A Cylindrical Convolution Network for Dense Top-View Semantic Segmentation with LiDAR Point Clouds

Anonymous ACCV 2022 Submission

Paper ID 1011

Abstract. Accurate semantic scene understanding of the surrounding environment is a challenge for autonomous driving systems. Recent LiDAR-based semantic segmentation methods mainly focus on predicting point-wise semantic classes, which cannot be directly used before the further densification process. In this paper, we propose a cylindrical convolution network for dense semantic understanding in the top-view LiDAR data representation. 3D LiDAR point clouds are divided into cylindrical partitions before feeding to the network, where semantic segmentation is conducted in the cylindrical representation. Then a cylinder-to-BEV transformation module is introduced to obtain sparse semantic feature maps in the top view. In the end, we propose a modified encoder-decoder network to get the dense semantic estimations. Experimental results on the SemanticKITTI and nuScenes-LidarSeg datasets show that our method outperforms the state-of-the-art methods with a large margin.

1 Introduction

Semantic perception of the surrounding environments is important for autonomous driving systems. In order to achieve reliable semantic estimations in top-view representation, autonomous vehicles are usually equipped with camera and LiDAR sensors. Benefiting from the rapid development of convolutional neural networks (CNNs), a large number of camera-based semantic segmentation networks, like Fully Convolutional Network (FCN) [1], ERFNet [2], U-Net [3], etc., have been proposed and proved to be effective. However, most of these methods are only applicable to the segmentation in perspective view, and accurate transformation from perspective to top-view is still a challenge. The camera sensor lacks effective geometric perception of the environment. In recent years, with the release of large-scale 3D LiDAR semantic segmentation datasets (SemanticKITTI [4] and nuScenes-LidarSeg [5]), the LiDAR-based semantic segmentation performance has been significantly increased. Because these datasets only provide point level semantic labels, most works only perform sparse semantic segmentation and predict point-wise semantic classes. The obtained sparse results still need further processing before used. Therefore, some works conduct dense top-view semantic segmentation with sparse inputs. Compared with sparse predictions, the dense top-view segmentation results are more valuable for some upper-level tasks such as the navigation and path planning of autonomous driving vehicles.

In this paper, we focus on the dense semantic segmentation of LiDAR point clouds in top-view representation. Compared with 2D camera images, the 3D LiDAR data can retain precise and complete spatial geometric information of the surroundings. Therefore, we can generate accurate top-view maps in a simpler way. However, the main problem is that the LiDAR data is sparsely distributed, and the generated top-view maps are sparse. In order to deal with the sparsity of 3D LiDAR data, some LiDAR-based segmentation approaches [6] project the 3D point clouds onto 2D bird's-eye-view (BEV[1]) images, and conduct dense semantic segmentation with 2D convolution networks. However, the projection process inevitably leads to a certain degree of information loss. Some methods [7–9] use pillar-level representation and point-wise convolution to retain and obtain more information in the height direction. These approaches still focus more on 2D convolution, neglecting the rich geometric relationshipss between precise 3D point cloud data.

To solve the problems mentioned above, we make use of the cylinder representation and 3D sparse convolution networks in our work. Compared with 2D images or pillar representation, the cylinder representation can maintain the 3D geometric information. The cylindrical partition divides the LiDAR point cloud dynamically according to the distances in cylindrical coordinates, and provides a more balanced distribution than 3D voxelization. The 3D sparse convolution networks can effectively integrate the geometric relationships of LiDAR point clouds, extract informative 3D features and save significant memory at the same time.

After the 3D sparse convolution networks, we introduce a cylinder-to-BEV module to convert the obtained semantic features in cylindrical representation to BEV maps. The cylinder-to-BEV module uses the coordinate information of 3D points to establish corresponding relationships, and transfers features between the two representations. The transformed feature maps in top-view are sparse, so we further propose a modified U-Net network to get the final dense segmentation results. We use groups of dilated convolutions with different receptive field sizes in different stages of downsampling and upsampling to capture more descriptive spatial features, and use grouped convolutions to reduce the FLOPs while maintaining an acceptable level of accuracy.

The main contributions of this work lie in three aspects:

- We propose an end-to-end cylindrical convolution network that can generate accurate semantic segmentation results in top-view. The combination of cylinder representation and 3D sparse convolution greatly improves the segmentation performance.
- We propose a cylinder-to-BEV module and a modified U-Net to efficiently use 3D features to enhance the dense semantic segmentation in top-view.

[1] BEV is another expression for top view.

ACCV2022 #1011

- The proposed method outperforms the state-of-the-art methods on the SemanticKITTI and nuScenes-LidarSeg datasets, which demonstrates the effectiveness of the model.

2 Related Work

2.1 Image Semantic Segmentation in Bird's Eye View

Understanding the surrounding environment is an essential part of an autonomous driving system. To accomplish this, many previous works created a semantic map in Bird's Eye View(BEV) that can distinguish drivable regions, sidewalks, cars, bicycles and so on [10–14]. Image semantic segmentation in BEV usually consists of following components: an encoder for encoding features in the image view, a view transformer for converting the features from the image view to BEV, an encoder for further encoding the features in BEV and a semantic head for label classification [10–12]. Thomas Roddick *et al.* [13] chose feature pyramid networks like in [15] when extracting the image-view features. Weixiang Yang *et al.* [16] implemented cross-view transformation module that consists of the cycled view projection and the cycled view transformer in order to enrich the features getting from front-view image.

2.2 3D Data Representation

Since the recent launch of large-scale datasets, such as SemanticKITTI [4] and nuScenes [5], there have been an increasing number of studies on semantic segmentation of LiDAR point clouds. However, due to the sparse, irregular, and disorderly LiDAR point cloud data, it is still challenging to be processed and applied to semantic segmentation. The methods are mostly divided into three branches: point-based methods, grid-based methods, and projection-based methods.

Point-based methods directly process raw LiDAR point clouds. PointNet [17] is a pioneer and a representative of point-based approaches, which learns features for each point using shared MLPs. PointNet++ [18], an upgrade to PointNet, generates point cloud subsets by clustering and employs PointNet to extract point features from each subset. RandLA-Net [19] uses a random sampling to boost processing speed and local spatial encoding and attention-based pooling. KPConv [20] develops spatial kernels to adapt convolution operations to point clouds. However, due to computational complexity and memory requirements, the performance on large-scale LiDAR point cloud datasets is limited.

Grid-based methods convert point clouds into uniform voxels, after which, 3D convolution can be employed on voxel data. In VoxelNet [21], point clouds are quantized into uniform 3D volumetric grids, which maintains the 3D geometric information. Fully convolutional point networks [22] achieve uniform sampling in 3D space by collecting a fixed number of points around each sampled site and then apply a U-Net to extract information from multiple scales. Cyliner3D [23]

ACCV2022 #1011

recommends dividing the original point cloud into cylindrical grids to distribute the point clouds more evenly throughout the grids.

Furthermore, projection-based methods project 3D point clouds onto different 2D images. SqueezeSeg series [24, 25], RangeNet series [26] and Salsanet series [6,27] deploy the spherical projection on the LiDAR data. In VoloMap [28] and PolarNet [29], bird-eye view (BEV) and polar BEV projections on LiDAR points are proposed respectively. What's more, PointPillar series [7, 30, 31] split the point clouds into a group of pillars and project them onto a pseudo image. Projection-based methods can attain real-time performance, but their accuracy is often limited due to the information loss of projection.

2.3 Semantic Segmentation of LiDAR Input

Many approaches take point clouds as input and return point-wise semantic labels as output, such as PointNet [17], PointNet++ [18] and KPConv [20]. Other methods convert the input of point clouds into different forms and finally produce semantic voxel grids or semantic projected images. Voxelnet [21] and SegCloud [32] employ 3D fully convolutional networks on voxels to assign a class label to each grid, which is shared by all points in the grid. RangeNet++ [26], a typical method based on Range-View, converts point cloud to a range image and performs semantic segmentation of the image using an hourglass fully convolution network.

What's more, some approaches achieve top-view semantic segmentation after projecting the point cloud into a BEV image [8,33–35]. Bieder et. al [33] turn 3D LiDAR data into a multi-layer top-view map for accurate semantic segmentation. Following this route, PillarSegNet [8] is proposed to learn features from the pillar encoding and conduct 2D dense semantic segmentation in the top view. These methods take LiDAR point cloud as input and generates dense top-view semantic grid maps, which provide a fine-grained semantic understanding that is necessary for distinguishing drivable and non-drivable areas.

3 Proposed Method

In this section, we introduce the architecture of the proposed dense top-view semantic segmentation method, as illustrated in Fig. 1. The whole network consists of three parts, a 3D cylindrical encoding, a cylinder-to-BEV module, and a 2D encoder-decoder network. The network takes sparse a LiDAR point cloud as input and generates dense semantic maps in top-view. The design of each module will be detailed in the following.

3.1 3D Cylindrical Encoding

Effectively extracting the features of 3D point cloud data is an important part of the LiDAR-based semantic segmentation. Previous methods usually convert the

ACCV2022 #1011

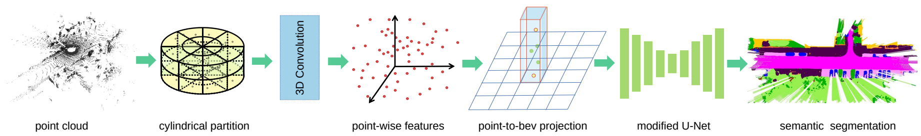


Fig. 1. Overview of the proposed dense top-view semantic segmentation network. Given a 3D LiDAR point cloud, the network first divides it into cylindrical partitions and applies a 3D sparse convolution module to obtain high-level features. Then, the cylinder-to-BEV module converts the semantic features in cylindrical representation to BEV maps. Finally, a modified U-Net is used to predict the dense top-view semantic segmentation results.

3D LiDAR point cloud into voxels [21] or pillar features [8]. The voxel representation quantizes the point cloud into uniform cubic voxels. However, the LiDAR data is irregular and unstructured. This leads to a large number of empty voxels and affects the computational efficiency. The pillar-based method uses MLPs to extract features, which cannot make full use of the 3D topology and rich spatial geometric relationships.

Based on these considerations, we apply cylindrical coordinates to represent LiDAR data in this paper. The density of the 3D LiDAR point cloud usually varies, and the density in nearby areas is significantly higher than that in remote areas. Uniformly dividing the LiDAR data with different densities will lead to an unbalanced distribution of points. The cylindrical partitions can cover areas with different size of grids, which grow by distance, evenly distribute points on different cylindrical grids, and provide a more balanced representation. The cylindrical coordinate system is defined as follows,

$$\begin{cases} \rho = \sqrt{x^2 + y^2} \\ \theta = arctan(y, x) \\ z = z \end{cases} \tag{1}$$

where (x, y, z) represents the Cartesian coordinate, and (ρ, θ, z) represents its corresponding cylindrical coordinate. The radius ρ and tangent angle θ denote the distance from the origin in the x-y plane and the tangent angle between the y and x directions, respectively.

Compared with the voxel representation, although the number of empty elements is reduced, the cylindrical representation of LiDAR data is still sparse. Therefore, we apply a 3D sparse convolution network to extract features, which can efficiently process sparse data and increase the computing speed. More details of the network can be referred in [23].

3.2 Cylinder-to-BEV Module

After the 3D feature encoding module, we can obtain high-level features with rich semantic information in the form of cylindrical representation. Since the goal is to predict dense semantic categories in top-view, we need to convert the cylindrical features into BEV maps before the 2D semantic segmentation module.

Figure 2 shows two types of transformations, without and with point guidance. Without point guidance means that we use the correspondence between the cylindrical coordinate system and the BEV coordinate system to directly convert the features. However, the cylindrical grid is different from the BEV grid, which can lead to deviation problems at the boundaries. As shown in the left of Fig.2, the cylindrical grid and the BEV grid have different shapes, in which the yellow denotes the cylindrical grid with features, and the purple represents the empty cylindrical grid without features. One BEV grid overlaps with two cylindrical grids. Transforming directly from cylinder to BEV may establish correspondence between the purple cylindrical grid and the BEV grid instead of the yellow one, resulting in loss of information. The right of Fig.2 shows the transformation with point guidance. The cylindrical features are first converted to point features according to Eq. 1. Then, the point features are converted to BEV features according to Eq. 2. Using point features as intermediate can preserve more useful features and cover the BEV grids more completely.

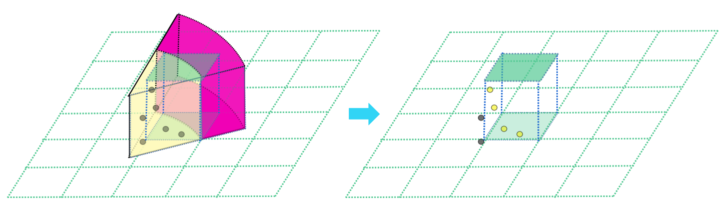


Fig. 2. Illustration of conversion from cylinder to BEV without and with point guidance. Left shows the transformation without point guidance, in which the yellow denotes cylindrical grid with features, and the purple represents empty cylindrical grid without features. Right shows the transformation with point guidance, the point serves as intermediate to connect the cylinder and BEV grids.

The transformation from point to BEV grid is described bellow. Given a point (x, y, z) and the feature f, its corresponding coordinates in BEV are calculated as,

$$\begin{cases} u = x/precision + W/2 \\ v = y/precision + H/2 \end{cases} \tag{2}$$

where (u, v) represents the corresponding point in BEV, $precision$ denotes the resolution of BEV. W and H represent the width and height of the BEV map, respectively.

When converting point features to BEV features, a lot of geometric information may be lost due to the many-to-one problem. Different from some methods

ACCV2022 #1011

that use maximum compression and retain only one point feature, we sort the points corresponding to the same BEV grid by height, and retain the features of the highest and lowest points. Therefore, the features of each BEV grid are as follows,

$$F_{bev} = (r_l, x_l, y_l, z_l, f_l, r_h, x_h, y_h, z_h, f_h) \tag{3}$$

where r denotes the distance and the subscripts l and h represent the lowest and highest points, respectively. The features (z_h, f_h, z_l, f_l) of the highest and lowest points can provide the height range and spatial features of each BEV grid. For example, the spatial features of the highest and lowest points for roads, vehicles, and pedestrians vary greatly, which is very useful in determining the semantic categories.

3.3 2D Semantic Segmentation

After the cylinder-to-BEV process, we can get sparse BEV maps with rich semantic information. In this section, we will introduce the 2D semantic segmentation that is used to densify the semantic predictions. The network is based on U-Net structure and added various convolution designs, including dilated convolution, depth-wise convolution, inverse bottleneck, etc., as shown in Fig.3.

Encoder-Decoder Architecture. Building upon the U-Net framework, we use convolutional blocks in both encoder and decoder, supplemented by appropriate design, to make the network more suitable for the LiDAR-based semantic segmentation task. As a characteristic of U-Net, skip-connection is also used to improve image segmentation accuracy by fusing low-level and high-level features. Considering the computational overhead, we use a separate downsampling layer and a pixel-shuffle layer instead of transpose convolution in the upsampling part.

Depth-wise Convolution and Inverse Bottleneck. Depth-wise convolution is adapted from grouped convolution, in which the number of groups equals the number of channels. The advantage is that it greatly reduces the floating-point operations while maintaining an acceptable level of accuracy. An important design of blocks in Transformer [36], MobileNetV2 [37] and ConvNet [38] is the inverse bottleneck. The dimension of the intermediate hidden layer is larger than that of the input and output layers. We implement depth-wise convolution and combine it with two 1×1 convolutions to form an inverse bottleneck. As shown in Fig. 3, we combine these two designs and apply them as the basic block whose color is pink.

Dilated Convolution Groups. Unlike increasing the size of the convolution kernel, which greatly increases the number of parameters, dilated convolution provides a cost-effective way to extract more descriptive features. Because different stages of U-Net have different scales of information, we use convolution groups with different receptive field sizes for each stage in downsampling and upsampling as shown in Fig. 3. In each group, a 1× 1 convolution is used to extract the spatial information from different receptive fields after concatenating the outputs of each dilated convolution. Meanwhile, a dropout layer and a pooling layer are added at the end. Following Transformer [36], the number of

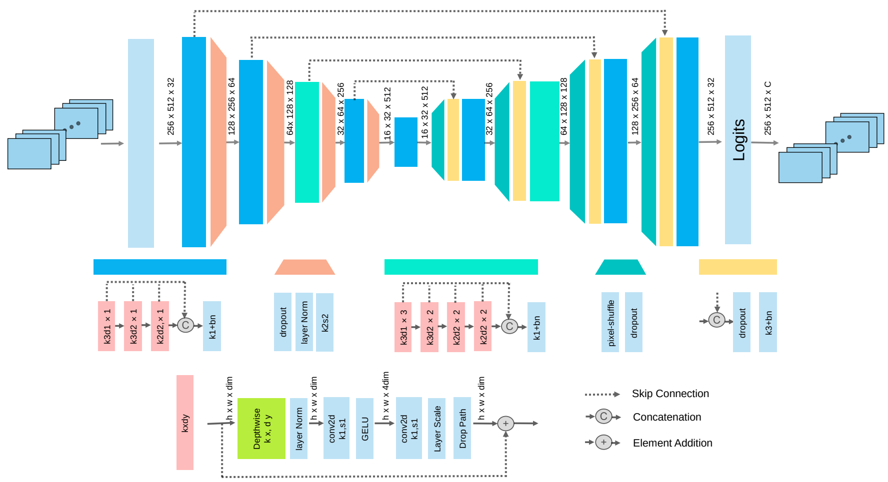


Fig. 3. Flowchart of the proposed modified U-Net. k, d, s, bn, and $\times$ represent the kernel size, dilation rate, stride, batch normalization, and block numbers, respectively. Blocks of different colors represent convolutional layers of different structures. Among them, the light blue blocks represent ordinary convolution, dropout, and normalization blocks. The pink blocks represent a designed convolution block, which is the base block of each convolution layer.

convolution blocks in different stages of downsampling and upsampling is adjusted to (3,3,9,3). As seen in the Fig. 3, the blue block represents a convolution group consisting of 3 convolution blocks, and the cyan block represents a convolution group consisting of 9 convolution blocks.

In addition, we use GeLU activation function instead of ReLU in the basic block, and reduce the number of activation functions used in each block. Similarly, we use fewer normalization layers and replace BatchNorm with LayerNorm, as in Transformer.

3.4 Loss Function.

The unbalanced data distribution in the dataset can make model training difficult. Especially for the classes with fewer samples, the network predicts them with a lower frequency than that of the classes with more samples. To solve this problem, we use the weighted cross-entropy loss function, whose weight is equal to the inverse square root of the frequency of each class, as shown below:

$$L_{wce}(y, \hat{y}) = -\sum_{i=1}^{n} \lambda_i p(y_i) log(p(\hat{y_i})) \tag{4}$$

ACCV2022 #1011

where n denotes the number of classes, y_i and $\hat{y}_i$ represent the ground truth and the prediction, respectively. $\lambda_i = 1/\sqrt{f_i}$ and f_i denotes the frequency of the i^{th} class.

In addition, we also incorporate the Lovász-Softmax loss in the training process. The Jaccard loss function is directly defined based on the Intersection over Union (IoU) metric. However, it is discrete and its gradient cannot be calculated directly. In [39], the lovász extension is proposed, which is derivable and can be used as the loss function to guide the training process. Specifically, the Lovász-Softmax loss can be expressed as follows:

$$\begin{aligned} L_{ls} &= \frac{1}{|C|}\sum_{c\in C}\overline{\Delta_{J_c}}(m(c)), \\ m_i(c) &= \begin{cases} 1 - x_i(c) & if \quad c = y_i(c), \\ x_i(c) & otherwise \end{cases} \end{aligned} \tag{5}$$

where C denotes the class number, $\overline{\Delta_{J_c}}$ represents the lovász extension of the Jaccard index. $x_i(c)$ and $y_i(c)$ represent the predicted probability and the ground truth of pixel i for class c, respectively.

The final loss function is a linear combination of the weighted cross-entropy loss and the Lovász-Softmax loss, as shown below:

$$L = L_{wce} + L_{ls} \tag{6}$$

4 Experiments

In order to evaluate the segmentation performance of the proposed network, we carry out experiments on SemanticKITTI [4] and nuScenes-LidarSeg [5] datasets with raw LiDAR data, sparse semantic segmentation ground truths, and the aggregated dense semantic segmentation ground truths. The experimental results show that our network achieves state-of-the-art performance in both SemanticKITTI and nuScenes-LidarSeg datasets.

4.1 Datasets

SemanticKITTI. The SemanticKITTI is a large-scale outdoor point cloud dataset with precise pose information and semantic annotations of each LiDAR point. The training set consists of sequences 00-07 and 09-10, and the evaluation set consists of sequence 08, containing 19130 and 4071 LiDAR scans, respectively. As in [8], we merge the 19 classes into 12 classes. Specifically, The *motorcyclist* and *bicyclist* are merged to *rider*. The *bicycle* and *motorcycle* are merged to *two-wheel*. The *car*, *truck* and *other-vehicle* are merged to *vehicle*. The *traffic-sign*, *pole* and *fence* are merged to *object*. The *other-ground* and *parking* are merged to *other-ground*. The *unlabeled* pixels are not considered in the training process.

ACCV2022 #1011

nuScenes-LidarSeg. The nuScenes-LidarSeg provides semantic annotations for each LiDAR point in the 40,000 keyframes, marking a total of 1.4 billion LiDAR points, including 32 classes. Similarly, we map the *adult*, *child*, $policeofficer$, and $constructionworker$ to $pedestrian$, *bendybus* and *rigidbus* to *bus*. These class labels for *barrier*, *car*, $constructionvehicle$, *truck*, *motorcycle*, $trafficcone$, *trailer*, $driveablesurface$, *sidewalk*, *manmade*, $otherflat$, *terrain* and *vegetation* remain unchanged. The other classes are mapped to *unlabeled*. As a result, we merge 32 classes into 16 classes on the nuScenes-LidarSeg dataset.

4.2 Label Generation

Sparse Label Generation. As described in [8], we project the 3D LiDAR point cloud onto the BEV grid map and perform weighted statistical analysis on the frequency of each class in each grid to obtain the most representative grid-wise semantic label. For each grid, the weighted calculation formula of its label c_i is defined as follows:

$$c_i = argmax_{c \in [1,C]}(w_c n_{i,c}), \tag{7}$$

where C is the number of the semantic classes, w_c denotes the weight for class c, and $n_{i,c}$ represents the number of points of class c in grid i. In addition, the weights of the traffic participant classes, such as *person*, *rider*, *two-wheel*, and *vehicle*, are chosen as 5. The weight of the *unlabeled* class is set as 0 and the weights of other classes are set as 1.

Dense Label Generation. We use the precise pose information provided by SemanticKITTI to aggregate consecutive LiDAR scans and generate dense top-view ground truths, which can provide fine-grained descriptions of the surrounding environment. As in [8], the neighboring LiDAR scans with a distance less than twice the farthest distance are selected as the supplement to the current frame. Based on the provided poses, we transform the adjacent LiDAR point clouds to the coordinate system of the current scan, and then we can get dense aggregation following Eq. 7. In addition, to avoid confusion caused by overlapping, we only aggregate static objects and ignore moving objects.

4.3 Evaluation Metrics

To evaluate the performance of the proposed dense top-view semantic segmentation method, we apply the widely used intersection-over-union (IoU) and mean intersection-over-union (mIoU) in all classes, which are defined as follows:

$$IoU_i = \frac{P_i \cap G_i}{P_i \cup G_i}, \quad mIoU = \frac{1}{C}\sum_{i=1}^{C} IoU_i, \tag{8}$$

where P_i denotes the set of pixels whose predicted semantic labels are class i, G_i represents the set of pixels whose corresponding ground truths are class i, and C represents the total number of classes.

ACCV2022 #1011

Table 1. Quantitative results on the SemanticKITTI dataset [4]

Mode	Method	mIoU [%]	vehicle	person	two-wheel	rider	road	sidewalk	other-ground	building	object	vegetation	trunk	terrain
Sparse Train Sparse Eval	Bieder *et al.* [33]	39.8	69.7	0.0	0.0	0.0	85.8	60.3	25.9	72.8	15.1	68.9	9.9	69.3
	Pillar [8]	55.1	79.5	15.8	25.8	51.8	89.5	70.0	38.9	80.6	25.5	72.8	38.1	72.7
	Pillar + Occ [8]	55.3	82.7	20.3	24.5	51.3	90.0	71.2	36.5	81.3	28.3	70.4	38.5	69.0
	Pillar + Occ + P	57.5	85.1	24.7	16.9	60.1	90.7	72.9	38.3	82.9	30.1	80.4	35.4	72.8
	Pillar + Occ + LP	57.8	85.9	24.2	18.3	57.6	91.3	74.2	**39.2**	82.4	29.0	80.6	38.0	72.9
	Pillar + Occ + LGP [9]	58.8	85.8	34.2	26.8	58.5	91.3	74.0	38.1	82.2	28.7	79.5	35.7	71.3
	Our	**67.9**	**89.5**	**59.7**	**52.7**	**74.1**	**92.7**	**76.2**	36.5	**85.8**	**37.5**	**83.3**	**50.6**	**75.7**
Sparse Train Dense Eval	Bieder *et al.* [33]	32.8	43.3	0.0	0.0	0.0	84.3	51.4	22.9	54.7	10.8	51.0	6.3	68.6
	Pillar [8]	37.5	45.1	0.0	0.1	3.3	82.7	57.5	29.7	64.6	14.0	58.5	25.5	**68.9**
	Pillar + Occ [8]	38.4	52.5	0.0	0.2	3.0	**85.6**	60.1	29.8	**65.7**	16.1	56.7	26.2	64.5
	Pillar + Occ + P	40.9	53.3	11.3	13.1	7.0	83.6	**60.3**	30.2	63.4	15.7	**61.4**	24.6	67.2
	Pillar + Occ + LP	**41.5**	**57.3**	11.3	9.5	**10.4**	85.5	60.1	**31.2**	64.6	16.9	59.5	25.3	66.8
	Pillar + Occ + LGP [9]	40.4	55.8	10.8	14.1	9.3	84.5	58.6	26.8	62.4	15.2	59.2	26.3	62.3
	Our	38.5	53.1	**21.2**	**26.4**	4.8	72.8	52.3	22.1	52.1	**20.0**	47.8	**31.5**	57.2
Dense Train Dense Eval	Pillar [8]	42.8	70.3	5.4	6.0	8.0	89.8	65.7	34.0	65.9	16.3	61.2	23.5	67.9
	Pillar + Occ [8]	44.1	72.8	7.4	4.7	10.2	90.1	66.2	32.4	67.8	17.4	63.1	27.6	69.2
	Pillar + Occ + P	44.9	72.1	6.8	6.2	9.9	90.1	65.8	37.8	67.1	18.8	**68.1**	24.7	**71.4**
	Pillar + Occ + LP	44.8	73.0	7.8	6.1	10.6	90.6	66.5	33.7	67.6	17.7	67.6	25.5	70.4
	Pillar + Occ + LGP [9]	44.5	**73.2**	6.5	6.5	9.5	90.8	**66.5**	**34.9**	68.0	18.8	67.0	22.8	70.0
	Our	**48.8**	70.0	**25.9**	**28.0**	**22.5**	**90.8**	65.4	32.7	**68.3**	**20.9**	64.4	**30.6**	66.1

4.4 Implementation Details

We deploy the proposed network on a server with a single NVIDIA Geforce RTX 2080Ti-11GB GPU, running with PyTorch. The initial learning rate is 0.01, the epoch size is 30, and the batch size of 2.

In the preprocessing step, the input LiDAR point cloud is first cropped into $[(-51.2, 51.2), (-51.2, 51.2), (-5.0, 3.0)]$ meters in the x, y, z directions, respectively. Then, the cropped data is divided into 3D representation $\mathbb{R} \in 512 \times 360 \times 32$ by cylindrical partition, where three dimensions represent radius, tangent angle, and height, respectively. After the 3D sparse convolution networks, the features are converted to a BEV map, covering the area of $[(-51.2, 51.2), (-25.6, 25.6)]$ meters in the x, y directions. The size of the BEV map is $B \times 48 \times 256 \times 512$, representing batch size, feature channels, image height and width, respectively. The resolution is $[0.2, 0.2]$ meters. The final output of the network is the semantic prediction result whose size is 256×512. Since the range of the semantic ground truth is $[(-50.0, 50.0), (-25.0, 25.0)]$ meters and the resolution is $[0.1, 0.1]$, we use linear interpolation to zoom in the network output, and then crop it to the same size as the ground truth.

4.5 Results on SemanticKITTI dataset

We use two training modes and two evaluation modes for dense top-view semantic segmentation, following [33]: Sparse Train and Sparse Eval, Sparse Train and

ACCV2022 #1011

Table 2. Quantitative results on the nuScenes-LidarSeg dataset [5].

Mode	Method	**mIoU** [%]	barrier	bicycle	bus	car	const-vehicle	motorcycle	pedestrian	cone	trailer	truck	drivable	other-flat	sidewalk	terrain	manmade	vegetation
Dense Train Dense Eval	Pillar [8]	22.7	10.8	0.0	5.3	1.6	6.0	0.0	0.0	0.8	19.59	0.8	83.4	35.5	45.0	52.3	48.5	54.3
	MASS [9]	32.7	**28.4**	0.0	24.0	35.7	**16.4**	2.9	4.4	0.1	**29.3**	21.2	**87.3**	**46.9**	**51.6**	56.3	56.8	**61.4**
	Our	**33.7**	25.0	**3.2**	**26.1**	**46.9**	15.0	**11.8**	**10.9**	**6.7**	22.6	**25.7**	85.6	40.2	48.3	**58.6**	**62.0**	51.2

Dense Train, Dense Train and Dense Eval. Among them, Sparse Eval represents using the sparse top-view semantic segmentation ground truth derived from a single LiDAR scan, Dense Eval represents using the generated dense top-view ground truth.

Table 1 shows the quantitative comparison with other state-of-the-art methods. The proposed method achieves a performance improvement of **9.1%** over the current best result in the sparse evaluation mode, and **3.9%** improvement in the dense evaluation mode. In particular, our method greatly improves the performance of classes with small spatial size, including *person*, *two-wheel* and *rider*, and also performs well on other classes. In the sparse mode, the IoUs of these three classes are improved by **25.5%**, **25.9%** and **25.6%**, respectively. In the dense mode, they are increased by **18.1%**, **21.8%** and **21.9%**. This proves the effectiveness of our method in semantic segmentation.

4.6 Results on nuScenes-LidarSeg dataset

In addition to SemanticKITTI dataset, we also evaluate our method on the nuScenes-LidarSeg dataset for dense top-view semantic segmentation. As shown in Table 2, our network achieves better performance than other ones. The proposed network obtains a **1.0%** performance improvement over the state-of-the-art method. Our method is superior in categories with sparse points, such as *bicycle*, *motorcycle*, *pedestrian* and *cone*. The IoU of *car* has been significantly improved by **11.2%**.

4.7 Ablation Studies

In this section, we conduct extensive ablation experiments to investigate the effects of different components in our method. We create several variants of our network to verify the contributions of each components Table 3 summarizes the semantic segmentation results on the SemanticKITTI evaluation dataset in dense mode. The *Baseline* represents the method of using raw point features, point-to-BEV projection and a simple encoder-decoder network with traditional convolution blocks. The *Cylinder* represents replacing point features with cylindrical features and direct cylinder-to-BEV projection without point-guidance. The *Cylinder-to-BEV* represents using cylinder-to-BEV projection with point as intermediate. The *ModifiedU-Net* means using a 2D modified U-Net in the 2D semantic segmentation part.

ACCV2022
#1011

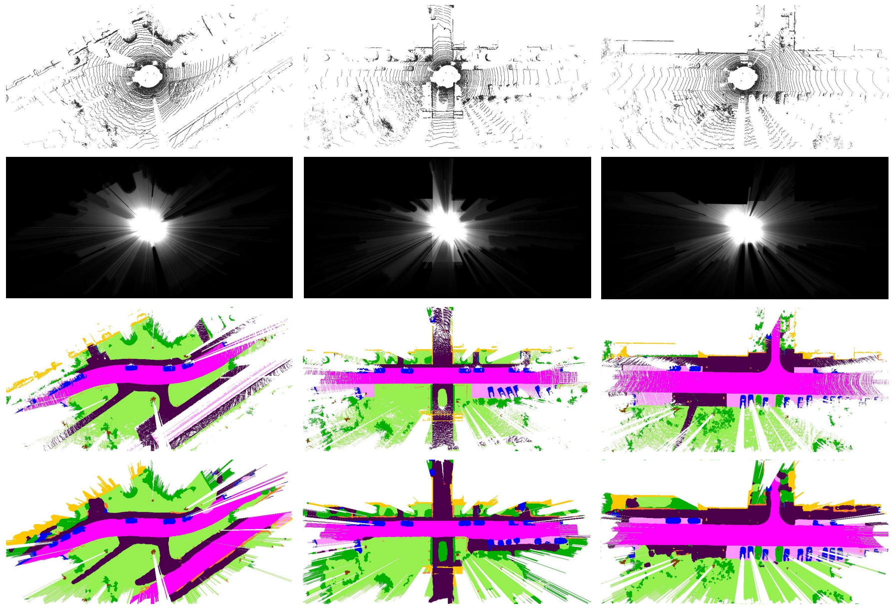


Fig. 4. Qualitative results generated by our approach on the SemanticKITTI validation set. From top to bottom in each column, we display the input point cloud, the 2D occupancy map, the ground truth and the prediction from our method. The unobserved areas were erased using the observability map as in [9]

Table 3. Ablation study on the SemanticKITTI dataset. All experiments are carried out in dense mode.

Baseline	Cylinder	Cylinder-to-BEV	Modified U-Net	mIoU [%]
✓				38.9
✓	✓			45.1
✓	✓	✓		47.5
✓	✓	✓	✓	48.8

The results in Table 3 show that when dealing with outdoor sparse point clouds, the cylindrical encoding is quite successful in gathering rich characteristics from input data, and greatly improves the spatial feature extraction. Compared with methods that ignore 3D information and convert LiDAR data to 2D representation directly, we focus on investigating the spatial geometric relationships of LiDAR points, thus achieving an improvement of 6.2%. The well-designed cylinder-to-BEV module selects key characters in each grid of the 2D top-view, and further increases the performance of 2.4%. The modified U-Net with dilated convolution, depth-wise convolution and inverse bottleneck can also bring a 1.3% performance improvement.

ACCV2022 #1011

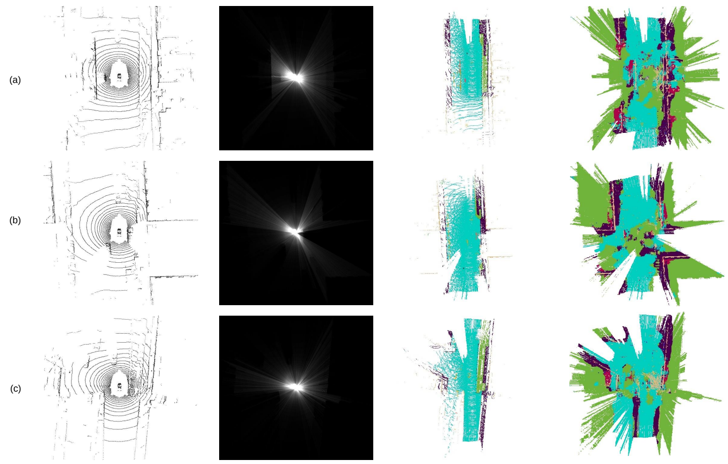


Fig. 5. Qualitative results generated by our approach on the nuScenes dataset. From left to right in each row, we display the input point cloud, the 2D occupancy map, the ground truth and the prediction from our method.

4.8 Qualitative Analysis

As shown in Fig. 4 and Fig. 5, the proposed network can get an accurate semantic understanding of the surrounding environment. It can not only recognize large objects like roads, vehicles, and buildings, but also segment smaller objects more accurately, such as pedestrians, bicycles, motorbikes, and riders. This demonstrates that our method can effectively deal with outdoor, large-scale, sparse, and density-varying 3D point cloud data, and improve the dense semantic segmentation performance in the 2D top-view.

5 Conclusion

In this paper, we propose an end-to-end cylindrical convolution network for dense top-view semantic segmentation with LiDAR data only. We use cylindrical LiDAR representation and 3D CNNs to extract semantic and spatial information, which can effectively preserve more 3D connections and deal with the sparse density of point clouds. Moreover, we introduce an efficient cylinder-to-BEV module to transform features from cylindrical representation to BEV map and provide guidance for the proposed modified U-Net based semantic segmentation in the top-view. We perform extensive experiments and ablation studies on the SemanticKITTI and nuScenes-LidarSeg datasets, and achieve state-of-the-art performance.

References

1. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 3431–3440
2. Romera, E., Alvarez, J.M., Bergasa, L.M., Arroyo, R.: Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. IEEE Transactions on Intelligent Transportation Systems **19** (2017) 263–272
3. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, Springer (2015) 234–241
4. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: Semantickitti: A dataset for semantic scene understanding of lidar sequences. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2019) 9297–9307
5. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. (2020) 11621–11631
6. Cortinhal, T., Tzelepis, G., Erdal Aksoy, E.: Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds. In: International Symposium on Visual Computing, Springer (2020) 207–222
7. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2019) 12697–12705
8. Fei, J., Peng, K., Heidenreich, P., Bieder, F., Stiller, C.: Pillarsegnet: Pillar-based semantic grid map estimation using sparse lidar data. In: 2021 IEEE Intelligent Vehicles Symposium (IV), IEEE (2021) 838–844
9. Peng, K., Fei, J., Yang, K., Roitberg, A., Zhang, J., Bieder, F., Heidenreich, P., Stiller, C., Stiefelhagen, R.: Mass: Multi-attentional semantic segmentation of lidar data for dense top-view understanding. IEEE Transactions on Intelligent Transportation Systems (2022)
10. Huang, J., Huang, G., Zhu, Z., Du, D.: Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. arXiv preprint arXiv:2112.11790 (2021)
11. Ng, M.H., Radia, K., Chen, J., Wang, D., Gog, I., Gonzalez, J.E.: Bev-seg: Bird's eye view semantic segmentation using geometry and semantic point cloud. arXiv preprint arXiv:2006.11436 (2020)
12. Pan, B., Sun, J., Leung, H.Y.T., Andonian, A., Zhou, B.: Cross-view semantic segmentation for sensing surroundings. IEEE Robotics and Automation Letters **5** (2020) 4867–4873
13. Roddick, T., Cipolla, R.: Predicting semantic map representations from images using pyramid occupancy networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 11138–11147
14. Philion, J., Fidler, S.: Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In: European Conference on Computer Vision, Springer (2020) 194–210
15. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 2117–2125

ACCV2022 #1011

16. Yang, W., Li, Q., Liu, W., Yu, Y., Ma, Y., He, S., Pan, J.: Projecting your view attentively: Monocular road scene layout estimation via cross-view transformation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2021) 15536–15545
17. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 652–660
18. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems **30** (2017)
19. Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A.: Randla-net: Efficient semantic segmentation of large-scale point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 11108–11117
20. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE/CVF international conference on computer vision. (2019) 6411–6420
21. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 4490–4499
22. Rethage, D., Wald, J., Sturm, J., Navab, N., Tombari, F.: Fully-convolutional point networks for large-scale point clouds. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 596–611
23. Zhu, X., Zhou, H., Wang, T., Hong, F., Ma, Y., Li, W., Li, H., Lin, D.: Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. (2021) 9939–9948
24. Wu, B., Wan, A., Yue, X., Keutzer, K.: Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE (2018) 1887–1893
25. Wu, B., Zhou, X., Zhao, S., Yue, X., Keutzer, K.: Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In: 2019 International Conference on Robotics and Automation (ICRA), IEEE (2019) 4376–4382
26. Milioto, A., Vizzo, I., Behley, J., Stachniss, C.: Rangenet++: Fast and accurate lidar semantic segmentation. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE (2019) 4213–4220
27. Aksoy, E.E., Baci, S., Cavdar, S.: Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving. In: 2020 IEEE intelligent vehicles symposium (IV), IEEE (2020) 926–932
28. Radi, H., Ali, W.: Volmap: A real-time model for semantic segmentation of a lidar surrounding view. arXiv preprint arXiv:1906.11873 (2019)
29. Zhang, Y., Zhou, Z., David, P., Yue, X., Xi, Z., Gong, B., Foroosh, H.: Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 9601–9610
30. Vedder, K., Eaton, E.: Sparse pointpillars: Exploiting sparsity in birds-eye-view object detection. arXiv e-prints (2021) arXiv–2106

ACCV2022
#1011

31. Paigwar, A., Sierra-Gonzalez, D., Erkent, Ö., Laugier, C.: Frustum-pointpillars: A multi-stage approach for 3d object detection using rgb camera and lidar. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2021) 2926–2933
32. Tchapmi, L., Choy, C., Armeni, I., Gwak, J., Savarese, S.: Segcloud: Semantic segmentation of 3d point clouds. In: 2017 international conference on 3D vision (3DV), IEEE (2017) 537–547
33. Bieder, F., Wirges, S., Janosovits, J., Richter, S., Wang, Z., Stiller, C.: Exploiting multi-layer grid maps for surround-view semantic segmentation of sparse lidar data. In: 2020 IEEE Intelligent Vehicles Symposium (IV), IEEE (2020) 1892–1898
34. Paigwar, A., Erkent, Ö., Sierra-Gonzalez, D., Laugier, C.: Gndnet: Fast ground plane estimation and point cloud segmentation for autonomous vehicles. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE (2020) 2150–2156
35. Kong, X., Zhai, G., Zhong, B., Liu, Y.: Pass3d: Precise and accelerated semantic segmentation for 3d point cloud. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE (2019) 3467–3473
36. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2021) 10012–10022
37. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 4510–4520
38. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2019) 2820–2828
39. Berman, M., Triki, A.R., Blaschko, M.B.: The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 4413–4421

A Cylindrical Convolution Network for Dense Top-View Semantic Segmentation with LiDAR Point Clouds Supplementary Material

Anonymous ACCV 2022 Submission

Paper ID 1011

This supplementary material provides additional visual results that cannot be included in the paper submission due to space limitations. In the first section, we show visual results for sparse predictions of our method on SemanticKITTI dataset [1]. In the second section, we display more visual results on SemanticKITTI dataset, including comparisons with the results of previous methods. Moreover, a demonstration video is included in the same zip file as the supplementary material.

1 Visual Results for Sparse Predictions

In the main text, we have shown dense semantic prediction results. To fully demonstrate the effectiveness of our model, we present sparse semantic prediction results here.

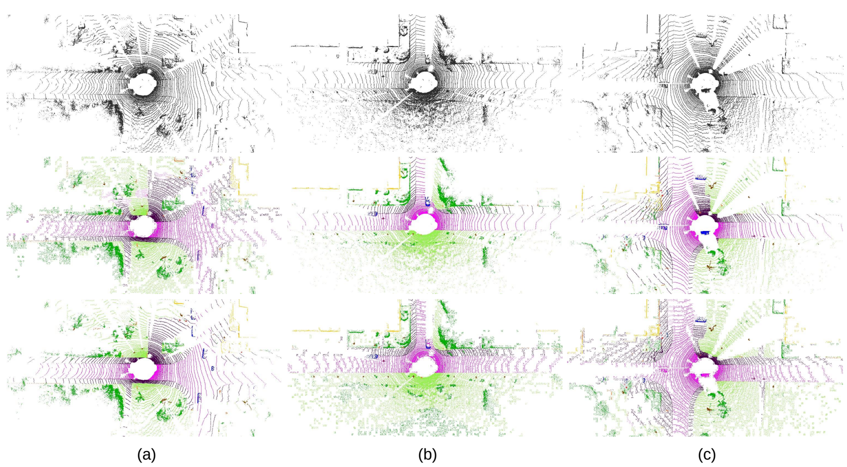


Fig. 1. Qualitative results generated on the SemanticKITTI validation set. From top to bottom in each column, we display the input point cloud, the ground truth, the prediction from our method, respectively.

ACCV2022 #1011

2 Addition Visual Results on SemanticKITTI

Here we show two groups of comparisons with the results for Bieder *et al.* [2], PillarSeg [3], MASS [4] and our method on SemanticKITTI. For a fair comparison, the unobservable regions in our predictions are also filtered out using the observability map as in [2].

As shown in Fig 2 and Fig 3, our method is able to produce very similar results to the ground truth for challenging urban scenes. Compared with other methods, our method achieves a higher level of accuracy, especially for the prediction of small volume objects.

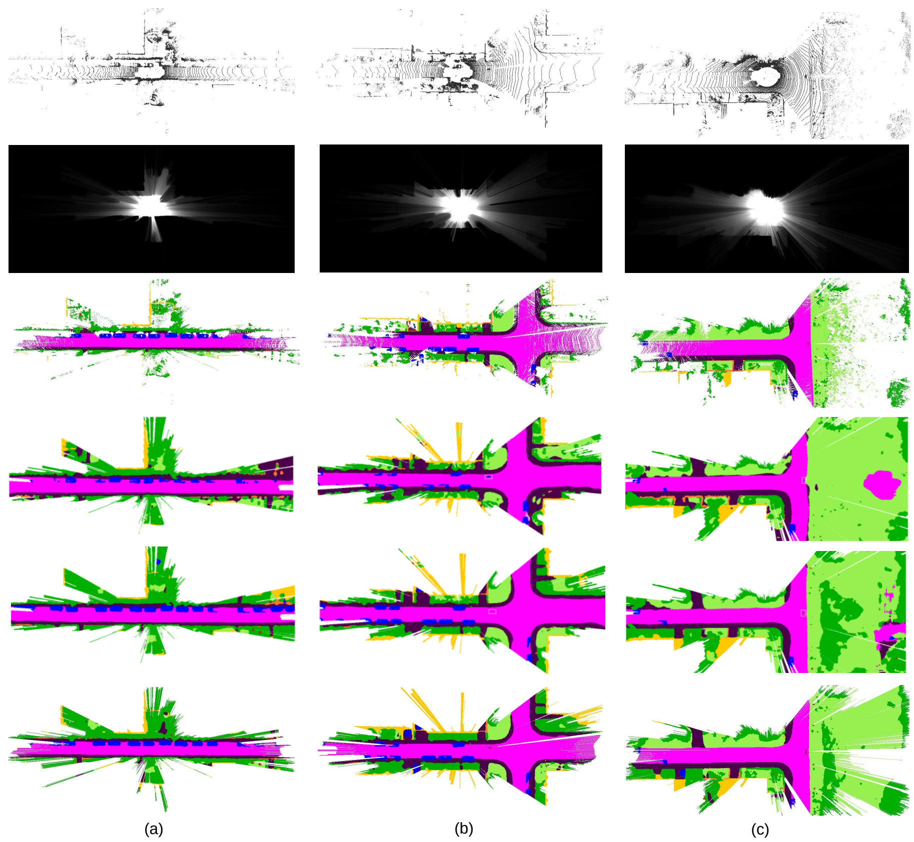
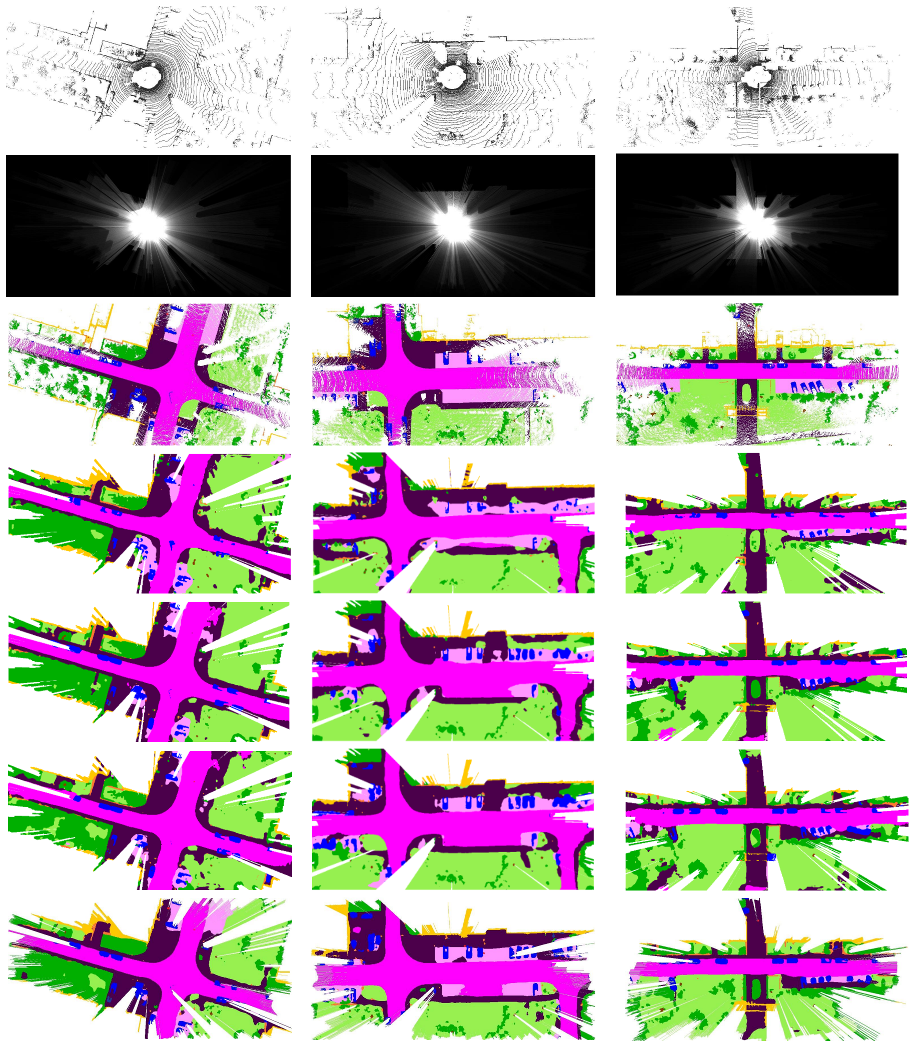


Fig. 2. Qualitative results generated on the SemanticKITTI validation set. From top to bottom in each column, we display the input point cloud, the 2D occupancy map, the ground truth, the prediction from Bieder *et al.* [2], PillarSeg [3] and our method, respectively. The unobserved areas were erased using the observability map as in [2]



(a) (b) (c)

Fig. 3. Qualitative results generated on the SemanticKITTI [1] validation set. From top to bottom in each column, we display the input point cloud, the 2D occupancy map, the ground truth, the prediction from Bieder *et al.* [2], PillarSeg [3], MASS [4] and our method, respectively. The unobserved areas were erased using the observability map as in [2]

References

1. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: Semantickitti: A dataset for semantic scene understanding of lidar sequences.

ACCV2022
#1011

In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2019) 9297–9307
2. Bieder, F., Wirges, S., Janosovits, J., Richter, S., Wang, Z., Stiller, C.: Exploiting multi-layer grid maps for surround-view semantic segmentation of sparse lidar data. In: 2020 IEEE Intelligent Vehicles Symposium (IV), IEEE (2020) 1892–1898
3. Fei, J., Peng, K., Heidenreich, P., Bieder, F., Stiller, C.: Pillarsegnet: Pillar-based semantic grid map estimation using sparse lidar data. In: 2021 IEEE Intelligent Vehicles Symposium (IV), IEEE (2021) 838–844
4. Peng, K., Fei, J., Yang, K., Roitberg, A., Zhang, J., Bieder, F., Heidenreich, P., Stiller, C., Stiefelhagen, R.: Mass: Multi-attentional semantic segmentation of lidar data for dense top-view understanding. IEEE Transactions on Intelligent Transportation Systems (2022)