LFNet: Local Rotation Invariant Coordinate Frame for Robust Point Cloud Analysis

Hezhi Cao, Ronghui Zhan, Yanxin Ma, Chao Ma, and Jun Zhang

***Abstract*—Deep neural networks have achieved great progress in 3D scene understanding. However, recent methods mainly focused on objects with canonical orientations in contrast with random postures in reality. In this letter, we propose a hierarchical neural network, named Local Frame Network (LFNet), based on the local rotation invariant coordinate frame for robust point cloud analysis. The local point patches in different orientated objects are transformed into an identical distribution based on this coordinate frame, and the transformed coordinates are taken as input features to eliminate the influence of rotations at the *input level*. Meanwhile, a discrete convolution operator is defined in the constructed coordinate frame to extract rotation invariant features from local patches, which can further remove the influence of rotations at the *convolution level*. Moreover, a Spatial Feature Encoder (SFE) module is utilized to perceive the spatial structure of the local region. Mathematical analysis and experimental results on two public datasets demonstrate that the proposed method can eliminate the influence of rotations without data augmentation and outperforms other state-of-the-art methods.**

***Index Terms*—Point cloud, rotation invariance, local coordinate frame, local point patches**

I. Introduction

THREE-DIMENSIONAL computer vision has been playing an important role in many real-world applications, e.g., robotics, augmented reality, and autonomous driving. Since data captured by 3D sensors such as LiDAR or Microsoft Kinect are point clouds, directly processing point clouds becomes popular. Pioneered by PointNet [1] and PointNet++ [2], these methods [3–10] have achieved remarkable results in point cloud learning and shape analysis.

However, there remains a fundamental problem that most previous methods do not allow the input point clouds to be rotated. The objects in these methods are mainly in a canonical orientation. In real applications, these methods degrade rapidly since objects and scenes captured by sensors are usually at unknown postures. An intuitive solution to analyze rotated shapes is to augment the training data with tremendous rotations. Although data augmentation is effective to some extent, it induces expensive computations in the training phase. Meanwhile, it lacks a solid guarantee of rotation invariance since SO(3)[1] is an infinite group, which means there are infinite rotated clones of the object to be sampled. To better learn from rotated shapes, a spatial transformer network [11] was applied in [1, 4] to canonicalize the input data before feature extraction. It improves the rotation-robustness of the model but still suffers from the drawbacks of data augmentation.

As the spatial distribution of input points relies on the orientation of the object, operations related to this spatial distribution will be affected by rotations. It is observed that these operations mainly exist in two levels, the first one is *input level* where the input features include the original coordinates of each point with or without additional color, normals, etc. Since the coordinates vary under different rotations, the results will be affected by rotations. Some methods addressed this issue by proposing manually defined features. RIConv [12] represents the neighboring points by the relative distance and angles in the local regions. PRIN [13] uses density aware adaptive sampling to transform input points into spherical voxel signals. Spherical-CNN [14] utilizes spherical correlation to map spherical inputs to features. However, these methods are either too naive or too sophisticated for obtaining input features compared with coordinates. The second one is the convolution level where the existing convolution operations implemented on point clouds are mostly defined with the spatial distribution of local point patches. Hence, obtaining rotation invariant features from the local point patches is difficult for these operators as the spatial distribution changes with rotations, as depicted in Fig. 1(b). To make the convolution operation free of the dependency between the spatial distribution of neighboring points and the orientation of object, RIConv [12] partitions the local region firstly by the reference point and centroid of the local region, and then a 1D convolution is conducted to obtain the local features. SrinNet [15] learns a local transformation matrix to permute the local features. Spherical-CNN [14] proposes spherical convolution which is invariant to rotations. However, these methods do not exploit the relative distribution information adequately since the original locations of neighboring points cannot be retrieved. In this letter, we construct a local rotation invariant coordinate frame for each reference point to remove the dependency between the spatial distribution of neighboring points and the orientation of the object. Based on the constructed coordinate frame, we further eliminate the influence of rotation at both *input level* and *convolution level* to achieve rotation invariance of the model. Besides, Spatial Feature Encoder (SFE) modules [16] are integrated to exploit the spatial structure of transformed neighboring. We compare LFNet with various state-of-the-art approaches on the benchmark dataset: ModelNet40 [17] and ShapeNet-Part [18]. Experimental results show that our approach has achieved the state-of-the-art performance with

Manuscript received November 19, 2020; revised December 12, 2020; accepted December 21, 2020. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Wei Li. *(Corresponding author: Ronghui Zhan)*

Hezhi Cao and Ronghui Zhan are with National University of Defense Technology, ChangSha, China (hezhicao@nudt.edu.cn, zhanrh@nudt.edu.cn).

[1]SO(3) is the space of all 3D rotations in R^3

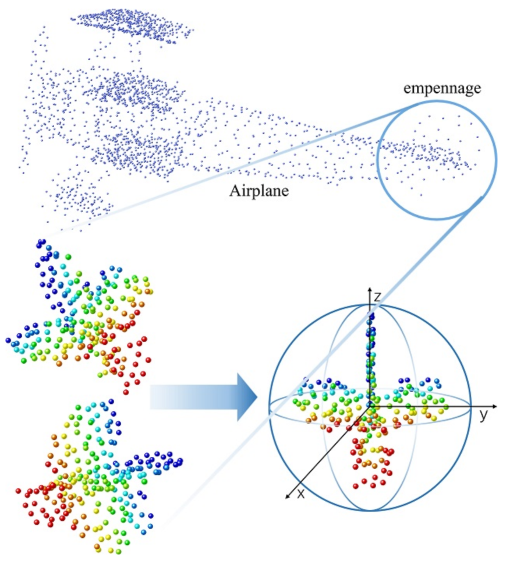

Fig. 1. Rotation invariant transformation of local point patches. Top row: point clouds of an airplane. Bottom left: local point patches in the different oriented airplanes. Since current convolution operations implemented on point clouds are related to the spatial distribution of neighboring points, it is hard for these operators to obtain an identical output features from these two local point patches. Bottom right: rotation invariant distribution of local patches in the constructed local coordinate frame. Different color denotes different input features. By transforming the local point patches into a rotation invariant distribution, the extraction of rotation invariant features is much easier.

the existence of random SO(3) rotations.

The key contributions of our work include: (1) A local rotation invariant coordinate frame is proposed, in which the local patches in different orientated objects can be transformed into a rotation invariant distribution. The transformed coordinates of local patches are taken as input features to eliminate the influence of rotations at *input level*. (2) A local discrete convolution operator is proposed and defined with the transformed neighboring points to extract local rotation invariant features at *convolution level*. Meanwhile, an additional branch with SFE modules is integrated to exploit the spatial structure of transformed neighboring points and combines the output spatial features with input features at each stage of the network progressively. (3) Extensive experiments on point cloud classification and segmentation tasks are conducted and demonstrate the impressive performance of the proposed method, when compared with the state-of-the-art methods under the existence of random SO(3) rotations.

II. Methods

Since rotations will change the spatial distribution of input points and affect the model in two levels: *input level* and *convolution level*, we construct a local rotation invariant coordinate frame to transform the neighboring points into a rotation invariant distribution. The transformed coordinates of local points are taken as input features to eliminate the influence of rotations at *input level*. A discrete convolution operator is defined with the transformed neighboring points subsequently to exploit local features and avoid the influence of rotations at *convolution level*. To further enrich the spatial information, an additional spatial encoding branch composed of SFE modules is utilized to perceive the spatial structure of neighboring points.

A. Construction of Rotation Invariant coordinate Frame

We construct the rotation invariant coordinate frame by the normal $\boldsymbol{n_i}$ of reference point $\boldsymbol{p_i}$ and the normalized point vector $\boldsymbol{p_i}/\left\|\boldsymbol{p_i}\right\|$ as:

$$\begin{aligned} \boldsymbol{e_x} &= (\boldsymbol{p_i}/\left\|\boldsymbol{p_i}\right\|) \times \boldsymbol{n_i} \\ \boldsymbol{e_y} &= \boldsymbol{e_x} \times \boldsymbol{n_i} \\ \boldsymbol{e_z} &= \boldsymbol{n_i} \end{aligned} \tag{1}$$

Where $\times$ denotes cross product. The transformed coordinates $\boldsymbol{p'_j} = (x'_j, y'_j, z'_j)^T$ of neighboring points $\boldsymbol{p_j}$ can be further calculated as:

$$\begin{aligned} x'_j &=< \boldsymbol{p_j} - \boldsymbol{p_i}, \boldsymbol{e_x} >= (\boldsymbol{p_j} - \boldsymbol{p_i})^T(\boldsymbol{p_i} \times \boldsymbol{n_i})/\left\|\boldsymbol{p_i}\right\| \\ y'_j &=< \boldsymbol{p_j} - \boldsymbol{p_i}, \boldsymbol{e_y} >= (\boldsymbol{p_j} - \boldsymbol{p_i})^T((\boldsymbol{p_i} \times \boldsymbol{n_i}) \times \boldsymbol{n_i})/\left\|\boldsymbol{p_i}\right\| \\ z'_j &=< \boldsymbol{p_j} - \boldsymbol{p_i}, \boldsymbol{e_z} >= (\boldsymbol{p_j} - \boldsymbol{p_i})^T\boldsymbol{n_i} \end{aligned} \tag{2}$$

Unlike handcraft features in [12–14], this transformation can be easily implemented and preserve the relative positional relationships among neighboring points. In the perspective of matrix form, obtaining the transformed coordinates corresponds with a linear transformation matrix $T_L = [\boldsymbol{p_j} \times \boldsymbol{n_i}/\|\boldsymbol{p_i}\|, (\boldsymbol{p_i} \times \boldsymbol{n_i}) \times \boldsymbol{n_i}/\|\boldsymbol{p_i}\|, \boldsymbol{n_i}]$. The transformed coordinates of $\boldsymbol{p_j}$ can be obtained by $\boldsymbol{p'_j} = (\boldsymbol{p_j} - \boldsymbol{p_i})^T T_L$ subsequently. The proof of rotation invariance of this transformation and the special case are detailed in the supplementary material.

B. Discrete Convolution Operator

By taking the transformed coordinates of local patches as input features, the influence of rotations to the model at *input level* can be eliminated. To make convolution operation free of the spatial distribution change of local patches incurred by rotations, we propose a discrete convolution operator defined with transformed local patches. The discrete convolution operator first transforms the local point patches formed by K-NN of reference point $\boldsymbol{p_i}$ into a rotation invariant distribution to remove the influence of rotations at *convolution level*. Then, a set of kernel points obtained by the optimization function in [7] are utilized to help the extraction of local features. An interpolation function $\phi(\boldsymbol{p'_j}, \boldsymbol{p_\kappa})$ is utilized to obtain the features at the locations of kernel points. The interpolation function $\phi(\cdot, \cdot) : R^3 \times R^3 \in R$ takes a transformed neighboring point $\boldsymbol{p'_j}$ and a kernel point $\boldsymbol{p_\kappa}$ as input, and measures their relations by explicit geometric distance. A normalization term $|N_{\boldsymbol{p_\kappa}}|$ is added for each kernel point to achieve sparsity invariant. The sampled feature $f(\boldsymbol{p_\kappa})$ is:

$$f(\boldsymbol{p_\kappa}) = \frac{1}{|N_{\boldsymbol{p_\kappa}}|} \sum_{\boldsymbol{p'_j} \in N_{\boldsymbol{p_\kappa}}} \phi(\boldsymbol{p'_j}, \boldsymbol{p_\kappa}) f(\boldsymbol{p_j}) \tag{3}$$

Since $\phi(\cdot, \cdot)$ is the correlation between $\boldsymbol{p_\kappa}$ and $\boldsymbol{p'_j}$, it should be higher when $\boldsymbol{p'_j}$ is closer to $\boldsymbol{p_\kappa}$. Inspired by the bilinear interpolation in [19], we use the trilinear interpolation function:

$$\begin{aligned} \phi(\boldsymbol{p'_j}, \boldsymbol{p_\kappa}) =& (1 - \left|x'_j - x_\kappa\right|/\sigma)(1 - \left|y'_j - y_\kappa\right|/\sigma) \\ & (1 - \left|z'_j - z_\kappa\right|/\sigma) \end{aligned} \tag{4}$$

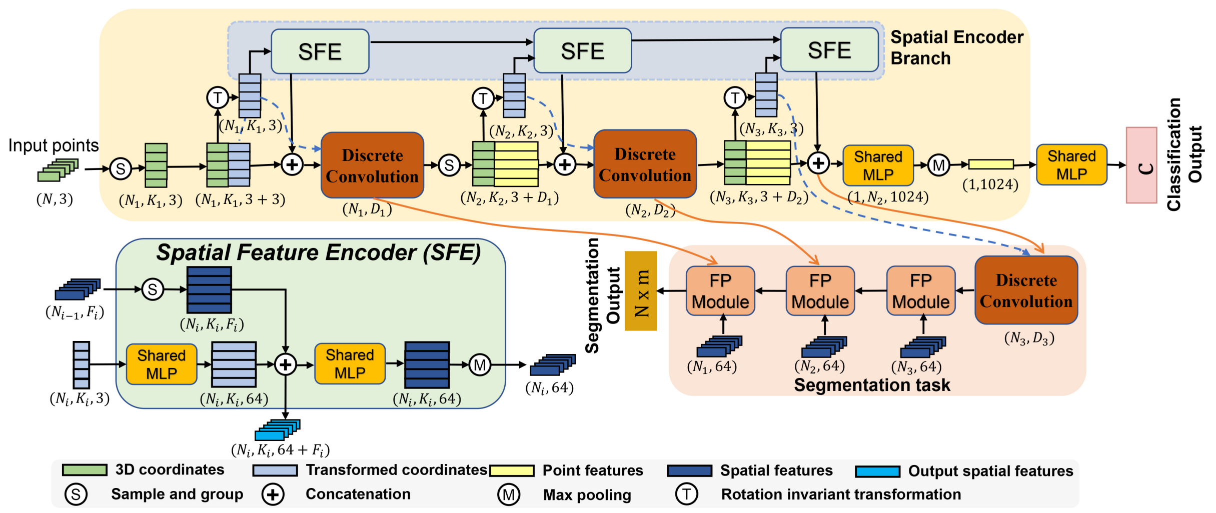

Fig. 2. The architecture of LFNet. The porposed network transforms the local point patches into a rotation invariant distribution at each stage. The tranformed coordinates of neighboring points are taken as input features at *input level* and help the definition of discrete convolution operator in *convolution level* (dashed arrows mean no backpropagation during training). The details of discrete convolution are shown in Fig. 3. A SFE module is applied perceive the spatial structure at each stage. In which the transformed coordinates of neighboring points are taken as input and lifted by shared MLPs. The spatial features from SFE module are concatenated with input features and sent to a shared MLP for the preparation of next SFE. FP modules from [2] are utilized to propagate features to the previous layer. Note that F_i stands for the dimension of spatial features from previous SFE module which is 0 for the first layer, 64 for others.

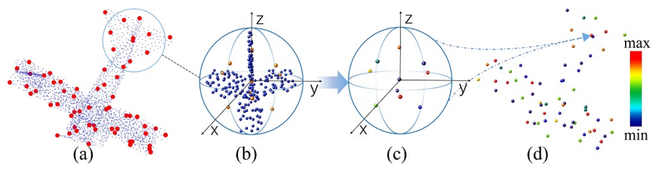

Fig. 3. Illustration of discrete convolution operation. Red points in (a) denote reference points sampled by FPS algorithm and the local point patches are formed by their k nearest neighbors. Since the spatial distribution change of local point patches make the convolution operators vulnerable to rotations, we first transform the local patches into a rotation invariant distribution as depicted in (b) to makes it easier to define a rotation invariant convolution operator. Orange points in (b) indicate the kernel points defined with the tranformed neighboring points. The corresponding features of kernel points are obtained by an interpolation function subsequently. The interpolated features of kernel points are passed to the output feature associated with reference point in (d) by a typical convolution which assigns each kernel point with a weight matrix.

where $\boldsymbol{p_\kappa} = (x_\kappa, y_\kappa, z_\kappa)$ are the coordinates of kernel point $\boldsymbol{p_\kappa}$ in the constructed local coordinate frame. σ is the influence distance of kernel points and proportional to the radius of local region. The proportional parameter is set as 1 by default.

After interpolating the features of kernel points, the output of convolution operation can be formulated as:

$$F_{out}(\boldsymbol{p_i}) = \sum_{\kappa} w(\boldsymbol{p_\kappa}) \frac{1}{|N_{\boldsymbol{p_\kappa}}|} \sum_{\boldsymbol{p'_j} \in N_{\boldsymbol{p_\kappa}}} \phi(\boldsymbol{p'_j}, \boldsymbol{p_\kappa}) f(\boldsymbol{p_j}) \quad (5)$$

Where $\boldsymbol{p'_j}$ is the transformed coordinates of neighboring point $\boldsymbol{p_j}$, $N_{\boldsymbol{p_\kappa}} = \{\boldsymbol{p'_j} | \|\boldsymbol{p'_j} - \boldsymbol{p_\kappa}\|_2 < \sigma\}$ denotes the influence area of each kernel point , w denotes the weight matrix corresponding with $\boldsymbol{p_\kappa}$.

C. Spatial Feature Encoder Branch

Unlike the distribution of pixels in images, the spatial distribution of points is much important in point clouds. The spatial distribution of pixels contains no information since their distributions in all local regions are grids. On the contrary, the spatial distribution of local point patches represents the underlying geometric structure of each local region. To perceive this spatial structure, we utilize an additional spatial encoder branch which consists of Spatial Feature Encoder (SFE) modules [16], as shown in Fig. 2. Each SFE module takes transformed coordinates of neighboring points in each layer as input and lift it by shared MLPs. The lifted features are combined with the spatial feature from previous SFE subsequently except the first layer. After that, we concatenate this feature with input to implement the spatial information and send it to shared MLPs to prepare for the next SFE. The shared MLP is rotation invariant as it operates the associated features of each point independently and do not consider the spatial distribution of neighboring points.

III. Experimental Results

In this section, we present the datasets, methods, and evaluation metrics to demonstrate the results of LFNet.

A. Experimental Setup

We evaluate our approach on ModelNet40 [17] and ShapeNet-Part [18] dataset for classification and segmentation, respectively. ModelNet40 consists of 12,311 CAD models from 40 categories, which are split into 9,843 for training and 2,468 for testing. ShapeNet Part contains 16,880 models from 16 classes, and each model is annotated with 2 to 6 parts with 50 different parts in total. We follow [2] to split the data with 14,006 for training and 2,874 for testing. We randomly select 1024 points as input for classfication and evaluate them with Overall Accuracy(OA). For segmentation, we sample 2048 points and compare segmentation performance by average category mIoU (Cat. mIoU).

TABLE I
COMPARISONS WITH OTHER POINT-BASED NETWORKS ON MODELNET40 IN OVERALL ACCURACY (OA), SHAPENET IN AVERAGE CATEGORY IOU (MIOU). OUR METHODS CAN GENERALIZE TO ROTATED SHAPES EVEN UNSEEN DURING TRAINING. 'NOR' DENOTES NORMAL, '-' DENOTES UNKNOWN.

Methods	Input	ModelNet40		ShapeNet	
		N/R	R/R	N/R	R/R
SO-Net [3]	xyz+nor	9.6	80.2	14.4	-
PointNet [1]	xyz	12.5	75.5	28.3	74.4
PointNet++(MSG) [2]	xyz	21.4	77.4	35.0	76.7
DGCNN [4]	xyz	29.7	81.1	30.9	73.3
PRIN [13]	xyz	70.3	80.1	54.2	54.2
SphericalCNN [14]	voxel	78.6	86.9	-	-
RIConv [12]	xyz	86.4	86.4	75.4	75.4
SPHNet [20]	xyz	86.6	87.7	-	-
SRINet [15]	xyz+nor	87.0	87.0	57.4	57.4
ClusterNet [21]	xyz	87.1	87.1	-	-
Ours	xyz	**90.9**	**91.0**	**80.4**	**80.4**

The experiments are conducted in two conditions: 1) N/R, 2) R/R, where N/R denotes training with no rotations and testing with random rotations, R/R denotes training with random rotations and testing with random rotations.

B. Comparative analysis

The results of classification and segmentation on rotated shapes are shown in Table I. In both datasets, we can see that LFNet significantly outperforms other methods when testing with rotated shapes. Note that we do not compare with ClusterNet [21], SphericalCNN [14] and SPHNet [20] on segmentation task, since they are designed for classification or the performance on ShapeNet dataset have not been reported. As can be seen from Table I, our method significantly outperforms both methods learning from the rotation augmented training data and rotation invariant methods. Although utilizing additional spatial transformer network [11], PointNet [1] and DGCNN [4] still cannot generalize to rotated shapes when lack of rotation augmented training data. Defined with the original coordiantes and lack of rotation robust structure, simply learning from the rotation augmented input data harms the performance of SO-Net [3] and PointNet++ [2]. Compared with recently published methods RIConv [12], PRIN [13], SphericalCNN [14], SRINet [15], SPHNet [20] and ClusterNet [21] which also try to solve the rotation invariant problem, our method still achieves better performance. Besides, the classification and segmentation performance of the proposed method in different situations are close, which proves that our method achieves rotation invariance. The small discrepancy between different experimental situations is induced by the heuristic of FPS operation. To verify our assumption that rotation affects the model by changing the spatial distribution at *input level* and *convolution level*, we replace the transformed coordinates in our methods with the original coordinates as shown in Table II. As can be seen, the original coordinates in each of the two levels will make the network vulnerable to rotations. Although this can be alleviated by rotation augmented training data, models A-C are still worse than model D which acheives rotation invariance at both levels. The positive effects of the proposed modules are shown in Table III, where the discrete convolution operator brings a performance improvement of 0.5% in basic cases and 0.6% in full case compared with commonly used shared MLPs, and SFE module brings more than 0.8% in both situations as it incorporates local spatial information. Fig. 4 illustrates some typical visual comparison results between RIConv and LFNet over the ShapeNet part dataset. It is clearly seen that our network generalizes well on unseen orientations.

TABLE II
EFFECTIVENESS OF TRANSFORMING NEIGHBORING POINTS ON MODELNET40 DATASET. (O: ORIGINAL COORDINATES, T: TRANSFORMED COORDINATES.)

Model	Input Level	Convolution Level	N/R	R/R
A	O	O	14.6	88.2
B	O	T	16.9	89.3
C	T	O	14.6	88.4
D	T	T	**90.9**	**91.0**

TABLE III
EFFECTIVENESS OF THE PROPOSED MODULE ON MODELNET40 DATASET

Model	Convolution	SFE	N/R
E	MLPs	✗	89.5
F	MLPs	✓	90.3
G	Discrete conv	✗	90.0
H	Discrete conv	✓	**90.9**

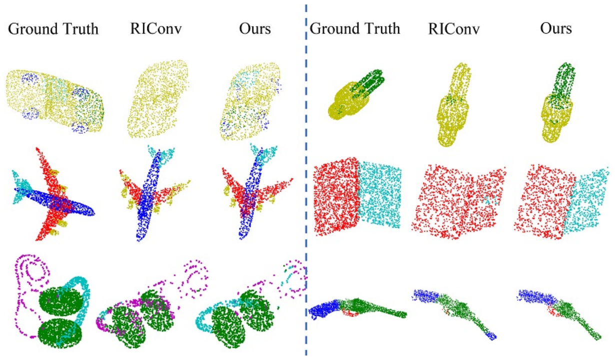

Fig. 4. Visualization of results. We test RIConv [12] and LFNet on rotated point clouds when trained with the canonical orientated dataset.

IV. CONCLUSION AND FUTURE WORK

In this letter, we present a method to remove the dependency between the spatial distribution of neighboring points and the orientation of the object. Based on the transformed rotation invariant neighboring points, we further eliminate the influence of rotations at both input and convolution levels to achieve rotation invariance of the model. Besides, an additional spatial encoding branch is utilized to perceive the local structure and enrich the spatial information. Experimental results demonstrates the superiority of our proposed method when compared with other competing approaches.

References

[1] K. M. C. R. Qi, H. Su and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 652–660, 2017.

[2] H. S. C. R. Qi, L. Yi and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 5099–5108, 2017.

[3] B. M. C. Jiaxin Li and G. H. Lee, "So-net: Selforganizing network for point cloud analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9397–9406, 2018.

[4] Z. L. S. E. S. M. M. B. Y. Wang, Y. Sun and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," in *ACM TOG*, 2019.

[5] M. S. W. W. X. D. Y. Li, R. Bu and B. Chen, "PointCNN: Convolution on x-transformed points," in *Advances in Neural Information Processing Systems*, pp. 828–838, 2018.

[6] S. X. Y. Liu, B. Fan and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–10, 2019.

[7] J.-E. D. B. M. F. G. H. Thomas, C. R. Qi and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 6411–6420, 2019.

[8] Z. Z. A. Komarichev and J. Hua, "A-cnn: Annularly convolutional neural networks on point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7421–7430, 2019.

[9] G. M.-J. L. S. X. C. P. Y. Liu, B. Fan, "Densepoint: Learning densely contextual representation for efficient point cloud processing," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 5239–5248, 2019.

[10] Q. Z. F. L. Wu, W., "Pointconv: Deep convolutional networks on 3d point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9621–9630, 2019.

[11] A. Z. Max Jaderberg, Karen Simonyan, "Spatial transformer networks," in *In Advances in neural information processing systems*, pp. 2017–2025, 2015.

[12] H. B. R. D.-Y. S. Zhang, Z., "Rotation invariant convolutions for 3d point clouds deep learning," in *International Conference on 3D Vision*, pp. 204–213, 2019.

[13] Q. L.-L. M. W. W. Y. T. Yang You, Yujing Lou and C. Lu, "Prin: Pointwise rotation-invariant network," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[14] A.-B. C. M.-A. D. K. Esteves, C., "Learning so (3) equivariant representations with spherical cnns.," in *Proceedings of the European Conference on Computer Vision (ECCV).*, pp. 52–68, 2018.

[15] L.-Z. X. J. Sun, X., "Srinet: Learning strictly rotation-invariant representations for point cloud classification and segmentation," in *MM 2019 - Proceedings of the 27th ACM International Conference on Multimedia*, pp. 980–988, 2019.

[16] S. L.-e. a. L.Z. Chen, X.Y. Li, "Lsanet: Feature learning on point sets by local spatial attention," 2019. arXiv preprint arXiv:1905.05442.

[17] A. K.-F. Y. L. Z. X. T. Z. Wu, S. Song and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1912–1920, 2015.

[18] D. C. I. S. M. Y. H. S. C. L. Q. H. A. S. L. G. L. Yi, V. G. Kim, "A scalable active framework for region annotation in 3d shape collections," *ACM Transactions on Graphics*, vol. 35, no. 6, 2016.

[19] Y. X. Y. L. G. Z. H. H. J. Dai, H. Qi and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 764–773, 2017.

[20] R. M. J. P. Y. O. M. Poulenard, A., "Effective rotation-invariant point cnn with spherical harmonics kernels," in *Proceedings - 2019 International Conference on 3D Vision (3DV)*, pp. 47–56, 2019.

[21] L. G. X. R. C. T. W. M. L. L. Chen, C., "Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4994–5002, 2019.

V. Supplementary Material

A. *Rotation Invariance of Transformed Coordinates*

A point cloud is a set of points often denoted as $P = \{(x_i, y_i, z_i)|x_i, y_i, z_i \in R\}$, and it also can be expressed as $P \in R^{N\times 3}$ in the matrix form where N is the number of points. The rotation transformation applied to the point clouds corresponds to a linear mapping with a 3×3 real orthogonal matrix. To better describe the rotation invariance, we conduct the definition of rotation invariant transformation as below.

Definition 1: (RI Transformation). If $N, C \in R^+$, a transformation $T : R^{N\times 3} \longrightarrow R^{N\times C}$ is a Rotation Invariant (RI) transformation when

$$T(P) = T(R(P))\,. \tag{6}$$

holds for all points $P \in R^{N\times 3}$ and all rotation transformation $R \in SO(3)$. Then T is called a rotation-invariant transformation of P.

For example, since the normals rotate together with point clouds, the inner product of two arbitrary points or their corresponding normals is a rotation-invariant transformation since

$$< R\boldsymbol{p_i}, R\boldsymbol{p_j} >= \boldsymbol{p_i}^T R^T R\boldsymbol{p_j} = \boldsymbol{p_i}^T\boldsymbol{p_j} =< \boldsymbol{p_i}, \boldsymbol{p_j} > \,. \tag{7}$$

holds for $\forall \boldsymbol{p_i}, \boldsymbol{p_j} \in R^3$ and $R \in SO(3)$. It degrades to the norm $\|\cdot\|_2$ when $\boldsymbol{p_i}$ is equal to $\boldsymbol{p_j}$.

In the constructed local coordinate frame, the transformed coordinates $\boldsymbol{p'_j}$ of an arbitrary point $\boldsymbol{p_j} \in P$ is rotation invariant. The proof is below:

Proof. Denotes the x-axis base vector of the constructed local coordinate frame after rotation as $\boldsymbol{e_{Rx}}$, it is constructed by the rotated $\boldsymbol{p_i}$ as:

$$\begin{aligned}\boldsymbol{e_{Rx}} &= (R\boldsymbol{p_i}/\|\boldsymbol{p_i}\|) \times (R\boldsymbol{n_i}) = (detR)(R^{-1})^T(\boldsymbol{p_i}/\|\boldsymbol{p_i}\| \times \boldsymbol{n_i})\\ &= R(\boldsymbol{p_i}/\|\boldsymbol{p_i}\| \times \boldsymbol{n_i}) = R\boldsymbol{e_x}\,.\end{aligned} \tag{8}$$

For an arbitrary point $\boldsymbol{p_j} \in S$, the x coordinate of rotated $\boldsymbol{p_j}$ in this local coordinate frame is obtained by inner product:

$$\begin{aligned}x'_{Rj} &=< R(\boldsymbol{p_j} - \boldsymbol{p_i}), \boldsymbol{e_{Rx}} >=< R(\boldsymbol{p_j} - \boldsymbol{p_i}), R\boldsymbol{e_x} >\\ &=< \boldsymbol{p_j} - \boldsymbol{p_i}, \boldsymbol{e_x} >= x'_j\,.\end{aligned} \tag{9}$$

Therefore, the transfer of x coordinate is a rotation invariant transformation. Similarly, the transformations of y, z coordinates of $\boldsymbol{p_j}$ can be proved to be rotation-invariant when

$$\begin{aligned}\boldsymbol{e_{Ry}} &= \boldsymbol{e_{Rx}} \times (R\boldsymbol{n_i}) = (R\boldsymbol{e_x}) \times (R\boldsymbol{n_i}) = R\boldsymbol{e_y}\,.\\ \boldsymbol{e_{Rz}} &= R\boldsymbol{n_i} = R\boldsymbol{e_z}\,.\end{aligned} \tag{10}$$

Hence, the transfer of an arbitrary point $\boldsymbol{p_j} \in P$ in the original space to its corresponding point $\boldsymbol{p'_j} = (x'_j, y'_j, z'_j)$ in the constructed coordinate frame is a rotation invariant transformation.

B. *Rotation Invariance of Grouping Operation*

For each point clouds, the reference points are sampled by the Farthest Point Sampling (FPS) strategy. From each of which K-Nearest Neighbors (K-NN) are obtained in the grouping operation to form the local point sets. As the K-NN neighborhood of a reference point $\boldsymbol{p_i}$ is uniquely determined by $\|\boldsymbol{p_j} - \boldsymbol{p_i}\|$ for all points $\boldsymbol{p_j} \in P$, which has been proven to be rotation-invariant, grouping the neighboring points by K-NN algorithm for the reference point $\boldsymbol{p_i} \in P$ is a rotation invariant operation.

C. *Rotation Invariance of Paralleled Cases*

A caveat from the local frame construction scheme is that the base vectors $\boldsymbol{e_x}$ and $\boldsymbol{e_y}$ degenerate into $\boldsymbol{0}$ when the reference vector $\boldsymbol{p_i}$ parallels with normal $\boldsymbol{n_i}$. This will make the transformed x, y-coordinates of neighboring points degrade into 0, thus the information in x, y-axis are discarded and squashed into z-axis. Such cases occur when the local surface at reference point $\boldsymbol{p_i}$ is distributed symmetrically and perpendicular with $\boldsymbol{p_i}$. In this cases, we select the first unparalleled neighboring point $\boldsymbol{p_\ell}$ in K-NN neighbors of $\boldsymbol{p_i}$ to construct the local coordinate frame, the base vectors $\boldsymbol{e_x}$ and $\boldsymbol{e_y}$ are then recalculated by:

$$\begin{aligned}\boldsymbol{e_x} &= \boldsymbol{p_i} \times (\boldsymbol{p_\ell} - \boldsymbol{p_i})/\|\boldsymbol{p_\ell} - \boldsymbol{p_i}\|\\ \boldsymbol{e_y} &= \boldsymbol{e_x} \times \boldsymbol{n_i}\end{aligned} \tag{11}$$

Since rotation transformation will not change the permutation of point clouds, the relative order of neighboring points with equal distance in K-NN is stable. Meanwhile, as the K-NN neighbors of the reference point $\boldsymbol{p_i}$ is uniquely determined by the distance, the selection of first unparalleled neighboring point $\boldsymbol{p_\ell}$ of reference point $\boldsymbol{p_i}$ will not be affected by rotations. Therefore, the base vectors $\boldsymbol{e_{Rx}}$ and $\boldsymbol{e_{Ry}}$ of the constructed coordinate frame after rotation can be formulated as:

$$\begin{aligned}\boldsymbol{e_{Rx}} &= (R\boldsymbol{p_i}) \times (R\boldsymbol{p_\ell} - R\boldsymbol{p_i})/\|R\boldsymbol{p_\ell} - R\boldsymbol{p_i}\|\\ &= R\boldsymbol{p_i} \times (\boldsymbol{p_\ell} - \boldsymbol{p_i})/(\|R\| \cdot \|\boldsymbol{p_\ell} - \boldsymbol{p_i}\|) = R\boldsymbol{e_x}\,.\\ \boldsymbol{e_{Ry}} &= \boldsymbol{e_{Rx}} \times (R\boldsymbol{n_i}) = R\boldsymbol{e_y}\,.\end{aligned} \tag{12}$$

Hence, the transformed x, y coordinates of an arbitrary point $\boldsymbol{p_j}$ can be proven to be rotation invariant as in Eq. 9. Therefore, the transformed distribution of neighboring points in the paralleled cases will not be affected by rotations either.

D. *Visualization of Kernel Points*

The kernel points used in our method is demonstrated in Fig. 5 which obtained by the optimization function in KPConv [7] with number of 15. Please refer to KPConv for more details.

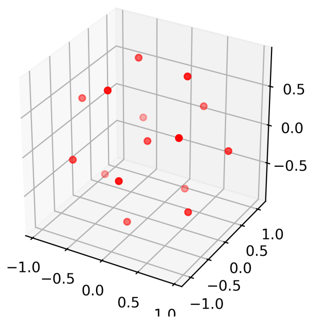

Fig. 5. Visualization of kernel points.