Efficient LiDAR Point Cloud Oversegmentation Network

Le Hui[1], Linghua Tang[2], Yuchao Dai[1], Jin Xie[2]* and Jian Yang[2]*
[1]Northwestern Polytechnical University, [2]Nanjing University of Science and Technology
huile@nwpu.edu.cn, {tanglinghua, csjxie, csjyang}@njust.edu.cn, daiyuchao@gmail.com

Abstract

Point cloud oversegmentation is a challenging task since it needs to produce perceptually meaningful partitions (i.e., superpoints) of a point cloud. Most existing oversegmentation methods cannot efficiently generate superpoints from large-scale LiDAR point clouds due to complex and inefficient procedures. In this paper, we propose a simple yet efficient end-to-end LiDAR oversegmentation network, which segments superpoints from the LiDAR point cloud by grouping points based on low-level point embeddings. Specifically, we first learn the similarity of points from the constructed local neighborhoods to obtain low-level point embeddings through the local discriminative loss. Then, to generate homogeneous superpoints from the sparse LiDAR point cloud, we propose a LiDAR point grouping algorithm that simultaneously considers the similarity of point embeddings and the Euclidean distance of points in 3D space. Finally, we design a superpoint refinement module for accurately assigning the hard boundary points to the corresponding superpoints. Extensive results on two large-scale outdoor datasets, SemanticKITTI and nuScenes, show that our method achieves a new state-of-the-art in LiDAR oversegmentation. Notably, the inference time of our method is 100× faster than that of other methods. Furthermore, we apply the learned superpoints to the LiDAR semantic segmentation task and the results show that using superpoints can significantly improve the LiDAR semantic segmentation of the baseline network. Code is available at https://github.com/fpthink/SuperLiDAR.

1. Introduction

In modern self-driving cars, 3D LiDAR sensor acquires precise distance measurements of surrounding objects and their surface characteristics for large-scale outdoor scene understanding. In recent years, LiDAR semantic segmentation [21] has been widely studied, and a variety of approaches have emerged with impressive results. However, LiDAR oversegmentation is rarely explored in 3D computer vision. Unlike LiDAR semantic segmentation, LiDAR oversegmentation outputs the perceptually meaningful tessellation of point cloud. The resulting superpoint is a set of points, which are semantically and geometrically homogeneous in the local regions of the objects. The superpoint representation can adaptively and flexibly represent local geometric structures of objects. Therefore, studying LiDAR oversegmentation is very meaningful for LiDAR point cloud based applications. However, the sparsity, noise, and irregularity of LiDAR point clouds bring great challenges to LiDAR oversegmentation.

Early point cloud oversegmentation methods are usually optimization based methods. Lin *et al.* [17] casted superpoint segmentation problem as a subset selection problem, and developed a heuristic algorithm that utilizes handcrafted local information of the point cloud to segment superpoints by minimizing the energy function. Guinard *et al.* [8] formulated point cloud oversegmentation as a structured optimization problem and used handcrafted local descriptors to produce geometrically simple superpoints through a greedy graph-cut algorithm [14]. However, due to sparse LiDAR point clouds, the computed handcrafted features are less discriminative, so the resulting superpoints cannot produce clear boundaries between similar objects. Landrieu *et al.* [13] introduced a deep network to extract point embeddings, which are used to replace handcrafted features in [14] for segmenting superpoints. Since it is a two-stage method, the processing procedure of superpoint segmentation is complex and time-consuming. Recently, Hui *et al.* [11] proposed an end-to-end superpoint network that iteratively learns the point-superpoint association map for clustering superpoints. Nonetheless, it requires postprocessing to filter the noise points. In short, due to complex and inefficient procedures, the above methods cannot efficiently generate superpoints from large-scale LiDAR point clouds.

*Corresponding authors.

Le Hui and Yuchao Dai are with Shaanxi Key Laboratory of Information Acquisition and Processing, China. Linghua Tang, Jin Xie and Jian Yang are with Key Lab of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, and Jiangsu Key Lab of Image and Video Understanding for Social Security, China.

In this paper, we propose a simple yet efficient LiDAR oversegmentation network called SuperLiDAR, which directly outputs superpoint from LiDAR point clouds without any additional processing procedures. The key idea of segmenting superpoints is to group points based on low-level point embeddings. Specifically, to learn low-level point embeddings, we first formulate it as a deep metric learning problem structured by a local neighborhood defined on the point cloud. We introduce the local discriminative loss to embed 3D points within the local neighborhoods of the same object, thereby ensuring the point embeddings are similar to each other. After obtaining low-level point embeddings, we then propose a LiDAR point grouping algorithm, which groups the points to generate superpoints via the breadth-first search (BFS). By using the similarity of point embeddings and Euclidean distance of 3D point coordinates, we apply the BFS algorithm to produce compact superpoints. Finally, we propose a superpoint refinement module, which learns the affinity between the hard boundary point and its k-nearest candidate superpoints. By assigning the hard boundary point with the corresponding superpoint with the highest similarity, we can obtain high-quality superpoints. Notably, our LiDAR oversegmentation network can flexibly integrate with downstream tasks, such as semantic segmentation. In order to evaluate the effectiveness of the learned superpoints, we introduce a simple multi-scale superpoint aggregation module for LiDAR semantic segmentation. We conduct experiments on two large-scale benchmarks, SemanticKITTI and nuScenes, to demonstrate the effectiveness of our method. LIDAR oversegmentation experiments show that the proposed method not only achieves state-of-the-art performance, but also is 100$\times$ faster than other methods. Furthermore, LiDAR semantic segmentation experiments demonstrate that using superpoint can significantly improve the performance of the baseline network.

The contributions of this paper are as follows:

- We propose an efficient LiDAR oversegmentation network for learning superpoint segmentation from large-scale LiDAR point clouds.
- Our method achieves state-of-the-art performance in LiDAR oversegmentation while being 100$\times$ faster than current oversegmentation methods.
- We demonstrate that the proposed LiDAR oversegmentation network can be integrated into LiDAR semantic segmentation network in an end-to-end manner and further improve the performance of LiDAR semantic segmentation.

2. Related Work

Point Cloud Oversegmentation Existing point cloud oversegmentation methods can be roughly grouped into two categories: optimization based methods and deep learning based methods. Optimization based methods usually utilize handcrafted descriptors to extract point features for superpoint segmentation. Papon *et al.* [21] were the pioneers in the field of oversegmentation in 3D data. The proposed voxel cloud connectivity segmentation method uses color and depth information as well as three dimensional geometric relationships to segment RGB-D data into superpoints. Lin *et al.* [17] formulated the superpoint segmentation problem as a subset selection problem. Based on handcrafted local information of the point cloud, the superpoint is segmented by directly minimizing the energy function. Guinard *et al.* [8] regarded point cloud oversegmentation as a structured optimization problem. They used the greedy graph-cut algorithm [14] to produce geometrically simple superpoints, where the handcrafted local descriptors (such as linearity, planarity, scattering, and verticality) are used to describe the local geometry of each point. However, when there are objects with complex local geometric structures in the point clouds of the surrounding environments, the handcrafted features usually cannot provide the discriminative features to produce high-quality superpoints, especially in sparse LiDAR point clouds.

Deep learning based methods leverage the deep network to extract discriminative point features for improving oversegmentation performance. Guinard *et al.* [13] cased point cloud oversegmentation as a deep metric learning problem. A graph-structured contrastive loss is proposed to learn discriminative point features. By using the learned features to replace the handcrafted features used in the greedy graph-cut algorithm [14], it can generate higher quality superpoints from the point cloud than [8]. However, it cannot be end-to-end trained due to the two-stage superpoint segmentation strategy. Recently, Hui *et al.* [11] proposed an end-to-end superpoint network to cluster superpoints by iteratively learning the point-superpoint association map. Nonetheless, the clustered superpoints may have much noise near the boundary of the superpoints, especially in the sparse LiDAR point clouds. Therefore, post-processing is required to filter noise, increasing the complexity of its application as well as the time cost.

Current point cloud oversegmentation methods are limited by the complex processing procedures, which make them unable to be flexibly applied to downstream tasks. Therefore, it is necessary to design an efficient point cloud oversegmentation with scalability and flexibility.

LiDAR Semantic Segmentation LiDAR semantic segmentation [30, 40, 9, 6, 27, 34, 42, 24, 26, 10] has emerged with various methods based on different representations of point clouds. In order to utilize conventional 2D segmentation schemes, projection-based methods that project the LiDAR point cloud into an image plane, obtaining rendered images [16], snapshots [2], tangent

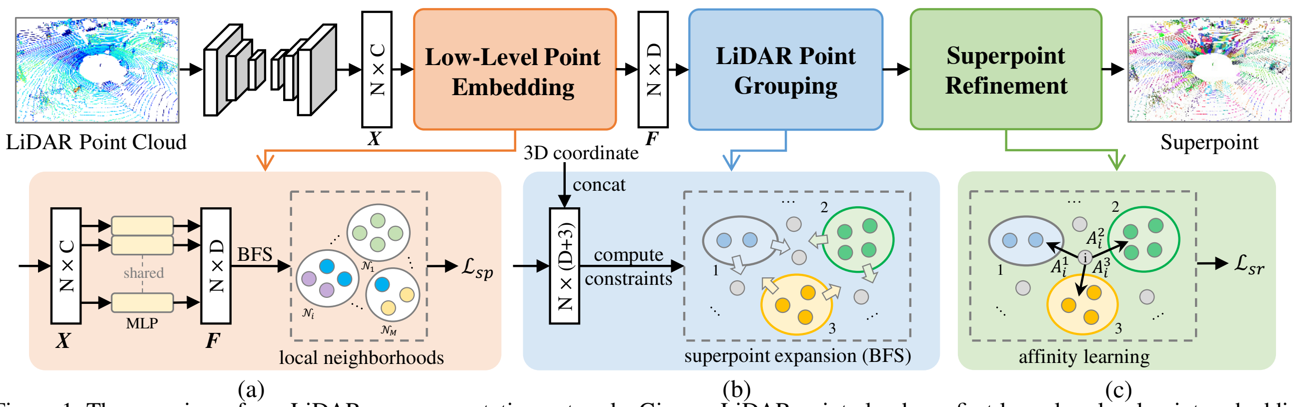

Figure 1. The overview of our LiDAR oversegmentation network. Given a LiDAR point cloud, we first learn low-level point embedding after extracting point features by the sparse 3D network. Then, we propose the LiDAR point grouping algorithm to generate superpoints from the point cloud. Finally, we use the superpoint refinement module to assign unallocated free points to the corresponding superpoints.

images [28], range images [32, 33], and bird's-eye view images [18]. Although projection-based methods are efficient for LiDAR segmentation, the spatial geometric information is inevitably lost due to space compression. In order to directly utilize 3D geometric information, point-based methods [22] generally use multi-layer perceptron network to handle 3D LiDAR points by using different designed local descriptors, including max pooling function [23, 31], adaptive weighting [29, 41], and non-local weighting [38, 4]. Despite higher performance compared with projection-based methods, point-based methods are usually time-consuming due to complex local aggregation operations and a large number of points in large-scale LiDAR point clouds. Recently, most advanced techniques are voxel based methods, which use sparse convolution [19] to process large-scale LiDAR point clouds. Compared with traditional 3D convolution, it only performs convolution operations on non-empty voxels, which can greatly reduce computational cost and memory consumption. Based on SparseConv, different methods use network architecture search [25], multi-scale feature aggregation [5], cylindrical partition and asymmetrical 3D convolution network [43], range-point-voxel fusion network [35] and multi-modal network [37] to boost segmentation performance. Although voxel based methods have achieved impressive results, the quality of the voxel is sensitive to its resolution. The larger the voxel size is, the lower the quality is. This motivates us to explore the superpoint representation of LiDAR point clouds.

3. Methodology

3.1. LiDAR Oversegmentation Network

Low-Level Point Embedding For LiDAR oversegmentation, we use the low-level embedding of point clouds to segment superpoints. As shown in Fig. 1, we formulate it as a deep metric learning problem structured by the local neighborhoods defined on the point clouds.

Specifically, given a LiDAR point cloud $P = \{\boldsymbol{p}_i\}_{i=1}^N$, we follow [37] and use the sparse 3D network to obtain the feature map $\boldsymbol{X} \in \mathbb{R}^{N \times C}$, where N is the number of neighborhoods. And we use a multi-layer perceptron (MLP) to map $\boldsymbol{X}$ into low-dimension embedding $\boldsymbol{F} \in \mathbb{R}^{N \times D}$. By computing the Euclidean distance between a point and its surrounding points, we construct the local neighborhoods for each point, denoted by $\mathcal{N} = \{\mathcal{N}_i\}_{i=1}^N$. Each local neighborhood $\mathcal{N}_i$ is a set of 3D points, containing the points lying in the i-th local neighborhood.

After constructing the local neighborhoods of the LiDAR point cloud, we develop a local discriminative loss $\mathcal{L}_{sp}$ to map the point features into the low-dimension embedding space with similar local geometric structures. For the i-th local neighborhood $\mathcal{N}_i$, we can identify different semantic parts according to the point labels, *i.e.*, ground truth semantic labels (circles of different colors in Fig. 1(a)). In the constructed local neighborhood with limited scope, points with the same semantic label can be regarded as having similar geometric structures. The local discriminative loss $\mathcal{L}_{sp}$ consists of two terms:

$$\mathcal{L}_{sp} = \mathcal{L}_{ident} + \mathcal{L}_{dist} \tag{1}$$

where the identical term $\mathcal{L}_{ident}$ draws the point embedding towards the mean embedding of the corresponding semantic part, and the distance term $\mathcal{L}_{dist}$ pushes the point embedding away from other semantic parts. The detailed formulation are as follows:

$$\mathcal{L}_{ident} = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}\sum_{k=1}^{T_i}\mathbb{I}(j,k)\cdot[\|\boldsymbol{z}_k - \boldsymbol{f}_j\|_2 - \alpha]_+^2, \tag{2}$$

$$\mathcal{L}_{dist} = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{|\mathcal{N}_i|\,(T_i-1)}\sum_{j\in\mathcal{N}_i}\sum_{k=1}^{T_i}(1-\mathbb{I}(j,k))\cdot [2\beta - \|\boldsymbol{z}_k - \boldsymbol{f}_j\|_2]_+^2 \tag{3}$$

where $\mathbb{I}(j,k)$ is the indicator function. $\mathbb{I}(j,k)$ equals to 1 if point j belongs to the k-th semantic part, and 0, otherwise. T_i is the number of semantic parts in the i-th local neighborhood $\mathcal{N}_i$. In addition, $\boldsymbol{f}_j \in \mathbb{R}^D$ represents the point embedding of the i-th points, and $\boldsymbol{z}_k \in \mathbb{R}^D$ represents the mean point embedding of the corresponding semantic part that contains the i-th point. In the experiment, the thresholds α and β are set to 0.01 and 0.2, respectively.

LiDAR Point Grouping To increase the inference speed of superpoint segmentation, we propose a simple yet efficient LiDAR point grouping algorithm. As shown in Fig. 1, the key idea is to group points based on the learned low-level point embeddings by applying the breadth-first search (BFS) algorithm.

Specifically, in order to generate homogeneous and compact superpoints in the 3D space, we simultaneously consider the embedding space and 3D coordinate space. Given the LiDAR point cloud, we first randomly select a seed point as the BFS starting point, which has not been assigned to a superpoint. Then, the BFS algorithm is executed based on the starting point to group the surrounding points to form a superpoint. The procedure is shown in Fig. 1(b). During BFS, if a point can be assigned to a corresponding superpoint, it should meet two constraints. For one constraint, the Euclidean distance between point embeddings should be less than the threshold β, which is regarded as the margin used to push the point embeddings away in Eq. (3). For another constraint, the Euclidean distance between points should be less than the threshold γ, which is used to maintain the compactness of the superpoints in the 3D coordinate space. Note that during BFS, if the superpoint size (*i.e.*, the number of points within the superpoint) is greater than the maximum size N_{max}, the current superpoint growth process will be terminated. After BFS, if the generated superpoint size is less than the minimum size N_{min}, the superpoint will be discarded. The LiDAR point grouping algorithm repeats the BFS procedure to generate new superpoints from the LiDAR point cloud until it cannot meet the generation conditions. The detailed procedure is shown in Algorithm 1.

In superpoint generation, clustering-based algorithms can also be adopted, but they will introduce noise points to the superpoints. In order to eliminate noise points, post-processing is required, which will bring additional time costs. In contrast, the BFS algorithm can generate good superpoints without noise points under the time complexity of $O(N)$, where N is the number of LiDAR points. Note that instead of performing BFS in sparse 3D space, we convert the sparse LiDAR point clouds into a dense range image. Due to the fact that the edge of adjacent points in the range image is the edge of adjacent pixel gird, which can greatly save the time of edge traversal, executing BFS on the range image can effectively reduce the BFS time of the 3D space. Therefore, BFS complexity is approximately $O(N)$.

Algorithm 1 LiDAR Point Grouping

Input: point coordinates $P = \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_N\}$; point embeddings $F = \{\boldsymbol{f}_1, \dots, \boldsymbol{f}_N\}$; threshold γ of the 3D coordinate space; threshold β of the embedding space; threshold N_{min} of minimum size of superpoint; threshold N_{max} of maximum size of superpoint

Output: superpoints $\mathbf{S} = \{S_1, \dots, S_O\}$

```
1:  initialize an empty superpoint set $\mathbf{S}$
2:  initialize an array $v$ (visited) of length $N$ with all zeros
3:  for $i = 1$ to $N$ do
4:    if $v_i == 0$ then
5:      initialize an empty queue $Q$
6:      initialize an empty set $S$
7:      $v_i = 1$; $Q$.pushBack($i$); $S$.add($i$)
8:      while $Q$ is not empty do
9:        $h = Q$.popFront()
10:       $j$ = FindClosestPoint($h$, $I$)
11:       // Distance Constraint
12:       $e_{ij} = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2^2$
13:       if $v_j == 0$ and $e_{ij} < \gamma$ then
14:         // Embedding Constraint
15:         $d_{ij} = \|\boldsymbol{f}_i - \boldsymbol{f}_j\|_2^2$
16:         if $d_{ij} > \beta$ and $S$.size() $< N_{max}$ then
17:           $v_j = 1$; $Q$.pushBack($j$); $S$.add($j$)
18:     if $S$.size() $> N_{min}$ then
19:       add $S$ to $\mathbf{S}$
20: return $\mathbf{S}$
```

Superpoint Refinement In LiDAR point grouping, we discard the generated superpoints that do not conform to the minimum size of superpoints, so some LiDAR points are still not assigned to any superpoints. A simple way to handle these unassigned LiDAR points is to assign the point to the nearest superpoint center in the coordinate space. However, the unassigned points are mostly boundary points. Thus, it is hard to accurately assign the boundary points to the corresponding superpoints according to the distance. To maintain the efficiency of superpoint segmentation, we propose a simple yet effective superpoint refinement module to accurately assign the point to the corresponding superpoint by learning the affinity between the point and its K-nearest superpoints, as shown in Fig. 1.

Specifically, given an unassigned LiDAR point i, we first find the K-nearest superpoints ($\mathcal{K}_i$) in the 3D coordinate space according to the Euclidean distance between the point and superpoints. The affinity between the point i and the superpoint $j \in \mathcal{K}_i$ is defined as:

$$\boldsymbol{A}_i^j = \mathrm{Concat}(\boldsymbol{f}_i; \boldsymbol{z}_j; \boldsymbol{f}_i - \boldsymbol{z}_j; \boldsymbol{p}_i - \boldsymbol{x}_j; \|\boldsymbol{p}_i - \boldsymbol{x}_j\|^2) \quad (4)$$

where $\mathrm{Concat}(\cdot)$ represents the concatenation operation. Vectors $\boldsymbol{f}_i - \boldsymbol{z}_j$ and $\boldsymbol{p}_i - \boldsymbol{x}_j$ capture the difference between the point and superpoint in the embedding and coordinate spaces, respectively. Note that $\boldsymbol{z}_j \in \mathbb{R}^D$ and $\boldsymbol{x}_j \in$

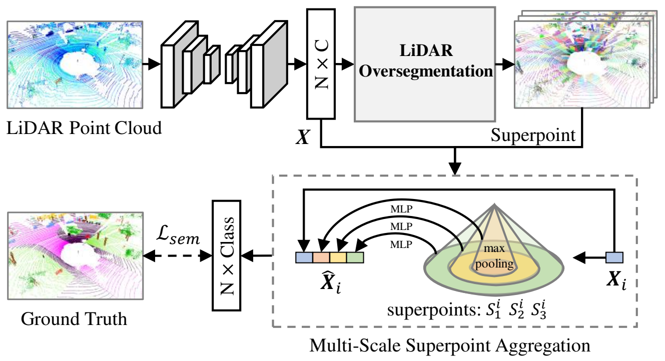

Figure 2. The overview of our end-to-end LiDAR semantic segmentation framework.

$\mathbb{R}^3$ are the embedding and coordinate of the superpoint. They are obtained by averaging the point coordinates and embeddings. In this way, we can obtain the affinity matrix $\boldsymbol{A}_i \in \mathbb{R}^{K\times(4D+1)}$ between the point and its K-nearest superpoints. After that, we map the affinity matrix to obtain the affinity score, which is defined as:

$$\boldsymbol{Y}_i = \boldsymbol{W}\boldsymbol{A}_i^\top \tag{5}$$

where $\boldsymbol{W} \in \mathbb{R}^{1\times(4D+1)}$ is the weight to be learned. Finally, we employ the softmax function to the affinity score $\boldsymbol{Y}_i \in \mathbb{R}^K$ for obtaining the affinity probability. By assigning the point i to the superpoint with the highest probability, we can generate high-quality superpoints from the LiDAR point clouds. Note that the ground truth of i-th point is the nearest superpoint with the same semantic label. During training, it is supervised by the cross-entropy loss, denoted by $\mathcal{L}_{sr}$ (see Fig. 1(c)).

3.2. Superpoint based LiDAR Segmentation

In order to show the scalability and flexibility of the proposed LiDAR oversegmentation network, we apply the learned superpoints to the LiDAR semantic segmentation task. Specifically, we propose a simple multi-scale superpoint aggregation module and integrate it with our LiDAR oversegmentation network to form an end-to-end LiDAR semantic segmentation framework, as shown in Fig. 2.

Multi-Scale Superpoint Aggregation In LiDAR oversegmentation network, sparse convolution [19] is used to extract the local feature of points. However, voxel based representation cannot elaborately capture the local geometric structures of the point cloud, resulting in a rough feature map $\boldsymbol{X} \in \mathbb{R}^{N\times C}$. Based on the rough point features, we design a simple multi-scale superpoint aggregation module to enhance point features by fusing multi-scale local features. Specifically, we first obtain multi-scale superpoints by adjusting the minimum and maximum sizes of superpoint during superpoint generation. For i-th point, we can obtain the corresponding superpoints of L scales, denoted by $S_1^i, S_2^i, \ldots, S_L^i$. Then, we employ the max pooling function on the point features within the corresponding superpoint to aggregate superpoint features (three circular cones in Fig. 2). After that, the superpoint features are processed independently by a MLP comprised of a series of layers, including linear, ReLU [20], and batch normalization [12]. The resulting superpoint features are denoted by $\boldsymbol{E}_1^i, \boldsymbol{E}_2^i, \ldots, \boldsymbol{E}_L^i$. Finally, we fuse superpoint features of L scales to aggregate multi-scale geometric information of the point cloud. The resulting new feature of the i-th point is written as:

$$\hat{\boldsymbol{X}}_i = \mathrm{Concat}(\boldsymbol{X}_i; \boldsymbol{E}_1^i; \boldsymbol{E}_2^i; \ldots; \boldsymbol{E}_L^i) \tag{6}$$

Compared with coarse point feature $\boldsymbol{X}_i \in \mathbb{R}^C$ learned from voxel based representation, the new point feature $\hat{\boldsymbol{X}}_i \in \mathbb{R}^{C*(L+1)}$ can elaborately describe the local geometric structures of the point through multi-scale superpoints. Based on the fused point features, we employ a classification head to predict point labels.

Essentially, the superpoint provides an adaptive neighborhood, which is generated along the geometric structure of the point cloud surface. Therefore, compared with voxel based, ball query based, and k-NN based neighborhoods, adaptive neighborhoods can extract more discriminative local features, thereby improving the performance.

Loss Functions Since the proposed LiDAR semantic segmentation framework is an end-to-end network, it can be directly optimized through a joint loss function. To train it, the joint loss function is defined as:

$$\mathcal{L}_{joint} = \mathcal{L}_{sp} + \mathcal{L}_{sr} + \mathcal{L}_{sem} \tag{7}$$

where $\mathcal{L}_{sp}$ is the local discriminative loss for training superpoints defined in Eq. (1). $\mathcal{L}_{sr}$ and $\mathcal{L}_{sem}$ are the cross-entropy losses for superpoint refinement and semantic segmentation, respectively.

4. Experiments

4.1. Datasets and Metrics

In this paper, for LiDAR oversegmentation and semantic segmentation, we use two large-scale outdoor datasets, *i.e.*, SemanticKITTI [1] and nuScenes [3]. The details of the datasets and evaluation metrics are as follows.

SemanticKITTI is a large-scale outdoor benchmark for self-driving cars with 20 semantic categories. It contains 22 sequences collected by the 64 beams LiDAR sensor, where sequences 00 to 10 are for the training set (sequence 08 is used as the validation set.), and sequences 11 to 22 are for the online hidden test set. Sequences 00 to 10 are provided with dense semantic annotations for each scan.

nuScenes contains 1000 scenes collected by the 32 beams LiDAR sensor. A total of 1000 scenes are split into training (750), validation (150), and testing (150) sets. The training and validation sets are annotated with 17 semantic categories, while the test set whose labels are held for blind online testing.

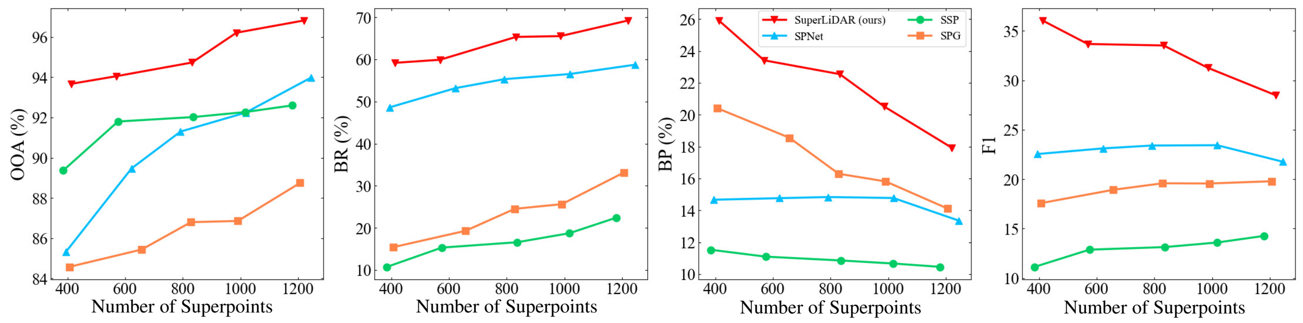

Figure 3. Performance of different methods on the validation set of SemanticKITTI.

Evaluation Metrics To evaluate the quality of superpoints, we use oracle overall accuracy (OOA), boundary recall (BR), and boundary precision (BP) as the evaluation metrics defined in [13]. OOA measures the theoretical upper bound of semantic segmentation using superpoints, while BR and BP measure the quality of superpoint boundaries. In order to balance BR and BP, we report the F1 score, which is formulated as F1=2BP·BR/(BR+BP). To evaluate the performance of LiDAR semantic segmentation, we use the mean intersection over union (mIoU).

4.2. Network Structure

We apply the same sparse 3D network used in [37] as the backbone of our LiDAR oversegmentation network. In the LiDAR point grouping algorithm, the threshold γ of distance constraint in BFS is set to 1.5. In the superpoint refinement module, the hyperparameter K is set to 5. In the multi-scale superpoint aggregation module, we use three scales of superpoints, which are controlled by different thresholds of superpoint size.

4.3. LiDAR Oversegmentation

For LiDAR oversegmentation, we compare the proposed SuperLiDAR with SPG [15], SSP [13], and SPNet [11]. Note that SPG uses handcrafted features, whereas SSP and SPNet use deep features for superpoint segmentation. We run the official codes to produce superpoints from the LiDAR point clouds. The superpoint evaluation results are computed on the validation set of the SemanticKITTI and nuScenes datasets, respectively.

Results on SemanticKITTI Tab. 1 reports the quantitative results of superpoint segmentation on the SemanticKITTI dataset. For a fair comparison, the number of superpoints generated by different methods is maintained similarly (about 1000 superpoints). It can be observed that our method achieves the best results on four metrics. Especially, our SuperLiDAR significantly outperforms other methods by about 4% in terms of OOA, which represents the theoretical upper bound of performance of semantic segmentation. Therefore, it means that our method can generate high-quality superpoints with higher accuracy. In addition, our method has higher BR and BP, which indicate that our method can generate superpoint with clear boundaries in sparse LiDAR point clouds. Due to sparse LiDAR point clouds, the handcrafted features of SPG [15] are less discriminative, resulting in worse performance. Although SSP [13] uses deep features, it generates superpoints through an optimization based method [14], which limits the performance on sparse LiDAR point clouds. In sparse point clouds, the superpoints generated by the clustering based method SPNet [11] are likely to contain noise, which affects the performance. In Fig. 3, we show the performance curves of different methods under different numbers of superpoints. It can be observed that the curves of our method are higher than others in terms of four metrics. The more the number of superpoints, the smaller the size of superpoints, and vice versa. Thus, it shows that our method can generate high-quality superpoints of different sizes.

Method	OOA	BR	BP	F1
SemanticKITTI				
SPG [15]	86.86	25.64	15.82	19.56
SSP [13]	92.27	18.74	10.67	13.59
SPNet [11]	92.24	56.52	14.78	23.43
SuperLiDAR (ours)	**96.21**	**65.52**	**20.52**	**31.25**
nuScenes				
SPG [15]	89.22	28.20	17.26	21.41
SSP [13]	92.01	22.04	15.63	18.28
SPNet [11]	87.67	68.92	18.34	28.97
SuperLiDAR (ours)	**96.31**	**74.72**	**25.12**	**37.59**

Table 1. Comparison results of generated superpoints on the validation sets of SemanticKITTI and nuScenes.

Results on nuScenes The quantitative results of superpoint segmentation on the nuScenes dataset is shown in Tab. 1. Note that the number of superpoints generated by different methods is maintained similarly (about 400 superpoints). Compared with the 64-beam point cloud of SemanticKITT, the 32-beam point cloud of nuScenes is more sparse. From the table, it can be observed that our method still achieves the best results on all four metrics. The results on the nuScenes dataset further demonstrate that our method can effectively generate high-quality superpoints in sparse LiDAR point clouds.

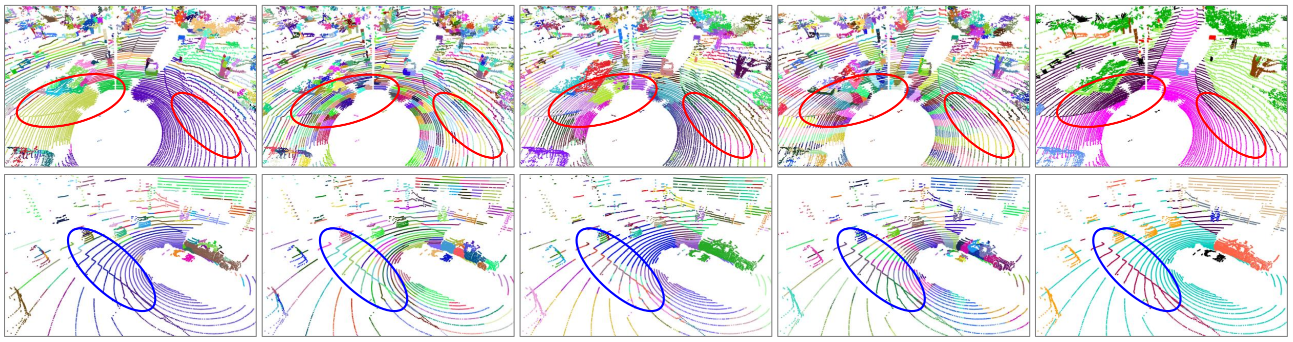

Figure 4. Visualization of generated superpoints on the validation sets of SemanticKITTI (first row) and nuScenes (second row).

Method	Type	Inference Time (ms) data proc. + superpoint gen.
SPG [15]	CPU	13152 (0+13152)
SSP [13]	GPU	25238 (11812 + 13426)
SPNet [11]	GPU	12966 (11812 + 1154)
SuperLiDAR (ours)	GPU	**72** (0 + 72)

Table 2. Inference time of different methods on the validation set of SemanticKITTI. Note that inference time consists of data processing and superpoint generation.

Visual Results In Fig. 4, we show the visualization results of different oversegmentation methods. It can be observed that the generated superpoints by our SuperLiDAR can effectively distinguish the road and sidewalk. Note that the road and sidewalk are almost in the same horizontal plane, so it is hard to discriminate them without color information. Nonetheless, our method is able to learn discriminative point features to separate them by using the local discriminative loss.

Time Costs The inference time is an important criterion for superpoint segmentation. For optimization based method SPG [15], we use a single Core i5 CPU to compute inference time. For learning based methods SSP [13], SPNet [11], and our SuperLiDAR, we use a single NVIDIA RTX 3090 to run the code based on the PyTorch deep learning platform. Tab. 2 reports the average inference time of different methods computed on each scan of the SemanticKITTI validation dataset. It can be observed that the inference time of our method is only 72ms, which is 100$\times$ faster than others. Although SSP and SPNet use the network to extract point features, they also feed the handcrafted point features into the network, resulting in a long data processing time. Note that the data processing of SSP and SPNet are the same, so their data processing time is the same. Since SPG and SSP adopt the same optimization based method to generate superpoints, they have a longer time for superpoint generation. Due to the iterative strategy in the superpoint generation of SPNet, it consumes more time than our method.

Ablation Studies In low-level point embedding learning, we construct the local neighborhoods in the range image instead of in 3D space and experiment on the SemanticKITTI dataset. In Tab. 3, we show the results of building neighborhoods in 3D space. It can be observed that the performance of building neighborhoods in 3D space ("build nei. in 3D space") is worse than in the range image ("Default (SuperLiDAR)"). Compared with 3D space, the points are more uniform in the range image. The resulting dense neighborhoods are profitable to learn good point embeddings by applying the local discriminative loss.

Setting	OOA	BR	BP	F1
build nei. in 3D space	92.78	45.69	15.06	22.65
w/o superpoint ref.	94.81	57.72	17.98	27.41
Default (SuperLiDAR)	**96.21**	**65.52**	**20.52**	**31.25**

Table 3. Ablation study results of different settings on the validation set of SemanticKITTI.

In order to tackle unassigned points after performing LiDAR point grouping, we propose a superpoint refinement module. To verify its effectiveness, we replace the superpoint refinement module by assigning the points to the nearest superpoint center in the coordinate space. In Tab. 3, the results ("w/o superpoint ref.") are lower than that of using the superpoint refinement module ("Default (SuperLiDAR)"). Due to the simple distance criterion, hard boundary points cannot be effectively assigned to the corresponding superpoints. By adaptively learning the affinity between the point and superpoint, we can more accurately assign the points to the corresponding superpoints.

4.4. LiDAR Semantic Segmentation

For LiDAR semantic segmentation, we experiment on the SemanticKITTI and nuScenes datasets to verify the effectiveness of the learned superpoints. Our LiDAR semantic segmentation framework is based on LiDAR oversegmentation network, and uses a multi-scale superpoint aggregation module for semantic prediction. We use the backbone (*i.e.*, sparse 3D network) of LiDAR oversegmentation network as our baseline. Since our baseline is the sparse 3D network used in [37], we directly take the

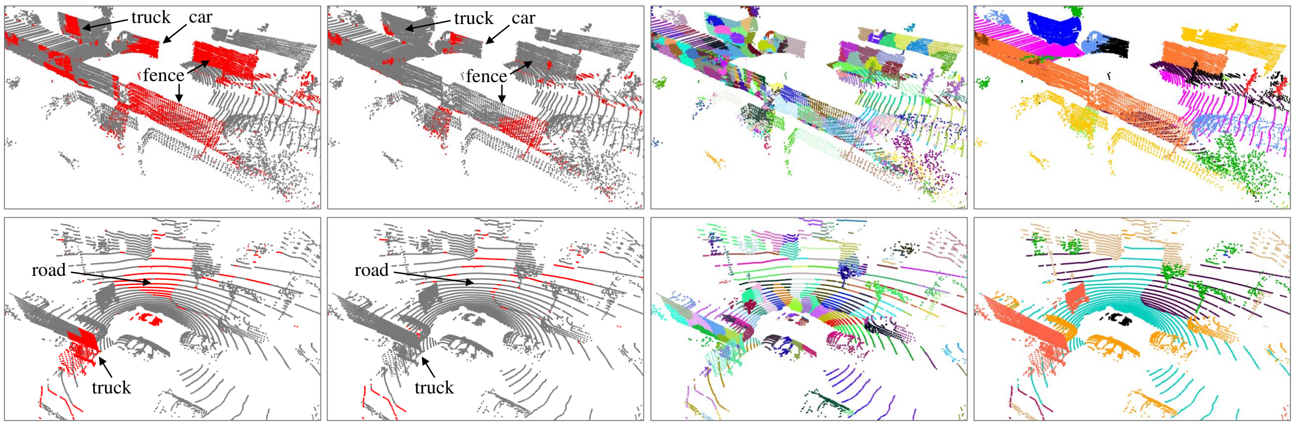

Figure 5. Visualization of semantic segmentation on the validation sets of SemanticKITTI (first row) and nuScenes (second row).

Method	Data	SemanticKITTI		nuScenes	
		mIoU	Speed (ms)	mIoU	Speed (ms)
PolarNet [39]	L	54.3	**62**	69.4	-
JS3C-Net [36]	L	66.0	471	73.6	-
Cylinder3D [43]	L	68.9	131	77.2	63
SPVNAS [25]	L	67.0	259	77.4	63
AF^2-S3Net [5]	L	70.8	-	78.3	270
RPVNet [35]	L	70.3	168	-	-
PMF [44]	L+C	-	-	77.0	125
2D3DNet [7]	L+C	-	-	80.0	-
2DPASS [37]	L+C	**72.9**	**62**	**80.8**	**44**
Baseline	L	67.4	**62**	77.6	**44**
SuperLiDAR (ours)	L	69.6	78	78.5	60

Table 4. Semantic segmentation results on the test sets of SemanticKITTI and nuScenes. The results are compared before 11/11/2022. "L" and "C" indicate the LiDAR and camera, respectively. Note that we only list the results of published works.

baseline results of [37] as the baseline results of our method on the SemanticKITTI and nuScenes datasets.

Results on SemanticKITTI Tab. 4 shows the results of semantic segmentation on the SemanticKITTI online test set. It can be observed that the mIoU of our SuperLiDAR is 2% higher than the baseline. In addition, Fig. 5 shows the visualization results (first row) of the baseline and our method. The results can demonstrate that the performance of semantic segmentation can be further improved by using the learned superpoints to enhance the local geometric information of the point clouds. Due to adopting complex point-voxel or range-point-voxel fusion framework, the performance of RPVNet [35] and AF^2-S3Net [5] is higher than our voxel based method. In addition, 2DPASS [37] is a multi-modal method that uses 2D image and knowledge distillation to achieve higher performance and shorter inference time. Due to the simple yet effective multi-scale superpoint aggregation module, our method can effectively improve the performance of the baseline without increasing too much time cost.

Results on nuScenes Tab. 4 shows the results of semantic segmentation on the nuScenes online test set. It can be observed that the mIoU of our SuperLiDAR exceeds the baseline by about 1%. In addition, Fig. 5 shows the visualization results (second row) of the baseline and our method. Since the point clouds of nuScenes are more sparse than those in SemanticKITTI, the quantitative and visualization results demonstrate that the learned superpoint can effectively improve the segmentation performance. Compared with methods only using LiDAR data, our method can achieve higher performance with a shorter inference time. Note that even if we only use LiDAR data, the performance of our method is better than that of PMF [44] using LiDAR point clouds and camera images. In addition, multi-modal methods 2D3DNet [7] and 2DPASS [37] adopt complicated multi-scale fusion networks to achieve higher performance.

5. Conclusion

We presented an efficient end-to-end LiDAR oversegmentation network for segmenting superpoints from the LiDAR point clouds. By utilizing the proposed LiDAR point grouping algorithm, our method can generate high-quality superpoints from sparse LiDAR point clouds. Notably, our method achieves new state-of-the-art in LiDAR oversegmentation, and the inference time is 100× faster than current methods. Furthermore, on LiDAR semantic segmentation, our method can significantly boost the baseline results on two large-scale benchmarks (*i.e.*, SemanticKITTI and nuScenes) by using superpoints. We believe that the proposed efficient LiDAR oversegmentation network can be applied to more downstream tasks, such as 3D detection and tracking.

Acknowledgments This work was supported by the National Science Fund of China (Grant Nos. U1713208, U62276144, 62271410) and the Fundamental Research Funds for the Central Universities.

References

[1] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. SemanticKITTI: A dataset for semantic scene understanding of lidar sequences. In *ICCV*, 2019. 5

[2] Alexandre Boulch, Bertrand Le Saux, and Nicolas Audebert. Unstructured point cloud semantic labeling using deep segmentation networks. *3DOR*, 2017. 2

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 5

[4] Mingmei Cheng, Le Hui, Jin Xie, Jian Yang, and Hui Kong. Cascaded non-local neural network for point cloud semantic segmentation. In *IROS*, 2020. 3

[5] Ran Cheng, Ryan Razani, Ehsan Taghavi, Enxu Li, and Bingbing Liu. AF^2-S3Net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. In *CVPR*, 2021. 3, 8

[6] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 2

[7] Kyle Genova, Xiaoqi Yin, Abhijit Kundu, Caroline Pantofaru, Forrester Cole, Avneesh Sud, Brian Brewington, Brian Shucker, and Thomas Funkhouser. Learning 3D semantic segmentation with only 2D image supervision. In *3DV*, 2021. 8

[8] Stéphane Guinard and Loic Landrieu. Weakly supervised segmentation-aided classification of urban scenes from 3D lidar point clouds. *ISPRS Workshop*, 2017. 1, 2

[9] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. RandLA-Net: Efficient semantic segmentation of large-scale point clouds. *arXiv preprint arXiv:1911.11236*, 2019. 2

[10] Le Hui, Linghua Tang, Yaqi Shen, Jin Xie, and Jian Yang. Learning superpoint graph cut for 3D instance segmentation. In *NeurIPS*, 2022. 2

[11] Le Hui, Jia Yuan, Mingmei Cheng, Jin Xie, and Jian Yang. Superpoint network for point cloud oversegmentation. In *ICCV*, 2021. 1, 2, 6, 7

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5

[13] Loic Landrieu and Mohamed Boussaha. Point cloud oversegmentation with graph-structured deep metric learning. In *CVPR*, 2019. 1, 2, 6, 7

[14] Loic Landrieu and Guillaume Obozinski. Cut Pursuit: fast algorithms to learn piecewise constant functions. In *AISTATS*, 2016. 1, 2, 6

[15] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, 2018. 6, 7

[16] Felix Järemo Lawin, Martin Danelljan, Patrik Tosteberg, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Deep projective 3D semantic segmentation. In *CAIP*, 2017. 2

[17] Yangbin Lin, Cheng Wang, Dawei Zhai, Wei Li, and Jonathan Li. Toward better boundary preserved supervoxel segmentation for 3D point clouds. *ISPRS*, 2018. 1, 2

[18] Venice Erin Liong, Thi Ngoc Tho Nguyen, Sergi Widjaja, Dhananjai Sharma, and Zhuang Jie Chong. AMVNet: Assertion-based multi-view fusion network for LiDAR semantic segmentation. *arXiv preprint arXiv:2012.04934*, 2020. 3

[19] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *CVPR*, 2015. 3, 5

[20] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 5

[21] Jeremie Papon, Alexey Abramov, Markus Schoeler, and Florentin Worgotter. Voxel cloud connectivity segmentation-supervoxels for point clouds. In *CVPR*, 2013. 1, 2

[22] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017. 3

[23] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 3

[24] Damien Robert, Bruno Vallet, and Loic Landrieu. Learning multi-view aggregation in the wild for large-scale 3d semantic segmentation. In *CVPR*, 2022. 2

[25] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3D architectures with sparse point-voxel convolution. In *ECCV*, 2020. 3, 8

[26] Linghua Tang, Le Hui, and Jin Xie. Learning inter-superpoint affinity for weakly supervised 3D instance segmentation. In *ACCV*, 2022. 2

[27] Liyao Tang, Yibing Zhan, Zhe Chen, Baosheng Yu, and Dacheng Tao. Contrastive boundary learning for point cloud segmentation. In *CVPR*, 2022. 2

[28] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3D. In *CVPR*, 2018. 3

[29] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPConv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019. 3

[30] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *CVPR*, 2019. 2

[31] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph CNN for learning on point clouds. *TOG*, 2019. 3

[32] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. SqueezeSeg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3D LiDAR point cloud. In *ICRA*, 2018. 3

[33] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. SqueezeSegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a LiDAR point cloud. In *ICRA*, 2019. 3

[34] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point Transformer V2: Grouped vector attention and partition-based pooling. In *NeurIPS*, 2022. 2

[35] Jianyun Xu, Ruixiang Zhang, Jian Dou, Yushi Zhu, Jie Sun, and Shiliang Pu. RPVNet: A deep and efficient range-point-voxel fusion network for LiDAR point cloud segmentation. In *ICCV*, 2021. 3, 8

[36] Xu Yan, Jiantao Gao, Jie Li, Ruimao Zhang, Zhen Li, Rui Huang, and Shuguang Cui. Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion. In *AAAI*, 2021. 8

[37] Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui, and Zhen Li. 2DPASS: 2D priors assisted semantic segmentation on LiDAR point clouds. In *ECCV*, 2022. 3, 6, 7, 8

[38] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. PointASNL: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *CVPR*, 2020. 3

[39] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. PolarNet: An improved grid representation for online LiDAR point clouds semantic segmentation. In *CVPR*, 2020. 8

[40] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Shell-Net: Efficient point cloud convolutional neural networks using concentric shells statistics. In *ICCV*, 2019. 2

[41] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. PointWeb: Enhancing local neighborhood features for point cloud processing. In *CVPR*, 2019. 3

[42] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point Transformer. In *ICCV*, 2021. 2

[43] Hui Zhou, Xinge Zhu, Xiao Song, Yuexin Ma, Zhe Wang, Hongsheng Li, and Dahua Lin. Cylinder3D: An effective 3D framework for driving-scene LiDAR semantic segmentation. *arXiv preprint arXiv:2008.01550*, 2020. 3, 8

[44] Zhuangwei Zhuang, Rong Li, Kui Jia, Qicheng Wang, Yuanqing Li, and Mingkui Tan. Perception-aware multi-sensor fusion for 3d lidar semantic segmentation. In *ICCV*, 2021. 8