Spherical Frustum Sparse Convolution Network for LiDAR Point Cloud Semantic Segmentation

Supplemental Material

In the supplemental material, we first introduce the detailed architecture of the Spherical Frustum sparse Convolution Network (SFCNet) in Sec. A. Then, the additional implementation details of SFCNet are presented in Sec. B. Next, the additional experimental results are illustrated in Sec. C. Finally, more visualization of the semantic segmentation results on the SemanticKITTI [1] and nuScenes [2] datasets are presented in Sec. D.
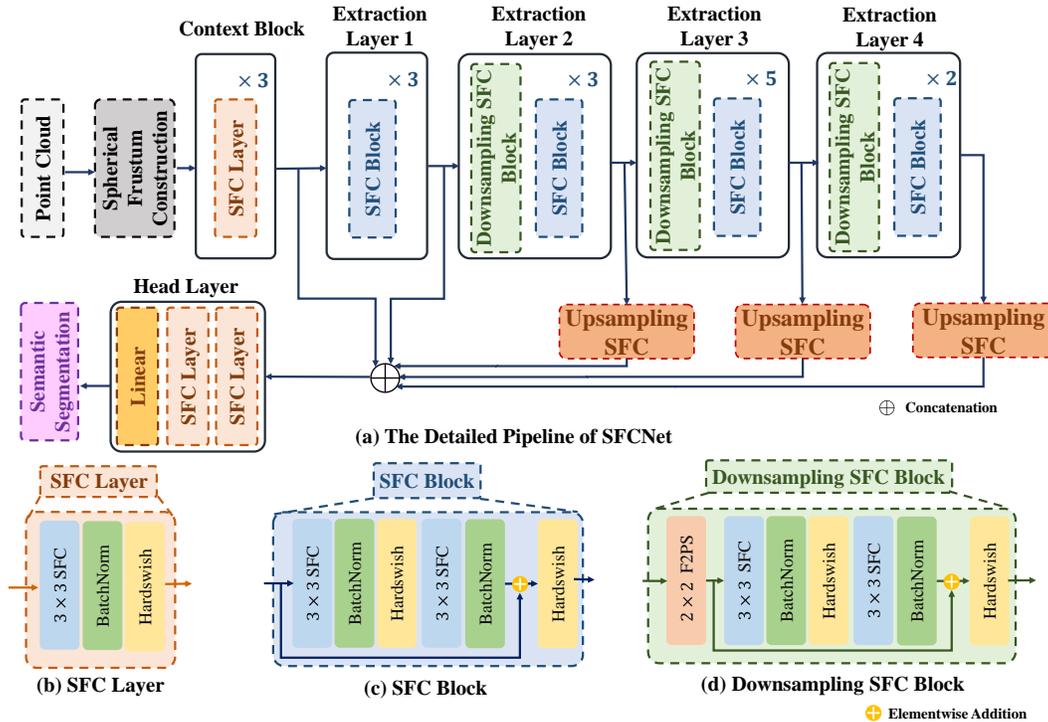
Figure 1: The Detailed Architecture of SFCNet. (a) presents the detailed pipeline of SFCNet. In addition, (b), (c), and (d) show the detailed module structures of the SFC layer, SFC block, and downsampling SFC block respectively, where SFC means spherical frustum sparse convolution, and F2PS means the frustum farthest point sampling.

A Detailed Architecture

Fig. 1 shows the detailed architecture of SFCNet. In SFCNet, the spherical frustum structure of the input point cloud is first constructed. Then, the encoder, which consists of the context block and extraction layers 1 to 4, is adopted for the point feature extraction. Next, in the decoder, the point features extracted in extraction layers 2 to 4 are upsampled by the upsampling Spherical Frustum sparse Convolution (SFC). The upsampled features are concatenated with the features extracted in

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Table 1: The Detailed Hyperparameters of the Components and Modules in SFCNet. Component shows the names of components in SFCNet. Module Type shows the types of basic modules used in the components. Kernel Size shows the convolution kernel size of Spherical Frustum sparse Convolutions (SFCs) used in the modules. In the column of Stride, $[1, 1]$ strides mean the SFC treats all the points as the center points, while $[2, 2]$ strides show the strides used in Frustum Farthest Point Sampling (F2PS). The column of Upsampling Rate shows the upsampling rate used in the upsampling SFCs. Number of Modules shows the number of composed modules used in corresponding components. Width shows the output channel dimensions of the modules. In addition, in the column of Width, C means the channel dimensions of the extracted point features, which is 128 for the SemanticKITTI dataset and 256 for the nuScenes dataset. n means the number of semantic classes, which is 19 for the SemanticKITTI dataset and 16 for the nuScenes dataset.

Component	Module Type	Kernel Size	Stride	Upsampling Rate	Number of Modules	Width
Context Block	SFC Layer	[3,3]	[1,1]	—	3	[C/2,C,C]
Extraction Layer 1	SFC Block	[3,3]	[1,1]	—	3	[C,C,C]
Extraction Layer 2	Downsampling SFC Block	[3,3]	[2,2]	—	1	[C]
	SFC Block	[3,3]	[1,1]	—	3	[C,C,C]
Extraction Layer 3	Downsampling SFC Block	[3,3]	[2,2]	—	1	[C]
	SFC Block	[3,3]	[1,1]	—	5	[C,C,C,C,C]
Extraction Layer 4	Downsampling SFC Block	[3,3]	[2,2]	—	1	[C]
	SFC Block	[3,3]	[1,1]	—	2	[C,C]
Upsampling	Upsampling SFC for Extraction Layer 2	[3,3]	[1,1]	[2,2]	1	[C]
	Upsampling SFC for Extraction Layer 3	[7,7]	[1,1]	[4,4]	1	[C]
	Upsampling SFC for Extraction Layer 4	[15,15]	[1,1]	[8,8]	1	[C]
Head Layer	SFC Layer	[3,3]	[1,1]	—	2	[2·C,C]
	Linear	—	—	—	1	[n]

the context block and the extraction layer 1. The concatenated features are fed into the head layer to decode the point features into the semantic predictions.

In addition, Fig. 1 also shows the three basic modules in SFCNet, including the SFC layer, SFC block, and downsampling SFC block. Moreover, we present the detailed hyperparameters of SFCNet in Tab. 1.

Basic Modules of SFCNet. Specifically, the SFC layer is composed of the SFC, batch normalization, and the activation function. Inspired by [3], we use Hardswish [4] as the activation function. The formula of Hardswish is:

$$Hardswish(x) = \begin{cases} 0 & if \ x \leq -3 \\ x & if \ x \geq 3 \\ x \cdot (x+3)/6 & otherwise \end{cases} . \tag{1}$$

The SFC block consists of two SFC layers. In addition, the residual connection [5] is adopted in the SFC block to overcome network degradation.

The downsampling SFC block combines the downsampling of Frustum Farthest Point Sampling (F2PS) and the feature aggregation of the SFC block. Notably, in the downsampling SFC block, the first SFC treats the sampled points as the center points and the features of the point cloud before sampling as the aggregated features.

Moreover, after the downsampling, the 2D coordinates of each spherical frustum are divided by the stride to gain the 2D coordinates on the downsampled 2D spherical plane. Meanwhile, each point is assigned a new point index in the downsampled spherical frustum point set according to the sampled order in F2PS.

Components in the Encoder of SFCNet. In the encoder, the context block consists of three SFC layers to extract the initial point features from the original point cloud. The subsequent four extraction layers are composed of $3, 3, 5$, and 2 SFC blocks respectively. In addition, a downsampling SFC block with $(2, 2)$ strides is adopted in the last three layers to downsample the point cloud into different scales. Thus, the multi-scale point features are extracted.

Components in the Decoder of SFCNet. We implement the upsampling SFC in the decoder of SFCNet according to the deconvolution [6] used in the 2D convolutional neural networks. In the

2

upsampling SFC, we first multiply the 2D coordinates of the spherical frustums in the corresponding layer by the upsampling rate to obtain the 2D coordinates on the original spherical plane. Then, each point in the raw point cloud is treated as the center point in SFC. The spherical frustums fall in the convolution kernel are convolved. As shown in Tab. 1, we set the appropriate kernel size according to the upsampling rate for each upsampling SFC.

After the upsampling, the point features from different extraction layers are of the same size. Thus, the point features can be concatenated. In the head layer, two SFC layers and a linear layer are adopted for the decoding of the concatenated features.

B Additional Implementation Details

Data Normalization. For the k-th point in the LiDAR point cloud $\mathcal{P}$, the combination of the 3D coordinates $\boldsymbol{x}_k = [x_k, y_k, z_k]^T$, the range $r_k = \sqrt{x_k^2 + y_k^2 + z_k^2}$, and the intensity is treated as the input point feature $\boldsymbol{f}_k$. Because of the different units of the different data categories, the input features should be normalized.

Specifically, for the SemanticKITTI [1] dataset, like RangeNet++ [7], we minus the features by the mean and divide the features by the standard deviation to obtain the normalized features. The mean and standard deviation are obtained from the statistics of each input data category on the SemanticKITTI dataset, which are presented in Tab. 2.

Table 2: The statistics of each input data category on SemanticKITTI dataset.

Statistics	x	y	z	$range$	$intensity$
Mean	10.88	0.23	-1.04	12.12	0.21
Standard Deviation	11.47	6.91	0.86	12.32	0.16

For the nuScenes [2] dataset, like Cylinder3D [8], a batch normalization layer is applied on the input point features to record the mean and standard deviation of the nuScenes dataset during training. During inferencing, the recorded mean and standard deviation are used to normalize the input point features.

Spherical Frustum Construction. We construct the spherical frustum structure by assigning each point with the 2D spherical coordinates (u_k, v_k) and the point index m_k in the spherical frustum point set, where k is the index of the point in the original point cloud. The 2D spherical coordinates can be calculated through Eq. **??**. Thus, the key process is to assign the point index m_k for each point based on the 2D spherical coordinates.

We implement this by sorting the 2D coordinates (u_k, v_k) of the points. The points with smaller u_k and v_k are ranked ahead of the points with larger u_k and v_k. Thus, the points with the same 2D coordinates are neighbors in the sorted point cloud. For each point, we count the number of points that have the same 2D coordinates and appear ahead or behind the point in the sorted point cloud separately. The number of the points appearing ahead is treated as the point index m_k of each point.

In addition, we assign each point an indicator $\xi_k \in \{0, 1\}$ according to the number of the points appearing behind. The point with zero point appearing behind is assigned a zero indicator. Otherwise, the point is assigned with an indicator equal to one. The indicator indicates the end of the frustum point set and is used for the subsequent spherical frustum point set visiting.

The sorting and the point number counting are implemented through the Graphics Processing Unit (GPU)-based parallel computing using Compute Unified Device Architecture (CUDA). Thus, the construction is efficient in practice.

Hash-Based Spherical Frustum Representation. After the construction of the spherical frustum structure, we build the hash-based spherical frustum representation. Specifically, we construct the key-value pairs between the key (u_k, v_k, m_k) and the value k. The key-value pairs are inserted into a hash table, which represents the neighbor relationship of spherical frustums and points.

In practice, we adopt an efficient GPU-based hash table [9]. The GPU-based hash table requires both key and value to be an integer. The value k satisfies the integer requirement. However, the key (u_k, v_k, m_k) in the hash-based spherical frustum representation is not an integer.

To adopt the GPU-based hash table for efficient processing, (u_k, v_k, m_k) is transferred to an integer as $v_k \cdot (W \cdot M) + u_k \cdot M + m_k$, where W is the width of the spherical projection, M is the maximal

3

point number of the spherical frustum point sets. Through this process, any point represented by the coordinates (u_k, v_k, m_k) can be efficiently queried through the GPU-based hash table.

Spherical Frustum Point Set Visiting. Both the SFC and F2PS require visiting all the points in any spherical frustums. Thus, we propose the spherical frustum point set visiting algorithm. The visiting obtains all the points in the given spherical frustum, whose 2D coordinates are (u, v), by sequentially querying the points in the hash table.

Specifically, we first query the first point in the spherical frustum using the key $(u, v, 0)$. If the key $(u, v, 0)$ is not in the hash table, the spherical frustum on (u, v) is invalid. Otherwise, the first point in the spherical frustum can be queried through the hash table.

Then, the points in the spherical frustum are sequentially visited. We first initialize the point index $m = 0$ in the spherical frustum. At each step, the point index m increases by one. Through the hash table, the point with m-th index in the spherical frustum is queried using the key (u, v, m). Meanwhile, the indicator ξ of this point is obtained. ξ indicates whether $(u, v, m + 1)$ refers to a valid point. Thus, the visiting ends when the indicator of the current point is zero.

Detailed Implementation of Frustum Farthest Point Sampling. In F2PS, we first sample the spherical frustums by stride. Then, we sample the points in each sampled spherical frustum by Farthest Point Sampling (FPS) [10]. As mentioned in Sec. **??**, FPS is an iterative algorithm. The detailed process of the j-th iteration can be expressed by the following formula:

$$S_j = S_{j-1} \cup \{\arg \max_{p \in P_s \backslash S_{j-1}} \min_{s \in S_{j-1}} dist(p, s)\}, \tag{2}$$

where P_s is the spherical frustum point set to be sampled, S_j and S_{j-1} are the sampled point sets in j-th and $(j-1)$-th iterations respectively. Notably, S_0 contains the point randomly sampled from P_s. In addition, $dist(p, s)$ is the distance between point p and point s in 3D space. The iteration starts at $j = 1$, and ends when the size of S_k equals the number of sampling points.

Moreover, since the distances between the points in S_{j-2} and the points in $P_s \backslash S_{j-1}$ have been calculated before the j-th iteration, we just need to calculate the distance between each p in $P_s \backslash S_{j-1}$ and the point sampled in $(j-1)$-th iteration for the calculation of $\min_{s \in S_{j-1}} dist(p, s)$, which is the minimal distance from point p to the point set S_{j-1}. Thus, the computing complexity of FPS for P_s of size n is $O(n^2)$. Since the point number of each spherical frustum is $O(1)$, the computing complexity of FPS for the spherical frustum is also $O(1)$, which ensures the efficiency of F2PS.

Loss Function. We use multi-layer weighted cross-entropy loss and Lovász-Softmax loss [11] to help the network learn the semantic information from different scales. To get the semantic predictions of extraction layers 1 to 4, we apply a linear layer to decode the extracted point features of each extraction layer into the semantic predictions.

Specifically, for extraction layer 1, the linear layer is applied on the extracted point features $\mathcal{F}_1$ to gain the prediction $\tilde{L}_1$. For the other extraction layers, the linear layer is applied on the upsampled point features $\mathcal{F}_2'$, $\mathcal{F}_3'$, and $\mathcal{F}_4'$ to obtain the predictions $\tilde{L}_2$, $\tilde{L}_3$, and $\tilde{L}_4$ respectively.

Based on the predictions of each layer and the final predictions of SFCNet $\tilde{L}_1$, the loss function is calculated as:

$$\mathcal{L} = \sum_{i=1}^{4} \mathcal{L}_{wce}(\tilde{L}_i, L) + \mathcal{L}_{Lov}(\tilde{L}_i, L), \tag{3}$$

where $\mathcal{L}_{wce}$ is the weighted cross-entropy loss, $\mathcal{L}_{Lov}$ is the Lovász-Softmax loss, and L is the ground truth. In addition, the weights of weighted cross-entropy loss are calculated as $w_c = (f_c + \epsilon)^{-1}$, where c is the semantic class, f_c is the frequency of class c in the dataset, and ϵ is a small positive value to avoid zero division.

C Additional Experiments

C.1 Efficiency Comparison

We evaluate the efficiency of the proposed SFCNet with the previous works and our 2D projection-based baseline model on a single Geforce RTX 4090Ti GPU.

4

Table 3: Efficiency comparison. The inference time of a single LiDAR scan, the processed point number, and the normalized time, the inference time per thousand points, are evaluated on the SemanticKITTI validation set with a single Geforce RTX 4090Ti GPU.

Approach	Time (ms)/Points ↓	Normalized Time (ms/K) ↓
PointNet++ [10]	131.0/$\sim 45K$	2.91
RandLA [14]	212.2/$\sim 120K$	1.77
Cylinder3D [8]	67.5/$\sim 40K$	1.69
SphereFormer [13]	108.2/$\sim 90K$	1.20
RangeViT [15]	104.8/$\sim 120K$	0.87
Baseline	46.4/$\sim 90K$	0.52
SFCNet (Ours)	**59.7/$\sim$ 120K**	**0.49**

Table 4: The quantitative results of different resolutions used in the baseline model with KNN-based post-processing [7] on the SemanticKITTI validation set.

Resolution	Preserved Points/All Points	mIoU (%) ↑
64×1800	$88K/120K$	**59.7**
64×2048	$97K/120K$	58.9
64×4096	$113K/120K$	57.0

We adopt the same baseline model used in Sec. **??**. For RangeViT, we adopt the official code for efficiency evaluation. Notably, in the inference, RangeViT splits the projected LiDAR image, inputs each image slice into the network to gain the predictions, and merges the predictions to gain the prediction of the entire projected LiDAR image. Thus, the inference time of RangeViT includes the time of all the processes. In addition, since RangeViT adopts the KPConv refinement [12], which restores the complete predictions from the partial predictions, we use the point number of the entire point cloud as the processed point number. For PointNet++ [10], we sample $45K$ points from the point cloud before inputting into the network as its original setting. For 3D voxel-based methods, Cylinder3D [8] and SphereFormer [13], only the points preserved after the voxelization are counted since these points are exactly processed in the 3D sparse convolution network.

The results are presented in Tab. 3. The results show that SFCNet costs 59.7 ms for a single scan inference, which reaches real-time LiDAR scan processing. In addition, our SFCNet also has the highest efficiency evaluated by normalized time (0.49 ms/K) compared to the previous 3D and 2D methods and the 2D baseline model, which indicates that SFCNet can adopt the 2D projection property to efficiently segment the large-scale point cloud.

C.2 Analysis on Different Resolutions of the Baseline Model

Since the limited projection resolution is the reason for quantized information loss, expanding the resolution of the projected range image can preserve more points during the spherical projection and ease the quantized information loss. However, expanding the resolution increases the sparsity of the projected points and makes the convolution hard to aggregate the local features. Thus, resolution expansion is not a feasible solution for resolving quantized information loss. To validate this, we expand the image horizon resolution of the baseline model to 2048 and 4096 and conduct the ablation studies of different resolutions on the SemanticKITTI validation set to show the effect of a larger resolution. As shown in Tab. 4, the increment of resolution preserves more points but results in worse performances. In contrast, SFCNet not only overcomes quantized information loss but also effectively aggregates local features with a suitable resolution by preserving all points using spherical frustum.

C.3 Additional Ablation Studies

In this subsection, we conduct additional ablation studies to evaluate the sensitivity of our SFCNet to the key parameters.

Stride Sizes in Frustum Farthest Point Sampling (F2PS). The ablation studies of four different settings of the stride sizes in the F2PS on the three downsampling layers are conducted, including $(1, 2), (2, 1), (2, 4)$, and $(4, 2)$. The results are shown in Tab. 5. The results show on all the downsampling layers, the $(2, 2)$ stride sizes show a better segmentation performance than the other stride

5

Table 5: Ablation study on the stride sizes of the Frustum Farthest Point Sampling in the downsampling layers on the SemanticKITTI validation set.

Stride Sizes (S_h, S_w)	mIoU (%) ↑		
	Layer 1	Layer 2	Layer 3
(2,1)	60.7	61.3	61.1
(1,2)	62.4	62.2	62.3
(2,4)	62.3	62.6	61.9
(4,2)	60.5	61.6	61.9
(2,2) (Ours SFCNet)		**62.9**	

Table 6: Ablation study on the maximal number of points in spherical frustums on the SemanticKITTI validation set.

Maximal Number of Points in Spherical Frustum	mIoU (%) ↑
2	61.0
4	61.9
Unlimited (Ours SFCNet)	**62.9**

size settings. $(2, 2)$ stride sizes suitably downsample the point cloud in the vertical and horizon dimensions. Higher or lower downsampling rates result in the oversampling or undersampling of the point cloud respectively.

Number of Points in the Spherical Frustums. In the spherical frustum structure, the number of points in the frustum is unlimited and only depends on how many points are projected onto the corresponding 2D location. To analyze the effect of the number of points in the frustum, we set the maximal number of points in each spherical frustum and the points exceeding the maximal point number are dropped. As shown in Tab. 6, preserving more points in the spherical frustum results in better segmentation performance, since more complete geometry information is preserved. These results further indicate the significance of overcoming quantized information loss in the field of LiDAR point cloud semantic segmentation.

Configuration of the Hash Table. The number of hash functions is the main parameter of the hash table, which means the number of functions used for the hash table retrieval. In the implementation, if the first hash function can successfully retrieve the location of the target point, the other functions will not be used. We change the number of hash functions to show the model sensitivity of hash table configurations. As shown in Tab. 7, the performance and inference time of SFCNet have little difference under different numbers of hash functions. The results show that in most cases, the first function can successfully retrieve the location, and thus the inference times change slightly in different function numbers. These results indicate that SFCNet is robust to different hash table configurations.

C.4 Comparison of Sampling Methods

We further validate the effectiveness and efficiency of the proposed Frustum Farthest Point Sampling (F2PS) by the qualitative comparison with stride-based 2D sampling and the comparison of time consumption with Farthest Point Sampling (FPS).

Qualitive Comparison. As shown in Fig. 2(a), Stride-Based 2D Sampling (SBS) only samples the point cloud based on 2D stride. The visualization shows that the stride-based sampled point cloud is relatively rough. Due to the lack of 3D geometric information, SBS fails to sample the 3D point cloud uniformly. Thus, the loss of geometric structure in the sampled point cloud is obvious, such as many broken lines on the ground. Our F2PS takes into account the 3D geometric information based on the FPS in the spherical frustum, which enables F2PS to sample the 3D point cloud uniformly and preserve the significant 3D geometric structure during the sampling.

Time Consumption Comparison. As shown in Fig. 2(b), with the increment of sampled point number, the cost time of our F2PS increases slowly, while the cost time of FPS increases dramatically. This result shows performing FPS on the frustum point sets is efficient and does not increase the computing burden.

6

Table 7: Ablation study on the configuration of the hash table for the spherical frustum structure on the SemanticKITTI validation set.

Number of Hash Functions	Inference Time (ms) ↓	mIoU (%) ↑
2	59.5	62.9
3	60.1	62.9
5	59.5	62.9
4 (Ours SFCNet)	59.7	62.9

Original Point Cloud	Stride-Based Sampled Point Cloud	**F2PS Sampled Point Cloud (Ours)**
	(a) Qualitative Comparison	

(b) Time Consumption Comparison
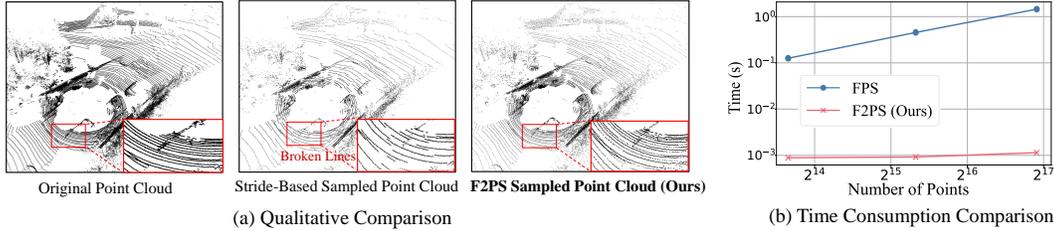
Figure 2: In this figure, (a) presents the qualitative comparison between stride-based 2D sampling and our Frustum Farthest Point Sampling (F2PS). The red boxes show the zoomed view of the point clouds in the close areas. (b) illustrates the time consumption comparison between Farthest Point Sampling (FPS) and our F2PS.

C.5 Comparison between SFCNet and Baseline Model on Small Object Categories

To further show the improvement of small object segmentation, we compare the quantitative results on the small object categories between SFCNet and the baseline model with the KNN-based post-processing [7] on SemanticKITTI and nuScenes validation sets. As shown in Tabs. 8 and 9, our SFCNet has higher performances on all small object categories compared to the baseline model. The results show overcoming quantized information loss preserves complete geometric information of the small objects and thus makes them better recognized and segmented by our SFCNet.

D More Visualization

To better demonstrate the effectiveness of SFCNet for LiDAR point cloud semantic segmentation, we conduct more visualization on the SemanticKITTI and nuScenes datasets. The results are shown in Figs. 3, 4, 5, 6 and the supplementary video `SupplementaryVideo.mp4`.

Qualitative Comparison on NuScenes Validation Set. The results of qualitative comparison between our SFCNet and RangeViT [15] are shown in Fig. 3. On the nuScenes validation set, SFCNet can also have fewer segmentation errors than RangeViT as the results in the SemanticKITTI dataset. Moreover, the better segmentation accuracy of the 3D small objects, like pedestrians and motorcycles, can also be observed on the nuScenes validation set. The results once more demonstrate semantic segmentation improvement of SFCNet due to the overcoming of quantized information loss.

More Qualitative Comparison on SemanticKITTI Test Set. The ground truths on the SemanticKITTI test set are not available. Thus, we search for the corresponding RGB image and project the semantic predictions on the image to compare the semantic segmentation accuracy between the state-of-the-art 2D image-based method CENet [3] and our SFCNet on the SemanticKITTI test set. As shown in Figs. 4 and 5, compared to CENet, SFCNet can more accurately segment the LiDAR point cloud in various challenging scenes on the SemanticKITTI test set.

Specifically, SFCNet recognizes the thin poles in distance on the rural road of Fig. 4(a) and in the complex intersections of Fig. 5(c), while CENet predicts the poles as wrong classes. In addition, SFCNet recognizes the thin trunks inside the vegetation on the rural scenes of Fig. 4(b) and Fig. 5(b) while CENet wrongly predicts the trunk as the fetch and vegetation respectively. Moreover, SFCNet successfully segments the boxed persons in the complex intersection of Fig. 4(c) and in the urban scene of Fig. 5(a) while CENet gives wrong predictions due to the information loss of the distant persons during 2D projection. These results further validate the better segmentation performance of SFCNet to 3D small objects.

7

Table 8: Quantative comparison of semantic segmentation between baseline model and SFCNet for the small object categories on SemanticKITTI validation sets. **Bold** results are the best in each column. The performance improvement of each category is highlighted in green.

Approach	SemanticKITTI							
	bicycle	motorcycle	person	bicyclist	motorcyclist	trunk	pole	traffic-sign
Baseline w/ KNN-Based Post-processing	44.2	46.0	48.8	71.6	73.6	67.4	63.1	45.7
SFCNet (Ours)	**44.9**(+0.7)	**60.6**(+14.6)	**50.5**(+1.7)	**73.1**(+1.5)	**83.1**(+9.5)	**68.5**(+1.1)	**64.6**(+1.5)	**47.8**(+2.1)

Table 9: Quantative comparison of semantic segmentation between baseline model and SFCNet for the small object categories on the nuScenes validation sets. **Bold** results are the best in each column. The performance improvement of each category is highlighted in green.

Approach	nuScenes			
	bicycle	motorcycle	pedestrian	traffic-cone
Baseline w/ KNN-Based Post-processing	30.6	77.0	73.9	62.8
SFCNet (Ours)	**40.4**(+9.8)	**82.0**(+5.0)	**78.0**(+4.1)	**65.8**(+3.0)

More Qualitative Comparison on NuScenes Validation Set. As the visualization on the SemanticKITTI test set, we provide the additional qualitative comparison between our SFCNet and RangeViT on the nuScene validation set with the projected predictions illustrated in Fig. 6. The results further demonstrate the better semantic segmentation of SFCNet for the challenging street scenes on the nuScenes validation set compared to RangeViT.

Specifically, in the first scene, the close motorcycle can be correctly segmented by SFCNet, while RangeViT recognizes the motorcycle as a car, which shows that SFCNet can help the autonomous car correctly recognize the type of close obstacles, and enable the car to make appropriate decisions.

In the second scene, the distant pedestrians on the other side of the crossing can also be correctly segmented by SFCNet due to the elimination of quantized information loss. In contrast, RangeViT wrongly predicts the pedestrians as traffic cones.

In the third scene, since the boxed pedestrian is close to the manmade, RangeViT confuses it with the manmade and does not segment the pedestrian, while our SFCNet can clearly recognize the boundary and successfully segments the pedestrian.

Sequential Qualitative Comparison on SemanticKITTI Validation Set. We demonstrate the qualitative comparison between our SFCNet and the SoTA 2D projection-based segmentation methods, CENet and RangeViT, on a continuous sequence on the SemanticKITTI validation set in the supplementary video `SupplementaryVideo.mp4`. In this video, the semantic predictions in both the 3D point cloud view and the RGB image view (where the colored point cloud is projected onto the RGB images) are presented. The results show that our SFCNet can consistently show higher segmentation accuracy on the point cloud of each frame in the sequence than 2D projection-based methods, which further indicates the stronger semantic segmentation capability of our SFCNet.

8
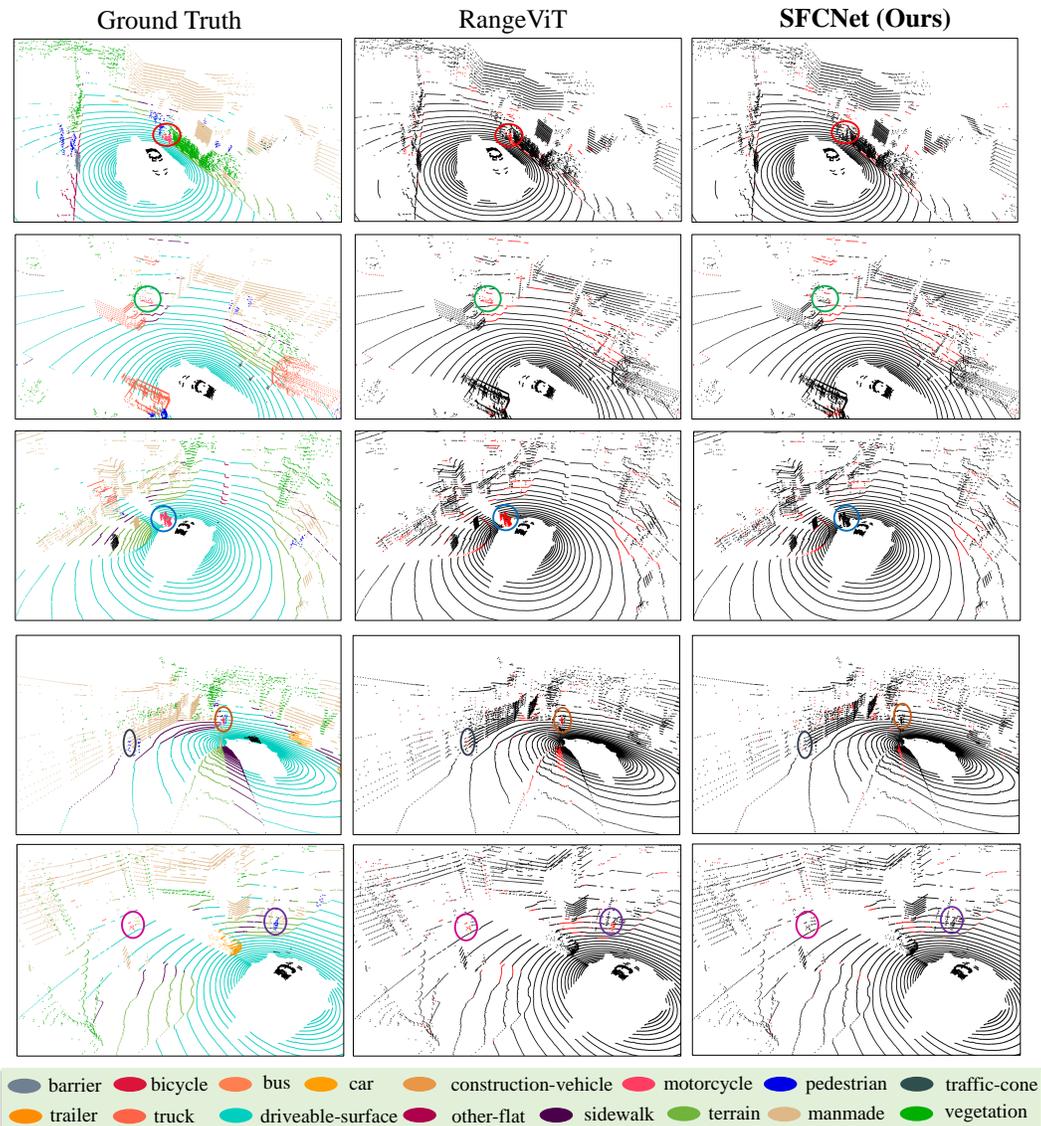
Figure 3: Qualitative Comparison on NuScene Validation Set. In this figure, we conducted the qualitative comparison between RangeViT [15] and our SFCNet of semantic segmentation on the nuScenes validation set. The first column presents the ground truths, while the following two columns show the error maps of the predictions of RangeViT and our SFCNet respectively. In addition, the reference from point color to the semantic class in the ground truths is shown at the bottom. Moreover, the false-segmented points are marked as red in the error maps. Furthermore, we use circles with the same color to point out the same objects in the ground truth and the two error maps.

9

Original RGB Image   Prediction   Projected Prediction

(a) Urban

(b) Rural

(c) Complex Intersections

CENet

SFCNet (Ours)

CENet

SFCNet (Ours)

CENet

SFCNet (Ours)

car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road
parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
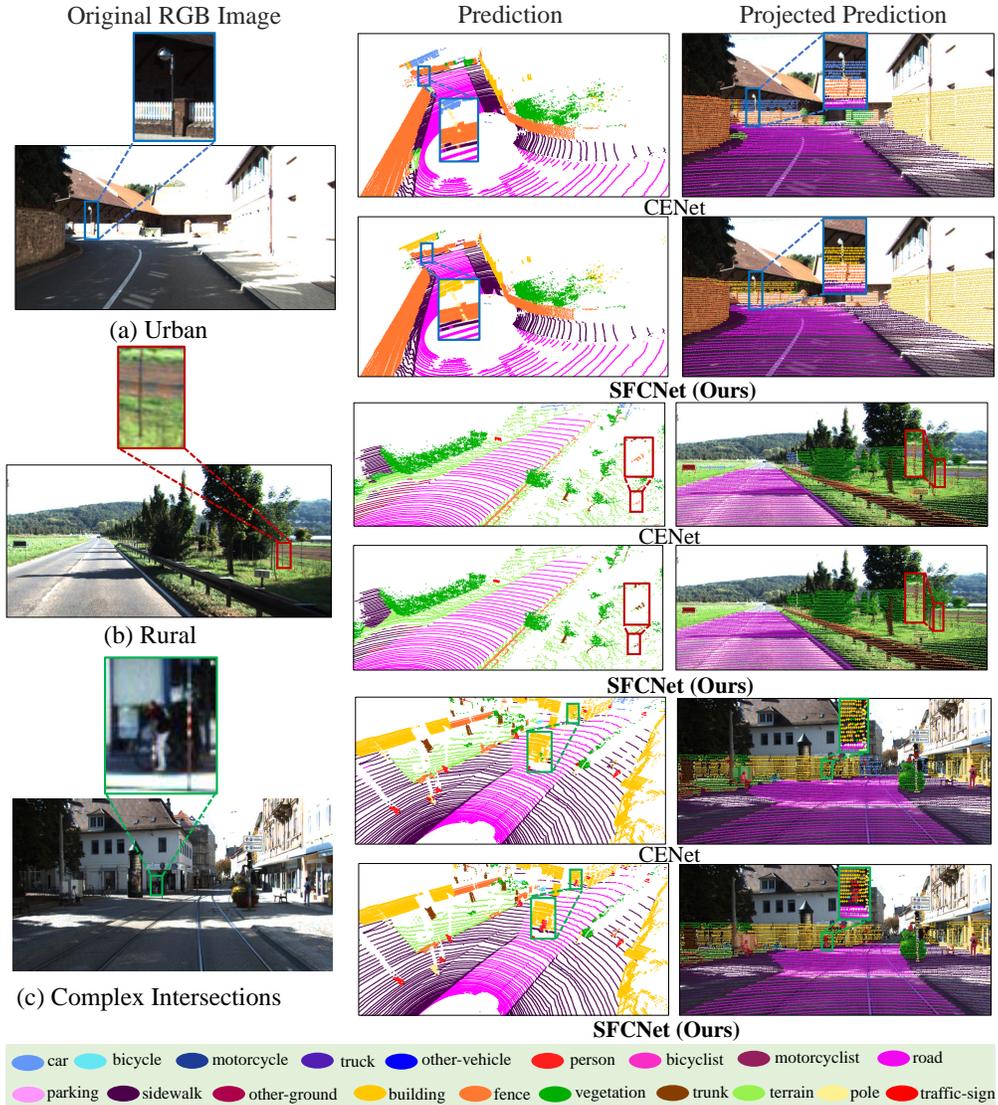
Figure 4: More Qualitative Comparison on Semantic Segmentation on SemanticKITTI Test Set. We show the qualitative comparison between our SFCNet and the state-of-the-art 2D image-based method CENet [3] on the SemanticKITTI test set. The visualized challenging autonomous driving scenes include urban, rural, and complex intersection scenes. The predictions projected on the corresponding RGB images are also illustrated. In addition, we use the same color boxes to point out the same objects in the point clouds and images for each scene. Meanwhile, we provide the zoomed-in view of some boxed objects for clear visualization. Moreover, the reference from point color to the semantic class in the predictions is shown at the bottom of the figure.

10
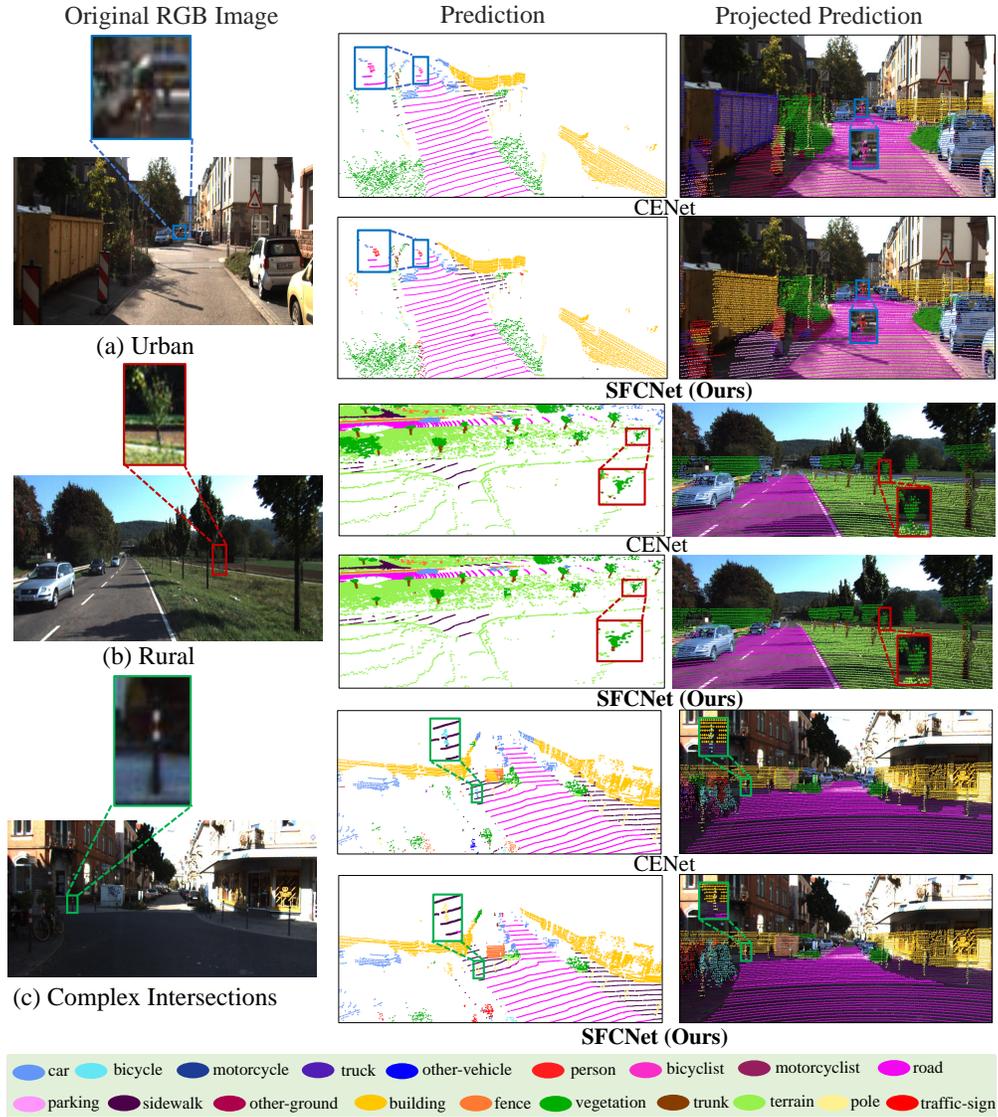
Figure 5: More Qualitative Results on Semantic Segmentation on SemanticKITTI Test Set. This figure shows more qualitative results of our SFCNet and the state-of-the-art 2D image-based method CENet [3] on the urban, rural, and complex intersection scenes of the SemanticKITTI test set. As in Fig. 4, the predictions projected on the corresponding RGB images are also illustrated. In addition, the same color boxes are adopted to point out the same objects in the point clouds and images for each scene. Meanwhile, the zoomed-in view of some boxed objects is illustrated for clear visualization. Moreover, the reference from point color to the semantic class in the predictions is shown at the bottom of the figure.

11

| | barrier | | bicycle | | bus | | car | | construction-vehicle | | motorcycle | | pedestrian | | traffic-cone |
| | trailer | | truck | | driveable-surface | | other-flat | | sidewalk | | terrain | | manmade | | vegetation |

Figure 6: More Qualitative Comparison of Semantic Segmentation on NuScenes Validation Set. We show more comparisons between our SFCNet and the state-of-the-art 2D image-based method RangeViT [15] on the nuScenes dataset. The predictions projected on the corresponding RGB images are also illustrated. In addition, we use the same color boxes to point out the same objects in the point clouds and images for each scene. Meanwhile, we provide the zoomed-in view of some boxed objects for clear visualization. Moreover, the reference from point color to the semantic class in the predictions and ground truths is shown at the bottom of the figure.

12

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9297–9307, 2019.

[2] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[3] Hui-Xian Cheng, Xian-Feng Han, and Guo-Qiang Xiao. Cenet: Toward concise and efficient lidar semantic segmentation for autonomous driving. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 01–06. IEEE, 2022.

[4] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[7] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. RangeNet ++: Fast and Accurate LiDAR Semantic Segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220. IEEE, November 2019.

[8] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9939–9948, 2021.

[9] Dan A Alcantara, Vasily Volkov, Shubhabrata Sengupta, Michael Mitzenmacher, John D Owens, and Nina Amenta. Building an efficient hash table on the gpu. In *GPU Computing Gems Jade Edition*, pages 39–53. Elsevier, 2012.

[10] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[11] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. 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*, pages 4413–4421, 2018.

[12] Deyvid Kochanov, Fatemeh Karimi Nejadasl, and Olaf Booij. Kprnet: Improving projection-based lidar semantic segmentation. *arXiv preprint arXiv:2007.12668*, 2020.

[13] Xin Lai, Yukang Chen, Fanbin Lu, Jianhui Liu, and Jiaya Jia. Spherical transformer for lidar-based 3d recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[14] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Learning semantic segmentation of large-scale point clouds with random sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[15] Angelika Ando, Spyros Gidaris, Andrei Bursuc, Gilles Puy, Alexandre Boulch, and Renaud Marlet. Rangevit: Towards vision transformers for 3d semantic segmentation in autonomous driving. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2023.

13