3D-GFE: a Three-Dimensional Geometric-Feature Extractor for Point Cloud Data

Yu-Chen Chou, Yen-Po Lin, Yang-Ming Yeh, Yi-Chang Lu
Graduate Institute of Electronics Engineering, National Taiwan University
{r07943018, r07943127, d05943006, yiclu}@ntu.edu.tw

***Abstract*—In this paper, we propose a novel framework that extracts rotation-invariant features relative to the centroid and the reference point in a local point set. Furthermore, we search different scales of nearest neighbors simultaneously to acquire more rotation-invariant information around certain specific point cloud data. We evaluate our architecture with two point cloud tasks, object classification and part segmentation. Experiment results show that our method generates consistent results on randomly rotated data and achieves state-of-the-art performance without any rotational data augmentation. For classification, when training and testing with arbitrary rotations, our model is able to reach averagely 89.0% and 73.5% accuracy on the ModelNet40 and ScanObjectNN datasets, respectively. On the ShapeNet dataset, which is a part segmentation task, our model can achieve 77.7% mIOU.**

***Index Terms*—Point Cloud, Classification, Segmentation, Rotation-Invariant, Geometric-Feature Descriptor**

I. Introduction

In recent years, deep learning methods for 3D data processing have brought about significant progress in object classification and object segmentation. Research in this area is increasing with more and more demands for consumer devices such as indoor navigation, autonomous driving, and virtual reality. It is worthwhile to mention that, unlike 2D images, 3D data can be depicted in different ways. For instance, 3D data can be represented as polygonal meshes, point clouds, volumetric grids, and depth images. Therefore, the approaches to handle different formats are also distinct.

Among many methods for processing 3D data, point cloud, a set containing displacement of points in 2D or 3D space, is a convenient data format to represent the shape of objects. The data format not only gets rid of uninformative 3D features but is also easy to process and observe. Each point in a point cloud is described by x, y, and z coordinates in 3D space. Previous studies have improved the learning of local features in a point cloud by introducing various convolution operators and demonstrate good performances. However, when it comes to rotation, the performances of some models are likely to degrade. Though some may make use of rotation augmentation to make their models more robust for rotated data, they are still not guaranteed to be rotation-invariant.

To resolve this issue, we propose a method to extract rotation-invariant features for point clouds instead of merely taking 3D coordinates as the input. We also search different scales of nearest neighbors when the features propagate through the model. In that way, our method can be rotation-invariant on processing point cloud objects.

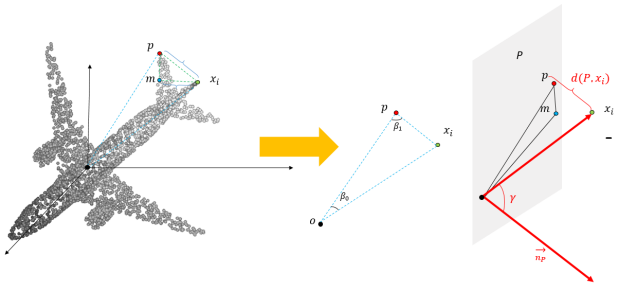

Fig. 1. Our proposed 3D Geometric-Feature Descriptor. Combined with Eq. 1, we take the origin o into consideration, and extend the features to reinforce the rotation-invariance.

II. Related Work

Multi-view and volumetric networks. To reuse the convolutional models that are widely explored in computer vision, some researchers proposed to project point cloud data onto grid-based 2D or 3D space. Su *et al.* [1] projected 3D data onto 2D images from different views. In that way, they could combine the data with some well-known architectures which have good performances on 2D image tasks. Wu *et al.* [2] represented geometric shapes as a probability distribution on a 3D voxel grid, and used binary tensor to represent whether the voxel is inside the mesh surface. Similarly, Maturana *et al.* [3] partitioned 3D space into many small grids, over which 3D convolutional neural networks can be applied.

Point-based deep learning methods. Despite the fact that it is convenient to apply 2D deep learning models on projection-based methods mentioned above, we may lose some information when projecting onto lower dimensions. Moreover, due to the sparsity of point cloud data, the above methods often cost much memory and thus inefficient. Qi *et al.* [4] advocated representing 3D data as points lying in 3-dimensional spaces depicting the outline of an object structure. They took coordinates of these points as input data, used several multi-layer perceptrons (MLPs) to aggregate the coordinate information as global features, and then applied symmetric functions, such as max-pooling, to distinguish the shape of the object. Notice that the authors also took the normal vectors of points as extra information for better results.

Local features learning. After PointNet [4] was proposed, more and more models focused on learning point-based information on 3D space because of its convenience as well as fewer resources required. [5], [6], [7], [8] not only learned the coordinates in 3D space but also learned the relation between points and points. Qi *et al.* [5] improved the performance of [4] by means of grouping different ranges and amounts of points so as to learn the local information between these points. They also took the relation between points and points to learn more information. Wang *et al.* [6] proposed a representation regarding graph-based methods to evaluate the strength of a point-based model, and extract local features by subtracting neighbors from local reference points. Li *et al.* [7] proposed χ-conv layers that can learn a χ-transform matrix to achieve permutation invariance, and then combined the layers with traditional convolutional layers. Zhang *et al.* [9] introduced ShellConv operator to group local neighbors abiding by the Euclidean distance from each reference point, and took difference number of nearest neighbors at different stages to make the model learn additional information.

Rotational invariant feature extraction. Although miscellaneous 3D deep learning models and learning methods were proposed, the robustness to arbitrary rotations is still a challenging task on point cloud data. PointNet [4] tried to deal with this problem by learning a T-net matrix to make their model less vulnerable to random rotations. LaLonde *et al.* [10] utilized on hierarchical clustering and extracted features invariant to rotation directly. Zhang *et al.* [11] calculated centroid of local points and took rotation-invariant features for learning to achieve rotation-invariance property.

Inspired by [11], our work aims to propose a rotation-invariant point cloud architecture that is robust to arbitrary rotations. We also show that our work is still competitive on real-world classification tasks, which include the background in the point cloud data.

III. Methods

In this section, we introduce our overall flow. We follow the approach to extract rotation-invariant features proposed by [11] at first. Then we combine these features with our newly proposed rotation-invariant features to formulate our 3D Geometric-Feature Descriptor. Subsequently, we introduce the proposed multi-kNN graph, which makes our learning process more robust. We combine 3D Geometric-Feature Descriptor and multi-kNN graph to formulate our proposed model. At the last of this section, we detail the architecture of our proposed neural network.

A. 3D Geometric-Feature Descriptor

Given a local reference point p, Zhang *et al.* [11] used k-nearest neighbors (k-NN) algorithm to find k points closest to p, and formed a local point set. We denote the point set as $X_p = \{x_i, i = 1, ..., k\}$. After k nearest points are found, we can easily find the centroid, m, of the point set. Then [11] used the relationship among the reference point p, the centroid point m, and each local point x_i, to construct rotation-invariant features. The visualized relation is shown in Fig. 2.

Fig. 2. Rotational invariant features extraction proposed in [11]. Here, d_0 is $\overline{px_i}$, d_1 is $\overline{mx_i}$, α_0 is $\angle mpx_i$, and α_1 is $\angle pmx_i$ mentioned in Eq. 1.

The three points mentioned above constitute a triangle. The $\overline{px_i}$, $\overline{mx_i}$, $\angle mpx_i$ and $\angle pmx_i$ are the selected rotation-invariant features. The feature extraction can be represented as:

$$RIF(x_i; \overrightarrow{pm}) = [\overline{px_i}, \overline{mx_i}, \angle mpx_i, \angle pmx_i]. \quad (1)$$

Note that the angles are represented as cosine values due to the convenience in calculations.

The method to extract local features indeed has outstanding results experimented on rotated point clouds. However, this way to depict relationships of a local point set in Euclidean space neglects the relations between these triangles formed by different nearest points $x_i \in X_p$ and $\overrightarrow{pm}$. That is to say, these triangles with different vertex x_i may lie in different planes in 3D space. We will lose the information of the angles between the planes containing different triangles. To deal with the problem, we propose 3D Geometric-Feature Descriptor, which combines features used in Eq. 1 with global information to help describe the geometric features.

Except that distances and angles of the points in one local point set are rotation-invariant, the displacement of origin o is also unchanged through rotation. As a result, we take the relations between the origin o and one local point set into consideration. Our descriptor is defined as:

$$f(p, x_i, m) = [RIF(x_i; \overrightarrow{pm}), d(P, x_i), \gamma, \beta_0, \beta_1]. \quad (2)$$

Here, P is the specific plane containing m, p and the origin o. $d(P, x_i)$ is the distance between x_i and plane P, and γ is the angle between the normal vector of P and $\overrightarrow{ox_i}$. β_0 and β_1 denote $\angle pox_i$ and $\angle opx_i$, respectively. The angles here are also represented as cosine values.

In this way, the calibration of the origin becomes more crucial. Therefore, we shift the whole point cloud consistently and take the centroid of the whole point cloud as the origin. To make it more clear, we visualize our feature descriptor in Fig. 1.

To learn the relations between different neighboring points, consider a plane P formed by three points o, m, p. (WLOG, these three points do not lie in a line.) The first two extra features, $d(P, x_i)$ and γ, are the distance between the nearest point x_i and plane P, and the angle between $\overrightarrow{ox_i}$ and the

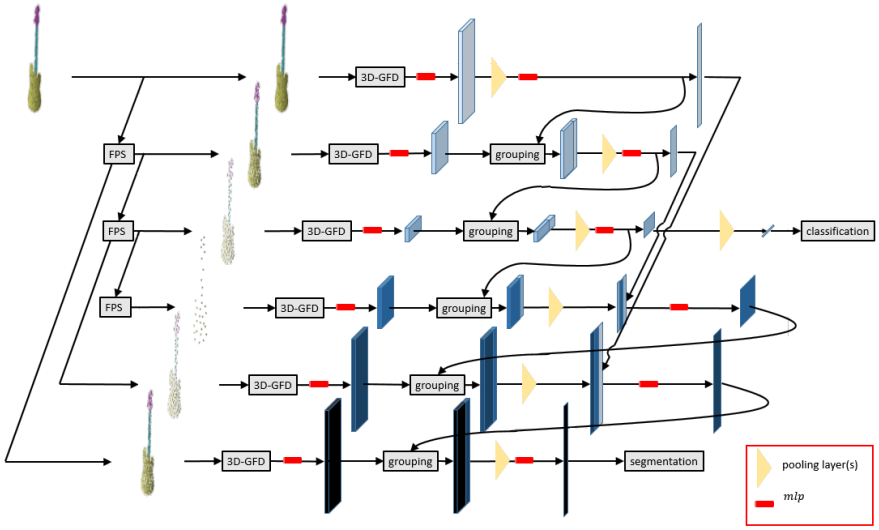

Fig. 3. Overall architecture of our neural networks. We perform farthest point sampling (FPS) to search different neighboring points in every module of the classification task, and follow [12] to build an encoding-decoding architecture for the segmentation task. For simplicity, we set the number of k-NN graphs N_K to 1 in this example.

normal vector of the plane. On account of the plane, P, is formulated by o, m, p, we can extract the stereoscopic information from these neighboring points. The latter two features, $\angle pox_i$ and $\angle opx_i$, give the information among the origin o, the local reference point p, and the nearest point x_i. Since the displacement of the centroid has the risk of being affected by the perturbation of these nearest points and the outlier if the number of neighbors is insufficient, adding these two features improves our learning robustness.

Through MLPs (multi-layer perceptrons), we can then embed these geometric features into higher dimensional feature spaces.

B. Multi-kNN Graph

Through the whole module, several top-of-the-notch models take the k-NN algorithm to acquire the information between the neighbors and the reference point. These models usually apply max-pooling layers to acquire the features with the highest response. Inspired by the architecture of [13] and multi-scale grouping mentioned in PointNet++ [5], we take different numbers of hyperparameter k at the same time. Details are illustrated in Fig. 4.

Using the property that the centroid will change if we take different numbers of neighbors in a point set, we search three k-NN graphs with different k values instead of only one scale k-NN graph. In this way, for each reference point p, we are capable of acquiring different information according to the distribution with different numbers of the neighbors.

C. Architecture

We formulate the 3D Geometric-Feature Descriptor, multi-kNN graph, multi-layer perceptrons, aggregation function, and grouping operation as a module called 3D Geometric-Feature

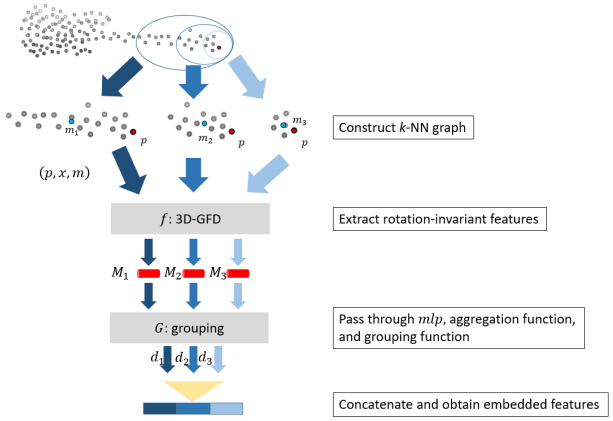

Fig. 4. Visualization of processing multi-kNN graphs with different scales of k.

Extractor (abbreviated as 3D-GFE). To effectively emphasize the features in Euclidean space, we adopt the architecture as that in ShellConv [9]. Our model is shown in Fig. 3, and each row represents a complete 3D-GFE module.

The model at the encoding stage is composed of three 3D-GFE modules. The step in each module can be formulated as:

$$F(p) = M([A(d_1(p)), A(d_2(p)), ..., A(d_{N_K}(p))]); \quad (3)$$

$$d_i(p) = G(M_i(f(p, x, m_i)), F_{prev}(x)). \quad (4)$$

Here, $F(p)$ is the output feature of local reference point p from this module, and $F_{prev}(\cdot)$ is the output feature obtained from the previous module. $f(\cdot)$ is our proposed 3D-GFD mentioned in Eq. 2. N_k is the number of k-NN graphs. m_i is defined by different scale of k, i.e. $m_i \neq m_j$ if $i \neq j$.

TABLE I
CLASSIFICATION ACCURACY (%) ON THE MODELNET40 DATASET [14] COMPARED WITH OTHER POINT-BASED MODELS. ALL THE NUMBER OF ORIGINAL SAMPLING POINTS ARE 1,024.

Methods	z/z	SO(3)/SO(3)	z/SO(3)
PointNet [4]*	89.2	75.5	16.4
PointNet++ [5]*#	91.9	77.4	18.4
PointCNN [7]**	91.3	84.5	41.2
DGCNN [6]*	**92.2**	81.1	20.6
ClusterNet [10]	87.1	87.1	87.1
SRINet [15]	87.0	87.0	87.0
RIConv [11]	86.5	86.4	86.4
LGR-Net [16]#	90.9	**91.1**	**90.9**
Ours	88.6	89.0	89.4

*: reported by [10] **: reported by [11]
#: require both point cloud and normal inputs

$M(\cdot)$ implies that the features are through MLPs (multi-layer perceptrons) with trainable weights. Every MLP is followed by a batch normalization layer and leaky ReLU activation function to achieve a better effect of learning. Notice that all the f functions with the same p and i are using the same MLP M_i. A is the aggregation layer. It is also a symmetric function to aggregate information from the nearest points with regard to p. Here, we take maximal and average values among the neighbors simultaneously as the aggregation function. We found that the model generates better results if we use both average-pooled and max-pooled features instead of using max-pooled features only. G is the grouping layer used to concatenate $F_{prev}(\cdot)$.

In the classification task, through the whole neural networks, we perform farthest point sampling (FPS) to reduce the burden of memory usage, and take the down-sampled points as input points to acquire different geometric features in every module. In segmentation tasks, we follow U-net [12] and design an encoder-decoder architecture with skip connections. At the decoding stage, starting from the number of points at the end of the encoder stage, our decoder uses 3D Geometric-Feature Extractors to output denser points and fewer channels until the number of outputs recovers to the original number of points. Skip connections are performed at the first two layers of the decoding stage to concatenate the output features from the encoding layers. Moreover, multi-kNN graphs are also implemented in the decoding stage.

IV. EXPERIMENTS

This section provides the implementation and training details according to different learning tasks: synthetic object classification, real-world object classification, and part segmentation. We also make a comparison with other existing state-of-the-art methods.

A. Parameter Setting

In our design, our neural network used for the classification task has three encoding layers. Except for encoding raw Euclidean features, each layer also utilizes the features obtained by the previous layer to train together with geometric features. We combine the model with another three decoding layers in the segmentation task.

TABLE II
CLASSIFICATION ACCURACY (%) ON THE TASK (PB_T50_RS) IN THE SCANOBJECTNN DATASET [17].

PB_T50_RS			
Method	z/z	SO(3)/SO(3)	z/SO(3)
PointNet [4]*	68.2	42.2	17.1
PointNet++ [5]*#	77.9	60.1	15.8
PointCNN [7]*	**78.5**	51.8	14.9
DGCNN [6]*	78.1	63.4	16.1
RICONV [11]*	67.9	68.3	67.9
LGR-Net [16]#	72.7	72.9	**72.7**
Ours	73.5	**73.5**	**72.7**

*: reported by [16]
#: require both point cloud and normal inputs

TABLE III
OBJECT PART SEGMENTATION RESULT ON THE SHAPENET PART DATASET [2]. THE METRIC IS MEAN PER-CLASS IOU (%).

	SO(3)/SO(3)	z/SO(3)
PointNet [4]*	74.4	37.8
PointNet++ [5]*#	76.7	48.3
PointCNN [7]*	71.4	34.7
DGCNN [6]*	73.3	37.4
RICONV [11]	75.3	75.3
LGR-Net [16]#	**80.1**	**80.0**
Ours	77.7	78.2

*: reported by [11]
#: require both point cloud and normal inputs

Our network is implemented in Pytorch, executed on a computer with Intel (R) Core (TM) i7-7700 CPU with 64 GB RAM, and run on two NVIDIA GeForce GTX 1080 Ti GPUs.

B. ModelNet40

The synthetic object classification task is trained on the ModelNet40 dataset [14], which is a widely used benchmark for point cloud analysis. The ModelNet40 dataset [14] includes 12,311 meshed CAD models categorized in 40 objects. The data is split into two parts, 9,843 models are used for training, and 2,468 models are used for testing. The point cloud data are uniformly sampled from the mesh face of these CAD models and re-scaled into a unit sphere. For a fair comparison, we only use 1,024 points from each data, and each point only contains the information of (x, y, z) coordinates. In addition, we conduct 5-fold cross-validation in the training data for steady performance. We save the model with the highest accuracy on the validation set in these five folds, and then use the model for testing.

Table I shows our results compared to state-of-the-art work. We perform experiments in three cases: training and testing data with only rotated along the z-axis (z/z), training and testing data with arbitrary rotations (SO(3)/SO(3)), and training with only rotated along the z-axis and testing with arbitrary rotations (z/SO(3)). As the results show, our proposed method generates the best results in the SO(3)/SO(3) and z/SO(3) cases even though we conducted extra cross-validation on the training set, which reduces the available training data. Moreover, among these cases, our method almost generates the same result, implying rotation-invariance.

C. ScanObjectNN

The real-world object classification task is trained on the ScanObjectNN dataset [17]. The ScanObjectNN dataset [17] contains approximately 15,000 objects that are categorized into 15 categories with 2,902 unique object instances. Additionally, part of the data are processed through random perturbation, scaling, and rotation, making it more challenging. Like what we have done on the ModelNet40 dataset, we conduct 5-fold cross-validation on the training data for a fair comparison.

We present Table II with the results conducted on the challenging task (PB_T50_RS) provided in [17]. We compare our results with those high-quality models, which also merely take coordinates as input features. Except for the augmented testing data provided in [17], we also show the result in three cases: z/z, SO(3)/SO(3), and z/SO(3). Among these cases, our model achieves the highest accuracy in the second and the third cases among models.

D. 3D Object Part Segmentation

Part segmentation task is to segment an object into many parts. We experiment on the ShapeNet part dataset [2], which contains 16,881 shapes from 16 object categories, annotated with 50 parts in total. There are 2,048 points sampled from each object, annotated with no more than six parts. We follow [2] and split the dataset into 14,006 training models along with 2,874 testing models.

We calculate the mean Intersection-over-Union (mIOU) on points to compare with other models. The result is shown in Table III. Similarly, the models that extract rotation-invariant features have better performances, and our model outperforms all other methods.

V. Conclusion

In this paper, we propose a novel rotation-invariant feature extractor to deal with 3D point cloud data with arbitrary rotations. We use distances, angles, and planes formulated by specific points to construct the stereotactic feature descriptor. Combined with multi-kNN that simultaneously takes different numbers of neighbors as the targets, our method is more robust to jittered point cloud data. Extensive experiments on well-known 3D benchmarks demonstrate that our proposed features and multi-kNN graph architecture improve the performance.

Acknowledgment

This work is partially supported by National Taiwan University under Grant number 110L890605, and the Ministry of Science and Technology, Taiwan, under Grant numbers 107-2221-E-002-123-MY2 and 110-2221-E-002-097. The authors would also like to thank National Center for High-performance Computing (NCHC) for providing computational and storage resources.

References

[1] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 945–953.

[2] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas, "A scalable active framework for region annotation in 3d shape collections," *ACM Transactions on Graphics (ToG)*, vol. 35, no. 6, pp. 1–12, 2016.

[3] Daniel Maturana and Sebastian Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 922–928.

[4] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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*, 2017, pp. 652–660.

[5] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, 2017, pp. 5099–5108.

[6] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 5, pp. 1–12, 2019.

[7] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen, "Pointcnn: Convolution on x-transformed points," in *Advances in neural information processing systems*, 2018, pp. 820–830.

[8] Wenxuan Wu, Zhongang Qi, and Li Fuxin, "Pointconv: Deep convolutional networks on 3d point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9621–9630.

[9] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung, "Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1607–1616.

[10] Rodney LaLonde, Dong Zhang, and Mubarak Shah, "Clusternet: Detecting small objects in large scenes by exploiting spatio-temporal information," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4003–4012.

[11] Zhiyuan Zhang, Binh-Son Hua, David W Rosen, and Sai-Kit Yeung, "Rotation invariant convolutions for 3d point clouds deep learning," in *2019 International Conference on 3D Vision (3DV)*. IEEE, 2019, pp. 204–213.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[13] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[14] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

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

[16] Chen Zhao, Jiaqi Yang, Xin Xiong, Angfan Zhu, Zhiguo Cao, and Xin Li, "Rotation invariant point cloud classification: Where local geometry meets global topology," *arXiv preprint arXiv:1911.00195*, 2019.

[17] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," *arXiv*, pp. arXiv–1908, 2019.