.8	68.1	77.2	83.5	53.6	63.8	82.5	71.9	84.0	84.0

TABLE II: Comparison of test set results on SemanticKITTI using *stuff*(St) and *thing*(Th) *categories*. All results in [%].

OS	Method	mIoU	PQ	PQ†	FPS
32	Transpose convolution	46.4	30.8	41.9	11.1
8	Transpose convolution	48.5	31.2	42.6	4.2
32	Ours (trilinear)	50.7	36.5	46.1	11.8

TABLE III: Influence of upsampling method evaluated on validation set with respect to panoptic quality and runtime.

Center	Embedding	Learnt Radius	RQ^{Th}	PQ
✓			23.7	34.4
✓		✓	26.7	35.8
	✓		18.0	32.8
	✓	✓	20.0	33.3
✓	✓	✓	28.2	36.5

TABLE IV: Features used for clustering of *things* on validation set with respect to recognition and panoptic quality.

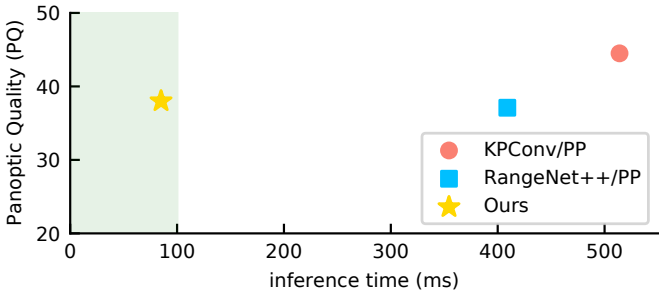

Fig. 6: Runtime of the evaluated approaches. Green area represents the zone of approaches faster than the rate of the sensor.

V. Conclusion

In this paper, we propose a novel approach for single-stage LiDAR-based panoptic segmentation which achieves high panoptic quality while still running over the frame rate of the sensor. Our experiments show that our approach achieves results that are on par with, but faster than the best performing real-time approach on SemanticKITTI. Our ablations studies also show that the addition of the novel range-image-based trilinear upsampling module allows our approach to use larger output strides than approaches using transpose convolutions, resulting in faster runtime without sacrificing accuracy. Furthermore, we show that the combination of our geometric and semantic feature embeddings helps increase the performance of the approach in terms of recognition quality. This is also the case for our learned point-wise radii, which adapts the clustering threshold for points of different difficulty. Finally, the addition of a category loss makes category-based results more robust. This means that if we get a class wrong, we will more likely confuse it with an instance of the same category, which is desired behavior.

References

[1] 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 International Conf. on Computer Vision (ICCV)*, 2019.

[2] J. Behley, A. Milioto, and Cyrill Stachniss. A Benchmark for LiDAR-based Panoptic Segmentation based on KITTI. *arXiv preprint*, 2020.

[3] B. De Brabandere, D. Neven, and L. Van Gool. Semantic instance segmentation with a discriminative loss function. In *Deep Learning for Robotic Vision workshop, IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[4] Z. Chen, V. Badrinarayanan, C.Y. Lee, and A. Rabinovich. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. *arXiv preprint*, 2017. Proceedings of the 35th International Conference on Machine Learning (2018), 793-802.

[5] B. Cheng, M.D. Collins, Y. Zhu, T. Liu, T.S. Huang, H. Adam, and L. Chen. Panoptic-DeepLab. In *Proc. of the ICCV Workshop: Joint COCO and Mapillary Recognition Challlenge Workshop*, 2019.

[6] M. Cordts, S. Mohamed Omran, Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] F. Engelmann, T. Kontogianni, J. Schult, and B. Leibe. Know What Your Neighbors Do: 3D Semantic Segmentation of Point Clouds. *arXiv preprint*, 2018.

[8] 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.

[9] 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.

[10] F. Groh, P. Wieschollek, and H. Lensch. Flex-Convolution (Million-Scale Pointcloud Learning Beyond Grid-Worlds). In *Proc. of the Asian Conf. on Computer Vision (ACCV)*, Dezember 2018.

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

[12] J. Hou, A. Dai, and M. Niessner. 3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[13] B. Hua, M. Tran, and S. Yeung. Pointwise Convolutional Neural Networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[14] J. Huang and S. You. Point Cloud Labeling using 3D Convolutional Neural Network. In *Proc. of the Internation Conference on Pattern Recognition (ICPR)*, 2016.

[15] M. Jiang, Y. Wu, and C. Lu. PointSIFT: A SIFT-like Network Module for 3D Point Cloud Semantic Segmentation. *arXiv preprint*, 2018.

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

[17] R. Klukov and V. Lempitsky. Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017.

[18] J.D. Lafferty, A. McCallum, and F.C.N. Pereira. Conditional Random Fields: Probabilistic Models for Segmentating and Labeling Sequence Data. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2001.

[19] L. Landrieu and M. Simonovsky. Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[20] A.H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. PointPillars: Fast Encoders for Object Detection From Point Clouds. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[21] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature*, 521:436–444, 2015.

[22] Y. Li, X. Chen, Z. Zhu, L. Xie, G. Huang, D. Du, and X. Wang. Attention-Guided Unified Network for Panoptic Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[23] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 740–755, 2014.

[24] H. Liu, C. Peng, C. Yu, J. Wang, X. Liu, G. Yu, and W. Jiang. An End-To-End Network for Panoptic Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[25] D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[26] A. Milioto, L. Mandtler, and C. Stachniss. Fast Instance and Semantic Segmentation Exploiting Local Connectivity, Metric Learning, and One-Shot Detection for Robotics. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

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

[28] G. Neuhold, T. Ollmann, S. Rota Bulo, and P. Kontschieder. The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017.

[29] D. Neven, B. De Brabandere, M. Proesmans, and L. Van Gool. Instance Segmentation by Jointly Optimizing Spatial Embeddings and Clustering Bandwidth. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[30] Q.H. Pham, D.T. Nguyen, B.S. Hua, G. Roig, and S.K. Yeung. JSIS3D: Joint Semantic-Instance Segmentation of 3D Point Clouds With Multi-Task Pointwise Networks and Multi-Value Conditional Random Fields. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[31] L. Porzi, S. Rota Bulo, A. Colovic, and P. Kontschieder. Seamless Scene Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[32] 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.

[33] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J .Guibas. Deep Hough Voting for 3D Object Detection in Point Clouds. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2019.

[34] C.R. 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 Advances in Neural Information Processing Systems (NIPS)*, 2017.

[35] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. *arXiv preprint*, 2018.

[36] D. Rethage, J. Wald, J. Sturm, N. Navab, and F. Tombari. Fully-Convolutional Point Networks for Large-Scale Point Clouds. *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2018.

[37] G. Riegler, A. Ulusoy, and A. Geiger. OctNet: Learning Deep 3D Representations at High Resolutions. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[38] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M-H. Yang, and J. Kautz. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[39] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese. SEGCloud: Semantic Segmentation of 3D Point Clouds. In *Proc. of the International Conference on 3D Vision (3DV)*, 2017.

[40] G. Te, W. Hu, Z. Guo, and A. Zheng. RGCNN: Regularized Graph CNN for Point Cloud Segmentation. *arXiv preprint*, 2018.

[41] H. Thomas, C.R. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L.J. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. *arXiv preprint*, 2019.

[42] W. Wang, R. Yu, Q. Huang, and U. Neumann. SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[43] X. Wang, S. Liu, X. Shen, C. Shen, and J. Jia. Associatively Segmenting Instances and Semantics in Point Clouds. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[44] Z. Wang and F. Lu. VoxSegNet: Volumetric CNNs for Semantic Part Segmentation of 3D Shapes. *arXiv preprint*, 2018.

[45] 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 *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[46] 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. *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[47] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun. UPSNet: A Unified Panoptic Segmentation Network. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[48] B. Yang, J. Wang, R. Clark, Q. Hu, S. Wang, A. Markham, and N. Trigoni. Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[49] L. Yi, W. Zhao, H. Wang, M. Sung, and L. Guibas. GSPN: Generative Shape Proposal Network for 3D Instance Segmentation in Point Cloud. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[50] W. Zeng and T. Gevers. 3DContextNet: K-d Tree Guided Hierarchical Learning of Point Clouds Using Local and Global Contextual Cues. *arXiv preprint*, 2017.