PointSIFT: A SIFT-like Network Module for 3D Point Cloud Semantic Segmentation

Mingyang Jiang **Yiran Wu** **Cewu Lu***

Shanghai Jiao Tong University, China

Abstract

Recently, 3D understanding research pays more attention to extracting the feature from point cloud [17, 19] directly. Therefore, exploring shape pattern description in points is essential. Inspired by SIFT [11] that is an outstanding 2D shape representation, we design a PointSIFT module that encodes information of different orientations and is adaptive to scale of shape. Specifically, an orientation-encoding unit is designed to describe eight crucial orientations. Thus, by stacking several orientation-encoding units, we can get the multi-scale representation. Extensive experiments show our PointSIFT-based framework outperform state-of-the-art method on standard benchmarking datasets. The code and trained model will be published accompanied by this paper.

1 Introduction

3D point cloud understanding is a long-standing problem. The tasks include 3D object classification [26], 3D object detection[6, 16, 21] and 3D semantic segmentation[17, 19, 20]. Among these tasks, 3D semantic segmentation is relatively fundamental and challenging. It aims to label all the points in 3D space, which meets many difficulties. For example, the sparseness of point cloud in 3D space makes the most of training operators inefficient. Additionally, the relationship between points is implicit and difficult to be represented. In retrospect of previous work, we can find several solutions. In [14], handcrafted voxel feature is used to get geometric relationship. In [15], 2D CNN features from both RGB and depth images (4 channels) are extracted. Though all the works above achieve convincing results, some problems are still unsolved. 2D convolution misses out the 3D geometry information (normal and shape), while 3D convolution requires heavy computation.

Recently, PointNet architecture [17] takes point clouds as input instead of voxels or mesh to extract features directly. It not only accelerates the computation but also notably improves the segmentation performance. In this paper, we adopt the scheme of PointNet[17], taking point clouds as input. The core problem is how to capture shape pattern of point inputs. When we revisit shape descriptors in the image domain, SIFT [11] is among the most successful ones and is widely used in image pattern representation. It is because SIFT considers two fundamental features of shape representation, namely orientation-encoding and scale-awareness. Specifically, scale-awareness means SIFT can select the most representative scale of targeting shape, while orientation-encoding can comprehensively perceive the pattern in different orientations. Therefore, we go down to explore a point cloud version SIFT-like architecture named as PointSIFT, including the design of orientation-encoding and scale-awareness. The main idea of PointSIFT is illustrated in Figure 1. Unlike SIFT, which is a handcrafted feature, our PointSIFT is a module of neural networks which can be optimized.

The basic block of PointSIFT is an orientation-encoding unit, which convolves the feature of nearest points in 8 orientations. In comparison to K-nearest searching in pointNet++ [19] that may make

*Cewu Lu is corresponding author lucewu@sjtu.edu.cn

Preprint. Work in progress.

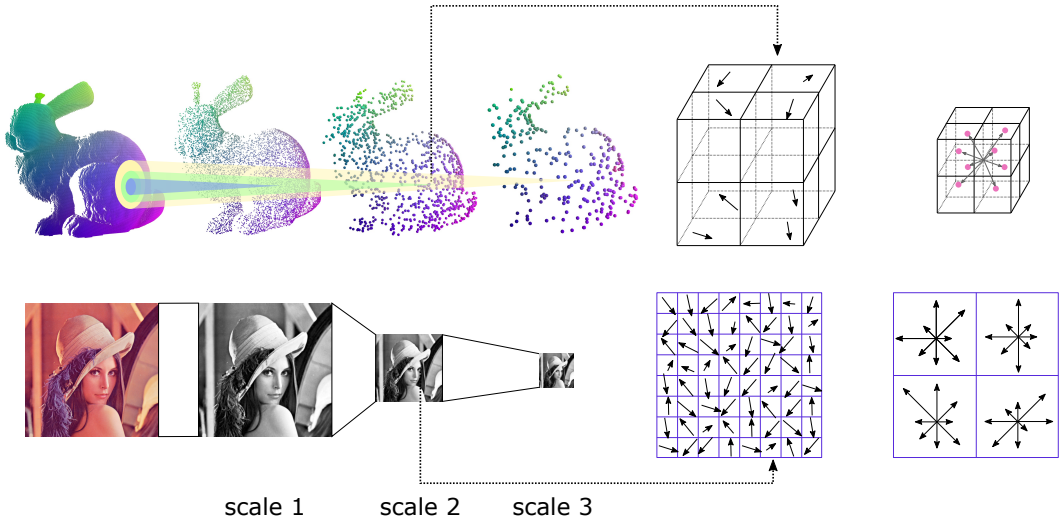

Figure 1: Structure of SIFT and PointSIFT. Both of them can capture multi-scale patterns and select the most representative scale. In each key point (pixel), orientational information is encoded.

K neighborhoods concentrate in one orientation, our orientation-encoding scheme can capture information of all orientations. We further stack several orientation-encoding (OE) units. In this way, the neurons in different stacked OE units can perceive different scales, making it scale-aware. Our scale-aware scheme is to put these OE units together by shortcut connections and let neural network (after training) select the appropriate scale. Given PointSIFT module, we build an end-to-end architecture like conventional segmentation framework [19], which has two stages (i.e., encoding (downsampling) and decoding (upsampling)). The PointSIFT module is inserted in each layer of the whole framework. It significantly improves the presentation ability of the network. Extensive results show that our PointSIFT-based method outperforms the state-of-the-art methods on S3DIS[1] and ScanNet[5] datasets.

2 Related Work

We investigate some 3D representation methods and point cloud segmentation architectures. After that, we briefly survey SIFT descriptor.

2.1 3D representation

Volumetric Representation The methods in [26, 13, 18] attempt to voxelize scene through 3D CNN. However, the main challenges in volumetric representation are spatial sparseness and computation complexity. Therefore, in [20, 10, 24], researchers try to solve the sparsity problems with special methods (e.g., OcTree structure). However, it still takes much time converting point clouds to voxels.

Polygonal Meshes Some research works [4, 27, 12] have focused on the use of spectral convolutional neural network for meshes. In spectral domain, Fourier transformation converts convolution to multiplication. However, this kind of methods is confined to manifold meshes.

Multi-view Representation Researchers seek to explore the multi-view relationship in the same scene [23, 18, 22]. However, these inputs neglect a large number of geometry details. It could substantially limit the performance in some particular tasks such as shape completion and semantic segmentation.

Point Clouds Recently, point cloud input methods [17, 19] propose a new architecture taking point clouds as input directly. [9] uses $\mathcal{X}$-Conv layer instead of vanilla mlp (Multi-layer Perceptron) layer. [8] performs geometric partitioning to build a superpoint graph. Then superpoints are fed into

PointNet to get embeddings. However, these methods still have room for improving in basic network block design to advantage shape pattern representation.

2.2 Point Clouds Segmentation

Point clouds segmentation task can be stated as follows: given a set of points in 3D space, we try to assign a category label to each point. PointNet[17] uses T-Net to balance unordered point clouds before extracting features with fully connected layers. [19] proposed a hierarchical neural network to capture local geometric details. Superpoint Graphs[8] (SPG) takes unsupervised method: Cut-pursuit algorithm[7] for point partition. All the point clouds are built into a superpoint graph. Each superpoint will be embedded by a PointNet.

2.3 SIFT feature

The scale-invariant feature transform (SIFT)[11] is a widely used handcrafted algorithm to describe the local feature in 2D images. SIFT and its variants[3] take several steps to extract features. This kind of descriptor can encode the orientation information and select well-suited scale. In this paper, we adopt their design principle in our network design.

3 PointSIFT-based Architecture

In this section, we present our architecture for point cloud segmentation. Our core module is a PointSIFT model that can select the appropriate scale and encode orientational information like SIFT description. We firstly present PointSIFT module with in-depth discussion (in Section 3.1), thus introduce the end-to-end architecture based on this core module (in Section 3.2).

3.1 PointSIFT Description Module

We present our core PointSIFT description module in this section. SIFT is a descriptor that widely used in image pattern representation, which is invariant to scale and orientation. Similar to SIFT representation, given a point p, our PointSIFT module computes the description of a set of point cloud centered at p. Different with SIFT feature that is a handcrafted design, the parameters of our module is free to be optimized by our end-to-end network architecture.

Overview Given of input n features (with dimension d) representing a point cloud, PointSIFT module outputs n features (with dimension d) corresponding to input features. Our module have several stacked units as Figure 2 illustrated. Different units (excluding the last units) represent different scales and the final layer summarizes information about them by shortcut connections to enable scale-aware representation. The single unit is an orientation-encoding convolution processing. In what follows, we present single unit structure (orientation-encoding convolution) and multi-unit construction (scale-aware representation).

3.1.1 Orientation-encoding Convolution

To robustly capture shape pattern, we hope shape information in different orientations can be explicitly encoded. Therefore, we propose orientation-encoding convolution that is operated on all points. Given point p_0 with feature f_0, the 3D space is partitioned centered at p_0 into 8 octants (sub-regions) that indicates 8 different orientations like Figure 4 (a). In each octant, we search the nearest point of p_0 and take its feature to represent this octant. Since some points far away from p_0 are useless to represent p_0. In an octant, if we can not find any point cloud inside searching radius r, feature f_0 is used to represent this octant.

To make convolution orientation-aware, we perform 3-stage convolution along three axes namely X, Y, and Z. We put features of searched points into a tensor $M \in R^{2\times2\times2\times d}$, where the first 3 dimensions indicate 8 octants. For example $(1, 1, 1)$ indicate the feature of top-front-right octant. As

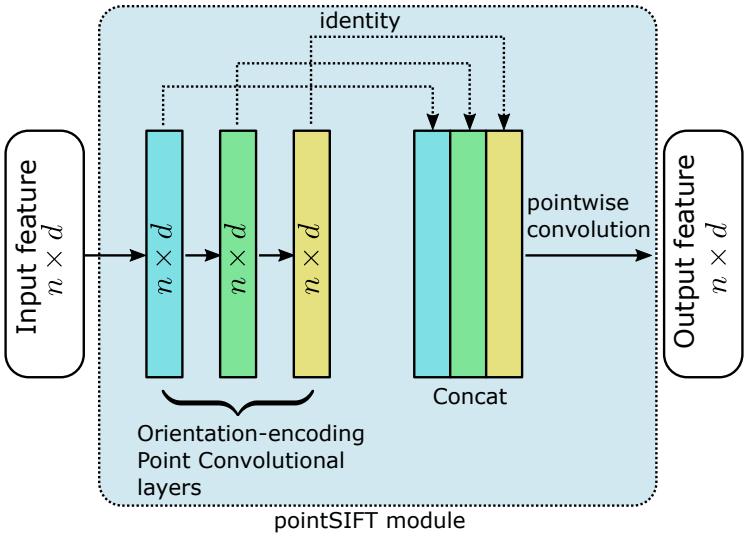

Figure 2: Module architecture. The input features are put into stacks of Orientation-encoding Point Convolutional layers. Then all the outputs are concatenated. Output features can be reached after point-wise convolutional layer.

shown in Figure 4 (c), three stages of directional convolution are:

$$\begin{aligned} M_1 &= g[Conv_x(A_x, M)] \in R^{2\times 2\times d} \\ M_2 &= g[Conv_y(A_y, M_1)] \in R^{2\times d} \\ M_3 &= g[Conv_z(A_z, M_2)] \in R^{1\times d} \end{aligned} \tag{1}$$

Where A_x, A_y and A_z are convolution weights to optimize, and $Conv_x$ ($Conv_y$,$Conv_z$) is convolution along X (Y,Z) direction. In this paper, we set $g[\cdot] = ReLU(Batch\_norm(\cdot))$. After our orientation-encoding convolution, each point would be represented by a d dimension vector. This vector represents the shape pattern around p_0 in an orientation-encoding manner.

Discussion Our point convolution scheme is very efficient and easy to implement. For each point, the computation complexity is the same as ball query searching. The space complexity is less than the method in [19] since we need to search one neighborhood only. In comparison to random point selection, our scheme can extract more reliable and stable representative points. For K nearest neighbors selection, it may meet a challenge that all K nearest neighbors concentrate in a small region and degrades representation ability (seeing Figure 3). By contrast, our method can capture the shape pattern in different orientations.

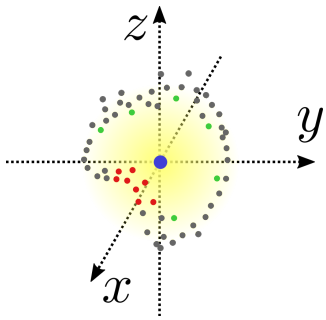

Figure 3: In this case, using K nearest neighbors, all chosen points are from one direction (red points). If we select points in different directions (green points), the representation ability will be better.

Scale-aware Architecture Taking orientation-encoding above as basic unit, we build the scale-aware network architecture. A single orientation-encoding unit captures a small scale (8 neighborhoods) only. But if we stack several orientation-encoding units to develop a deeper network, the last one can observe a large scale 3D region as illustrated in Figure 1 and different stacked units represent different scales. Ideally, in the i^{th} stacked unit, a neuron can observe 8^i points. Therefore, we have an opportunity to select an appropriate scale. Our strategy is to concatenate output of different stacked unit by a shortcut. Thus, we leave scale selection to overall network optimization. To optimize performance, the network should select the scale being well-suited to targeting shape.

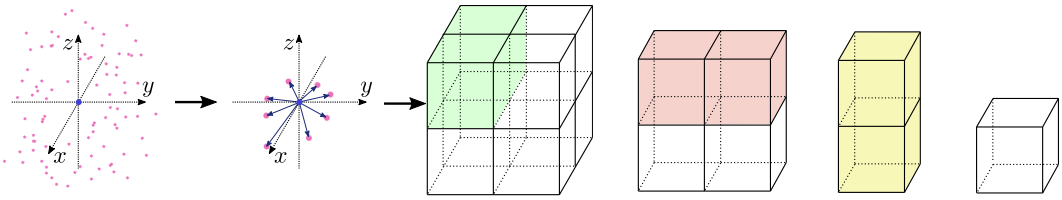

Figure 4: Illustration of the details in Orientation-encoding Point Convolutional layer. (a): point clouds in 3D space. (b) neighbors in eight directions. (c) three stages convolution combines all the features.

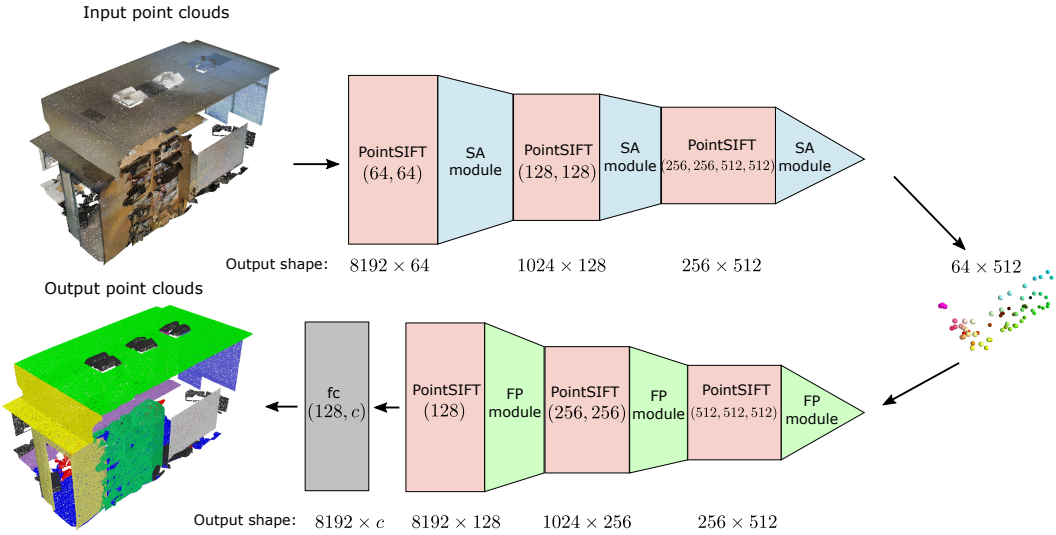

Figure 5: Illustration of end-to-end network architecture. The network contains two part: Encoder (Set Abstraction) and Decoder (Feature Propagation). PointSIFT modules (red color) are inserted between all adjacent layers of Set Abstraction (blue color) and Feature Propagation module (green color). Both set Abstraction module and Feature Propagation module are introduced in [19]. The encoder and the decoder are linked by several skip connections.

Discussion Similar to the principle of SIFT scale-invariant design, we keep different scales thus select the most representative one. Of course, the network may not just choose a single scale, but sparsely attend to few of scales. More details can be referred in Sec 4.1 to show our scale-aware point convolution can perceive object scale indeed.

3.2 Overall Architecture

Based on proposed PointSIFT module, we build overall architecture as Figure 5 shown. Similar to general semantic segmentation network[2], our architecture consists of two parts: encoder and decoder. The input is the 3D coordinates (or concatenating with RGB value) of $n = 8192$ points. Following [19], multi-layer perceptron (MLP) is used to transform the input 3D (or 6D) vectors into $d = 64$ dimension features. In the encoder part, SA (set abstraction) module from [19] will also be applied. In this module, we take three steps of downsampling operations, the point number in three steps are $n_1 = 1024$, $n_2 = 256$, $n_3 = 64$ respectively. For decoder part, we use FP(feature propagation) module in [19] to perform upsampling operations, the point number in three steps are $n_4 = 256$, $n_5 = 1024$, $n_6 = 8192$ respectively. Our PointSIFT module is inserted between all adjacent encoder and decoder layers. Finally, all the features in the last decoder layer go through a fully connected layer and convert into the probability of each category. Note that PointSIFT is a general add-on module for various architectures.

Table 1: Experiment on ScanNet[5] dataset. The first row is the size of point cloud in each downsampling step. The second and third rows are the point joining convolution computation (in average) by Pointnet++ and our PointSIFT frameworks respectively.

point cloud size	8192	1024	256	64
captured size of Pointnet++[19]	6570	1010	255	64
captured size of PointSIFT framework	8192	1024	256	64

Table 2: Architectures for comparison of different sampling methods. After ball query sampling, point-wise convolution takes 32×1 kernels for extracting features.

layer name	output size	baseline model	ball query sampling	PointSIFT sampling
conv_1	1024×128	SA module		
conv_2	256×256	SA module	ball query sampling point-wise convolution SA module	PointSIFT module (128, 128) SA module
conv_3	64×512	SA module	ball query sampling point-wise convolution SA module	PointSIFT module (256, 256) SA module
pf_conv_3	256×512	FP module	FP module ball query sampling point-wise convolution	FP module PointSIFT module (512, 512)
pf_conv_2	1024×256	FP module	FP module ball query sampling point-wise convolution	FP module PointSIFT module (256, 256)
pf_conv_1	8192×128	FP module		
fc	8192×21	fully connected layer		

4 Experiments

Our experiment includes two parts: verifying the effectiveness of PointSIFT module in section 4.1 and introducing the improvement of pixel labeling task in section 4.2. In what follows, SA and FP indicate Set Abstraction (in encoder part) and Feature Propagation (in decoder part) module respectively, which are defined in [19].

4.1 PointSIFT Analysis

Effectiveness of PointSIFT Module We insert PointSIFT module before each layer of SA processing. It can avoid information loss in downsampling processing. We take the conventional operation of PointNet++[19] as an example. Given input size of 8192, the first step is downsampling 8192 points to 1024 points. The method of [19] select 1024 centroids and convolve 32 nearest points inside the searching radius. We found 1622 points on average don't join the convolutional computation. That is, information of about 20% points are not integrated into the downsampled representation. By contrast, our PointSIFT module involves more points in computation by performing multi-layer convolution. By inserting PointSIFT module before each downsampling layer, we found all points can be processed and contribute their information to final prediction. The statistics of points being processed is reported in Table 1. It shows that our PointSIFT module helps to extract more information from original inputs.

4.2 Results on semantic segmentation task

Stacked 8-neighborhood searching vs ball query searching To represent the shape pattern centered at a point, the points around it should be searched. We adopt stacked 8-neighborhood searching scheme, which is fundamentally different from ball query searching proposed in PointNet++ [19]. There are two main differences between stacked 8-neighborhood (S8N) searching and ball query

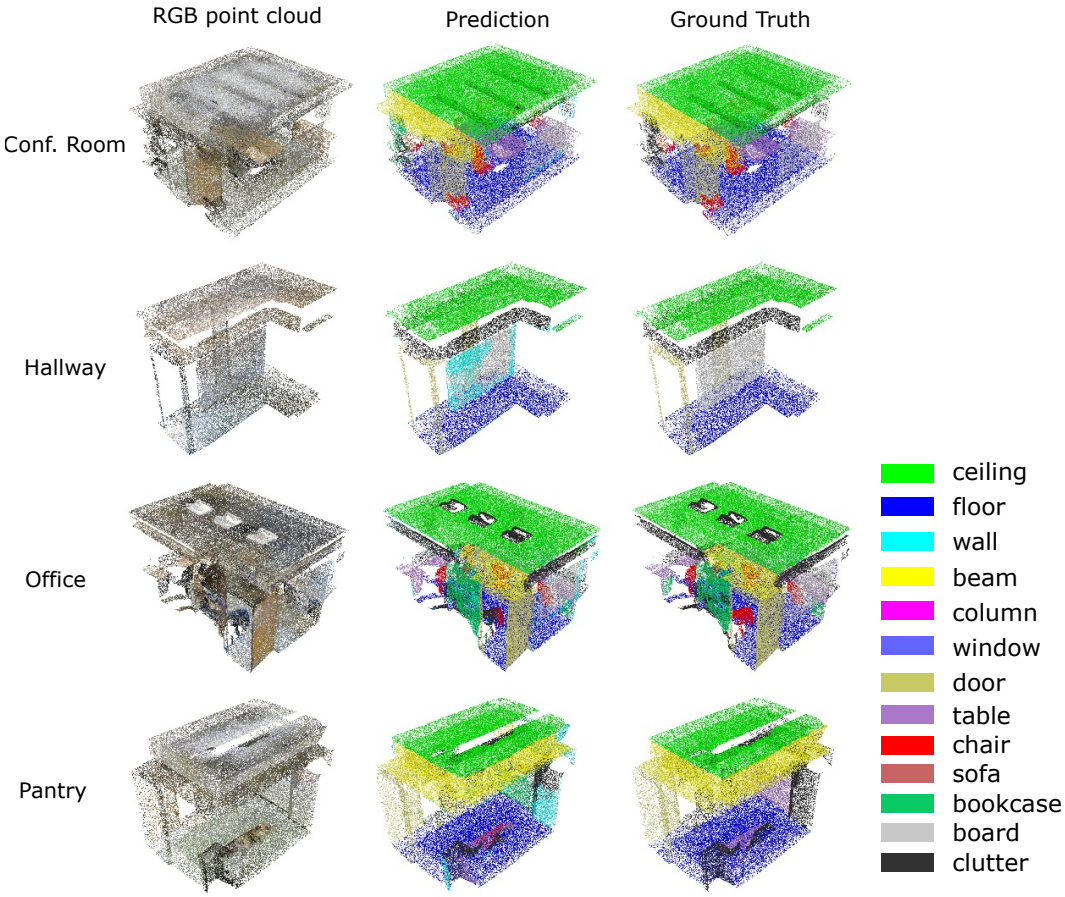

Figure 6: visualizations on S3DIS dataset[1]

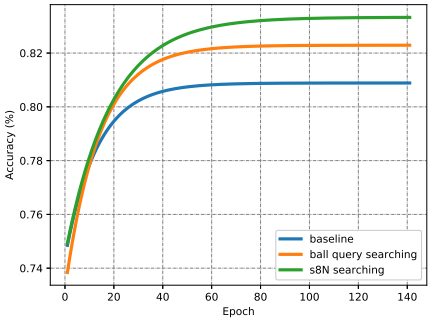

Figure 7: Accuracy on ScanNet[5] for different searching methods. All the lines are fitted into exponential function.

Table 3: ScanNet[5] label accuracy and mIoU

Method	Accuracy %	mean IoU
3DCNN[5]	73.0	-
PointNet[17]	73.9	-
PointNet++[19]	84.5	38.28
PointCNN[9]	85.1	-
Ours	**86.0**	**41.5**

searching: (1) S8N searching divides the space into 8 parts by space coordinate axis, but ball query searching uses whole area. (2) S8N searching takes the nearest points for each subspace. Ball query searching selects points randomly within the searching radius.

To figure out the performance, we build relative smaller networks as shown in Table 2. We stack three 8-neighborhood convolution as presented in section 3.1.1. This setting covers $25 \sim 35$ points and is closed to 32 points of ball query searching in [19]. The performances are compared in Fig 7. It verified that our method of selecting neighbor points by dividing space turns out to be more efficient.

Table 4: Overall accuracy and meaning intersection over union metric of S3DIS[1] dataset.

Method	Overall Accuracy (%)	mean IoU (%)
PointNet[17]	78.62	47.71
SegCloud[25]	-	48.92
SPGraph[8]	85.5	62.1
PointCNN[9]	-	62.74
Ours	**88.72**	**70.23**

Table 5: IoU for all categories of S3DIS[1] dataset.

Method	ceiling	floor	wall	beam	column	window	door	chair	table	bookcase	sofa	board	clutter
PointNet[17]	88.0	88.7	69.3	42.4	23.1	47.5	51.6	42.0	54.1	38.2	9.6	29.4	35.2
SegCloud[25]	90.06	96.05	69.86	0.00	18.37	38.35	23.12	**75.89**	70.40	58.42	40.88	12.96	41.60
SPGraph[8]	89.9	95.1	76.4	**62.8**	**47.1**	55.3	68.4	73.5	69.2	**63.2**	45.9	8.7	52.9
Ours	**93.7**	**97.9**	**87.5**	59.3	31.0	**73.7**	**80.7**	75.1	**78.7**	40.8	**66.3**	**72.2**	**65.1**

Effectiveness of Scale Selection To verify the effectiveness of scale selection, we generate 10000 scale point shapes (e.g., circle, rectangle). Then we train segmentation framework on them by our PointSIFT framework. As aforementioned, different layers in PointSIFT module represents different scales. We can know the estimated scale by checking which layer has highest activation energy. Thus, estimated scale by PointSIFT module and ground truth scale are compared to compute the accuracy which is 89%. It shows that proposed PointSIFT framework can select an appropriate scale nicely.

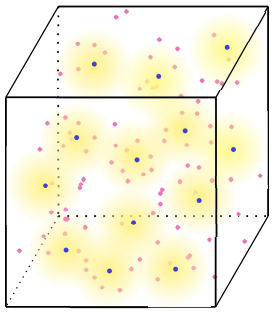

Figure 8: Blue points represent centroids; Yellow clouds represent the ball searching area. In this case, many pink points are not covered by yellow clouds. Because of random selection, some points within yellow clouds may also not be chosen.

Stanford Large-Scale 3D Indoor Spaces The S3DIS dataset[1] takes six folders of RGB-D point clouds data from three different buildings (including 271 rooms). Each point is annotated with labels from 13 categories. We split it into training and testing set following the way of [17] with k-fold strategy. Overall Accuracy and mIoU are shown in Table 4. We can see that our PointSIFT architecture outperforms other methods. IoU of all the categories are shown in Table 5, our method improves remarkably on results of semantic segmentation task for this dataset and wins in most of the categories. Our method can achieve great results in some hard categories that other methods perform poorly, *i.e.* nearly 11 mIoU points on the sofa and 42 mIoU points on board. Some results are visualized in Fig 6.

ScanNet ScanNet[5] is another semantic scene labeling task with a total of 1513 scanned scenes. We follow [19, 9] and take 1201 scenes in training phase and 312 in testing phase without RGB information. We report the result in Table 3. Compared with other methods, our proposed PointSIFT method achieves better performance on the accuracy of per-voxel basis. Moreover, our method outperforms PointNet++ even though we do not use multi-scale grouping (MSG) in SA module. That is, the PointSIFT module shows advantages in single scale points searching and grouping.

5 Conclusion

We have proposed a novel PointSIFT module and demonstrated a significant improvement in semantic segmentation task on standard datasets. It verifies the advantages of our design. First, orientation-encoding units capture information of different orientations. Second, the network can automatically select appropriated scales from multi-scale units and put them together. Our PointSIFT module can be easily incorporated into different point input networks. Our future work is to apply our pointSIFT into 3D instance segmentation which is a more challenging task.

References

[1] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[3] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *In ECCV*, pages 404–417, 2006.

[4] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2013.

[5] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[6] J. Lahoud and B. Ghanem. 2d-driven 3d object detection in rgb-d images. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4632–4640, Oct 2017.

[7] Loic Landrieu and Guillaume Obozinski. Cut Pursuit: fast algorithms to learn piecewise constant functions on general weighted graphs. *SIAM Journal of Imaging Sciences*, Vol. 10(No. 4):pp. 1724–1766, 2017.

[8] Loïc Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. *CoRR*, abs/1711.09869, 2017.

[9] Y. Li, R. Bu, M. Sun, and B. Chen. PointCNN. *ArXiv e-prints*, January 2018.

[10] Yangyan Li, Soeren Pirk, Hao Su, Charles R Qi, and Leonidas J Guibas. Fpnn: Field probing neural networks for 3d data. *arXiv preprint arXiv:1605.06240*, 2016.

[11] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[12] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 832–840, Dec 2015.

[13] D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *IROS*, 2015.

[14] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter. Voxel cloud connectivity segmentation - supervoxels for point clouds. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2027–2034, June 2013.

[15] Trung Pham, Thanh-Toan Do, Niko Sünderhauf, and Ian D. Reid. Scenecut: Joint geometric and object segmentation for indoor scenes. *CoRR*, abs/1709.07158, 2017.

[16] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. *arXiv preprint arXiv:1711.08488*, 2017.

[17] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016.

[18] Charles R Qi, Hao Su, Matthias Niessner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. *arXiv preprint arXiv:1604.03265*, 2016.

[19] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[20] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[21] Shuran Song and Jianxiong Xiao. Deep Sliding Shapes for amodal 3D object detection in RGB-D images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[22] H. Su, F. Wang, E. Yi, and L. Guibas. 3d-assisted feature synthesis for novel views of an object. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2677–2685, Dec 2015.

[23] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[24] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[25] Lyne P. Tchapmi, Christopher B. Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. *CoRR*, abs/1710.07563, 2017.

[26] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, June 2015.

[27] Li Yi, Hao Su, Xingwen Guo, and Leonidas Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. *arXiv preprint arXiv:1612.00606*, 2016.