Quadratic Terms Based Point-to-Surface 3D Representation for Deep Learning of Point Cloud

Tiecheng Sun, Guanghui Liu, *Senior Member, IEEE*, Ru Li, *Student Member, IEEE*, Shuaicheng Liu, *Member, IEEE*, Shuyuan Zhu, *Member, IEEE*, and Bing Zeng, *Fellow, IEEE*

***Abstract*—In this paper, we introduce a novel point-to-surface representation for 3D point cloud learning. Unlike the previous methods that mainly adopt voxel, mesh, or point coordinates, we propose to tackle this problem from a new perspective: learn a set of quadratic terms based static and global reference surfaces to describe 3D shapes, such that the coordinates of a 3D point (x, y, z) can be extended to quadratic terms (xy, xz, yz, . . .) and transformed to the relationship between the local point and the global reference surfaces. Then, the static surfaces are changed into dynamic surfaces by adaptive contribution weighting to improve the descriptive capability. Towards this end, we propose our point-to-surface representation, a new representation for 3D point cloud learning that has not been attempted before, which can assemble local and global geometric information effectively by building connections between the point cloud and the learned reference surfaces. Given 3D points, we show how the reference surfaces are constructed, and how they are inserted into the 3D learning pipeline for different tasks. The experimental results confirm the effectiveness of our new representation, which has outperformed the state-of-the-art methods on the tasks of 3D classification and segmentation.**

***Index Terms*—3D representation, point cloud segmentation, point cloud classification, 3D deep learning.**

I. Introduction

THREE-DIMENSIONAL deep learning has attracted extensive attentions in recent years, including but not limited to, 3D classification [1], [2], 3D segmentation [3], [4], shape completion [5], object detection [6], and 3D scene understanding [7]. The learning of 3D point cloud is still facing many challenges, due to the difficulties regarding the inference of the underlying shapes from irregular point clouds. Many works have been proposed to tackle this problem, from volumetric representation that converts 3D shape into 3D grids for 3D CNN [8], [9], to multi-view image representation [10], [11] and then to directly point cloud processing that feeds the points to the network inputs [1], [3], [12].

Manuscript received March 26, 2021; revised June 6, 2021 and June 21, 2021; accepted June 30, 2021. Date of publication July 6, 2021; date of current version May 5, 2022. This work was supported by the National Natural Science Foundation of China under Grant 62071097, Grant 61872067, and Grant U20A20184. This article was recommended by Associate Editor L. Zhang. *(Corresponding author: Guanghui Liu.)*

The authors are with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: tiechengsun@std.uestc.edu.cn; guanghuiliu@uestc.edu.cn; liru@std.uestc.edu.cn; liushuaicheng@uestc.edu.cn; eezsy@uestc.edu.cn; eezeng@uestc.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSVT.2021.3095233.

Digital Object Identifier 10.1109/TCSVT.2021.3095233

Qi *et al.* [1] proposed a point-wise feature learning network, PointNet, which consumes the points directly for 3D learning. However, though impressive, this design ignores the local structures that is important for 3D tasks [13]. Later, the PointNet++ [12] is proposed to use hierarchical networks to learn local features with increased contextual scales. In order to further extract topological information of local point clouds, Wang *et al.* [14] presented a Dynamic Graph Convolutional Neural Network (DGCNN) with Edge-conv. Then, based on hierarchical CNN like PointNet++ [12], Liu *et al.* [3] encoded the geometric priors of neighbors for high-level relation learning. However, these methods are limited to the relationship between point coordinates when extracting the local geometric features.

In general, there are three main challenges regarding the point cloud learning. First, how to capture good local and global features due to the disorder of point cloud. Second, how to represent effective geometric information of the underlying shape by giving the point coordinates. Third, how to calculate point cloud features in a simple and effective way. These challenges are open problems that are under exploration mainly by modifying the network architectures and designs [3], [12], [14]. In this work, we tackle the issue from a new perspective, where we modify the representation at the beginning.

To this end, we propose our quadratic terms based point-to-surface 3D representation considering both the point and the geometric surface simultaneously. Fig. 1 shows some simple examples. The surfaces to the left of the dashed line are defined as the reference surfaces. Each surface can be expressed as $f(x, y, z) = 0$. The function $f(x, y, z)$ is defined as a surface function, which is composed of the first and second order terms $(x, y, z, xy, xz, yz, x^2, y^2, z^2)$ and a constant. Fig. 1 gives three point clouds with simple shape, a sphere and two ellipsoids. In order to describe the point cloud locally and globally, we use the point cloud of the entire dataset to learn the global reference surface. Then, all points in the point cloud are taken as inputs of the global surface function $f(x, y, z)$. Through this operation, if a point $\mathbf{p} = (x, y, z)$ is on the surface, then $f(\mathbf{p}) = 0$. If all the function values of local points are equal to zero, then the surface $f(x, y, z) = 0$ is part of the input model around these local points. Otherwise, the value of $f(x, y, z)$ is not equal to zero but depends on the distance. As shown in Fig. 1, when the point cloud coincides with the surface, it appears red. If the point is moving away from the surface, it appears

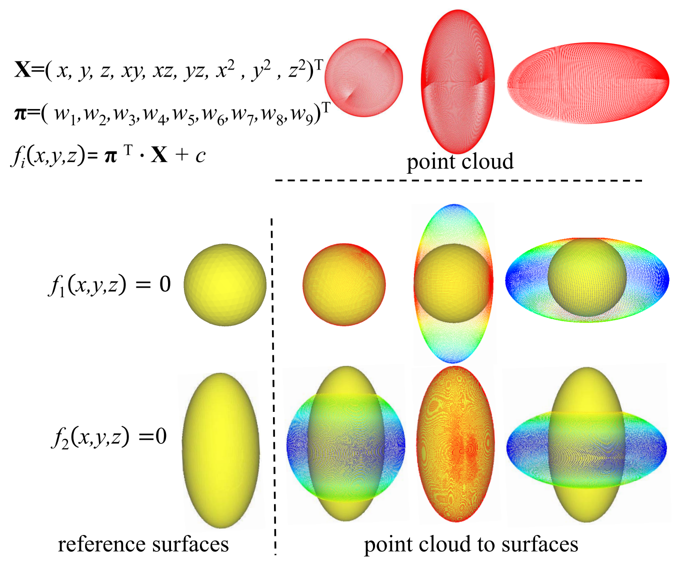

$\mathbf{X}=(x, y, z, xy, xz, yz, x^2, y^2, z^2)^T$
$\boldsymbol{\pi}=(w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9)^T$
$f_i(x,y,z)=\boldsymbol{\pi}^T \cdot \mathbf{X} + c$
point cloud
$f_1(x,y,z)=0$
$f_2(x,y,z)=0$
reference surfaces
point cloud to surfaces

Fig. 1. The shape of a point cloud can be described by comparing with reference surfaces. In order to represent the local and global geometrical information for a point **p**, the coordinates are extended to high-order and weighted by every different surface function coefficients to obtain function values. These values include local and global features. Then, they are used for point representation.

in a gradient of other colors. Therefore, the function values of local points reflect the local features. Moreover, the surface $f(x, y, z) = 0$ is learned through the entire point clouds, it is a global shape, so the surface function value $f(\mathbf{p})$ of point $\mathbf{p} = (x, y, z)$ can also indicate the relative position relationship between the local point and the global surface. Therefore, both local and global information is implicit in this representation.

In this paper, the reference surfaces are determined by learning the coefficients for the terms of $f(x, y, z)$, as shown in Fig. 1 ($w_1 \sim w_9$, c). These coefficients do not require special supervised learning. They can be easily learned within only one convolution layer in the specific 3D tasks training. In this process, we can learn N_f reference surfaces at one time. Then, every point in the point cloud can be fed into the N_f surface functions, resulting N_f point-wise representation as the input of the subsequent networks for specific 3D analysis. After training, these coefficients are stored in the checkpoint. For the testing data, however, these surfaces are static, which limits the richness and description capability of the reference surfaces. We prefer dynamic surfaces to describe different shapes accordingly. Therefore, we utilize the initial representation to learn the contribution weights of each surface to each point and weight the initial representation to implement the dynamic representation.

In summary, our main contributions are as follows:

- We propose a quadratic terms based point-to-surface representation for point cloud learning, which addresses the point cloud representation from a new perspective.
- We propose a dynamic representation based on the adaptive contribution weighting to improve the descriptive capability of the point-to-surface representation.
- The new representation can effectively grasp local and global features, facilitating the point cloud analysis.
- The proposed point-to-surface module is plug-and-play, which can be inserted into various 3D pipelines seamlessly, creating new state-of-the-art performances, validated on 3D classification and segmentation tasks.

II. Related Works

A. Depth Image Representation

Compared to point cloud, depth image is orderly and regular. Depth image can be easily obtained by stereo disparity estimation [15], monocular depth estimation [16] and some depth camera such as Kinect [17]. Therefore, depth-based 3D tasks are widely concerned. For example, Soltam *et al.* [18] proposed to synthesize 3D shapes via multi-view depth maps with generative network. Ren *et al.* [7] proposed monocular depth estimation for 3D indoor and outdoor scene understanding. Some works [19], [20] studied the pose estimation using depth image directly. In contrast, Chen *et al.* [21] proposed a method that converts the depth image into point cloud with the intrinsic parameters of the camera. Then, they reconstructed the encoder of the SO-Net [22] to estimate the hand pose using point cloud. Yavartanoo *et al.* [23] projected the 3D model to multiple 2.5D depth maps for learning representations of 3D objects with a Stereographic Projection Neural Network (SPNet). However, depth-based methods are limited by the accuracy of depth estimation.

B. Mesh Representation

Mesh is very difficult to extract shape features due to the complexity and irregularity. Han *et al.* [24] proposed mesh convolutional restricted Boltzmann machines to simultaneously learn local and global features from the mesh data. For 3D mesh segmentation, Le *et al.* [25] rendered the mesh model into multiple views to learn edges with CNN. Then, these edges were unprojected back to 3D mesh for mesh segmentation. Although this method avoids the complexity and irregularity of mesh data through multi-view images, the original mesh information is still not used for feature extraction. Recently, Feng *et al.* [26] proposed a mesh neural network (MeshNet) for 3D shape representation using original mesh data. They regarded the triangular face as the unit. So the disorder problem is solved by per-face processing. Then, the geometrical information of triangular face are calculated for 3D task learning. The experiments demonstrate the effectiveness of MeshNet. Taha *et al.* [27] defined an inverse mapping between the 3D mesh and the 2D texture image. They mapped the 3D mesh features to 2D images for 3D shape representation. The method has been applied to facial expression and action recognition.

C. Multi-View Representation

Some researches proposed to render the 3D shape to multi-view images for 3D classification [28] and shape retrieval [29], [30]. Gadelha *et al.* [31] proposed to utilize multi-view images to restore 3D shape with projective generative adversarial networks. Considering the similarities and differences between multi-view images, Feng *et al.* [11] proposed a group-view CNN network for 3D shape recognition.

Different from the view-wise feature extraction, Yu *et al.* [10] aggregated the multi-view local features by bilinear pooling. Then, they harmonized these bilinear feature components for 3D representation. Huang *et al.* [2] proposed to render 3D object as multi-view images to learn a disentangled representation for 3D classification. These methods convert disordered 3D data into ordered 2D images. Then, the mature 2D scene understanding techniques are applied to 3D understanding.

D. Volumetric Grids Representation

Volumetric grids representation is another commonly used 3D representation. Wu *et al.* [8] first proposed ShapeNet to represent a geometric 3D shape as a probability distribution on volumetric grid. Following the ShapeNet [8], Sharma *et al.* [9] presented an unsupervised learning volumetric representation for denoising, shape completion and classification. Wu *et al.* [32] proposed 3D generative adversarial network based on volumetric convolutional networks for 3D shapes generation. Xu *et al.* [33] extended the voxel hashing to point cloud voxel representation. They used an efficient multi-scale voxel representation for collision detection in augmented reality scene. In order to extract the point-wise feature, Zhou and Tuzel [34] proposed VoxelNet to partition the space into voxels and aggregate the point-wise feature in each voxel to shape information as voxel-wise feature. Finally, the voxel-wise feature can be used for object detection. Le and Duan [35] proposed the PointGrid to integrate point and grid for preserving the original coordinates. Signed Distance Function (SDF) is also commonly used for voxel-based representation [36], [37]. SDF values are calculated for each voxel. Then the zero iso-surface represents the shape information that can be used for shape completion and 3D reconstruction. However, these voxel-based methods require a lot of memory and computation. Moreover, the representation power is limited by voxel resolution.

E. Point-Wise Representation

Recently, Qi *et al.* [1] proposed the PointNet to directly extract the point-wise features. Then a symmetric function was used for extracting the global feature. However, PointNet does not capture local structures. To address this problem, Qi *et al.* [12] also proposed a hierarchical neural network to extract the local features of multi-scale neighborhood points. Achlioptas *et al.* [38] proposed a learning representation method based on encoder and decoder network for transforming point into a latent space. Wang *et al.* [39] proposed a parametric continuous convolution to extract points features. Considering the relationship between neighborhood points, Wang *et al.* [14] proposed Dynamic Graph CNN (DGCNN) to expand the receptive field with dynamic graph updates. Inspired by DGCNN, Wang and Solomon [40] utilized DGCNN to align two point clouds. Liu *et al.* [3] proposed a Relation-Shape Convolutional Neural Network (RSCNN) for feature learning from the relation between neighbor points.

However, when extracting shape features, these methods only use the coordinates to extract the relationship between points, instead of analyzing the features between the point and surface. Therefore, in this paper, we propose to use the quadratic terms of the point coordinates and the learned reference surfaces to construct a new point-to-surface representation without loss of information, which is different from any of the previous 3D representations mentioned above.

III. METHOD

In this section, quadratic terms based point-to-surface representation is first formulated and proposed (Sec. III-A). In order to enhance the description capability of local characteristics, an adaptive contribution weighting optimization is proposed (Sec. III-B). Then, we introduce our point-to-surface module (Sec. III-C). After that, we apply the point-to-surface module to the previous network structures for classification and segmentation (Sec. III-D). Finally, we introduce the detailed training process (Sec. III-E).

A. Quadratic Terms Based Point-to-Surface Representation

From the point of view of human perception, when we perceive an object, we usually look at the appearance of the object and compare it with our reference models accumulated through our past experience. Then, we can determine if the object has been seen. If so, what is it? Based on this understanding, this paper proposes to use the reference surface to describe the 3D object in the form of point cloud. In the real world, no matter from which angle we look at an object, we cannot see the whole picture of it. This is because objects have front and back views in any perspective. Therefore, if a function can represent a 3D shape, the function must be quadratic. For example, if the shape is a sphere, the surface can be expressed as:

$$x^2 + y^2 + z^2 - r = 0, \tag{1}$$

where r is the radius of the sphere. For a ruled quadrics surface [41], the equation is

$$\text{xy-z} = 0. \tag{2}$$

More generally, these quadratic terms based surfaces can be formulated as:

$$\begin{aligned} \mathbf{X} &= (x, y, z, xy, xz, yz, x^2, y^2, z^2)^T, \\ \pi &= (w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9)^T, \\ f(x, y, z) &= \pi^T \cdot \mathbf{X} + c = 0, \end{aligned} \tag{3}$$

where the vector $\mathbf{X}$ represents the terms of the function $f(x, y, z)$, defined as quadratic terms, the notation π represents the coefficients of these terms, and c is a constant. Here, we use $F(x, y, z) = 0$ to represent a set of reference surfaces $f_i(x, y, z) = 0$ with the number of N_f, $i \in [1, N_f]$. If we represent the reference surface as a function $f(x, y, z)$, the value of the function $f(x, y, z)$ is related to the distance between the point and the surface. For example, if $f(x_i, y_i, z_i) = 0$, then the point $\mathbf{p}_i = (x_i, y_i, z_i)$ is on the surface; otherwise, it is not on the surface. The function $f(x, y, z)$ corresponding to a continuous surface is also continuous, so the value of $f(x, y, z)$ represents the relationship between a 3D point $\mathbf{p} = (x, y, z)$ and the surface $f(x, y, z) = 0$. For the surface

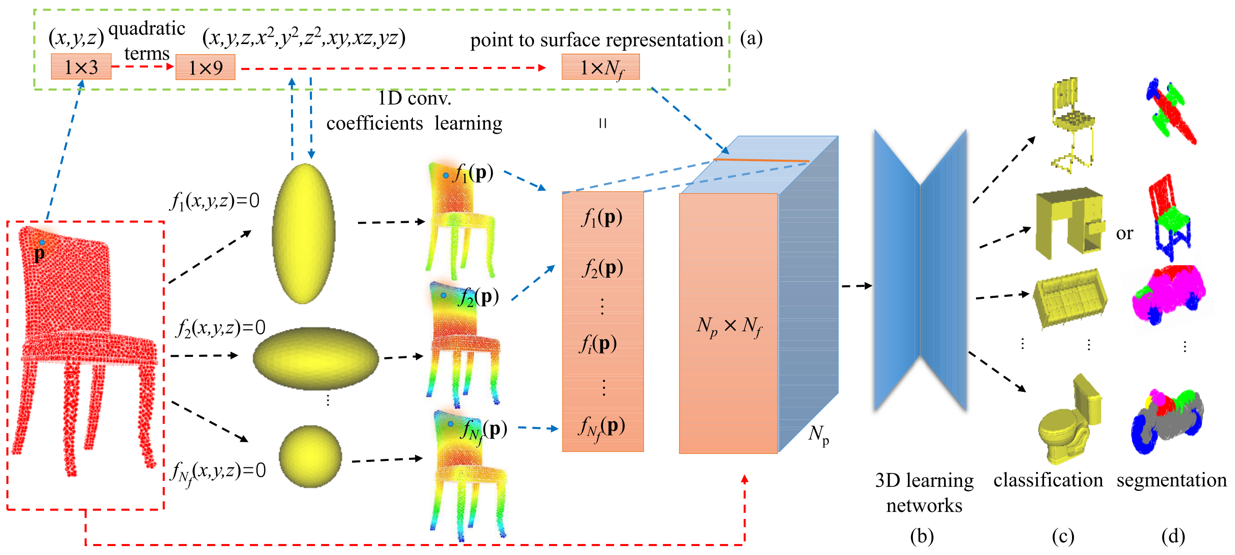


Fig. 2. The pipeline of point-to-surface representation. The quadratic terms are calculated by the point coordinates. Then a 1D convolution is used for learning the function coefficients. For a point $\mathbf{p}$, feeding quadratic terms into convolution layer can obtain its point-wise representation with the size of $1 \times N_f$. For a point cloud with N_p points, the surface functions yield a $N_p \times N_f$ feature. Then, this feature is inserted into 3D learning networks for point cloud analysis.

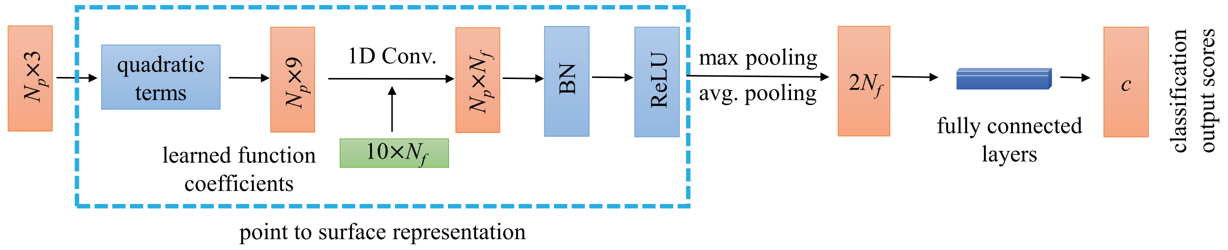


Fig. 3. The classification baseline based on the proposed point-to-surface representation. The baseline first calculates the quadratic terms by giving point coordinates. Then, the quadratic terms are transformed to point-to-surface representation through a 1D convolution layer, a batch normalization (BN) layer and a ReLU layer successively. Finally, the classification scores are obtained by the following pooling layers and fully connected layers.

function set $F(x, y, z)$, all the function values form a new representation, formulated as:

$$\mathbf{A}_{N_f \times 10}\bar{\mathbf{X}} = \mathbf{R}, \tag{4}$$

where the matrix $\mathbf{A}_{N_f \times 10}$ is made up of N_f surface parameters. The vector $\bar{\mathbf{X}}$ is the homogeneous expression of $\mathbf{X}$. The vector $\mathbf{R}$ is the new representation with the size of N_f. The 9 quadratic terms are considered as 9 unknowns. If the rank of matrix $\mathbf{A}_{N_f \times 10}$ is equal to 9, Equation (4) has a unique solution. Assuming that these surface functions are linearly uncorrelated, if the number of surface functions is greater than 9, i.e., $N_f \geq 9$, the original unique point coordinates can be solved with the new representation $\mathbf{R}$ and the surface parameters $\mathbf{A}_{N_f \times 10}$. Therefore, $N_f \geq 9$ is a sufficient condition for no-loss representation.

Fig. 2 shows our pipeline. Given a 3D point $\mathbf{p} = (x, y, z)$, the quadratic terms can be directly calculated by the coordinates of points while the coefficients of the terms need to be estimated. Here, a surface function contains 9 coefficient weights ($w_1 \sim w_9$) and one bias c as unknowns to be estimated. To estimate a surface function $f_i(x, y, z)$, we use the entire point clouds from dataset for the estimation. The coefficients estimation is implemented through the 3D tasks training. Instead of estimating a single surface function f_i one at a time, we estimate a set of N_f functions all at once. As such, we estimated $N_f \times 10$ unknowns for all surface functions. After N_f surfaces being learned, a point $\mathbf{p}$ can be calculated for its new features, yielding a feature of size $1 \times N_f$ (Fig. 2(a)). For a point cloud with N_p points, yielding a feature of size $N_p \times N_f$, which are inserted into 3D learning networks for subsequent tasks (Fig. 2(b)), such as classification and segmentation (Fig. 2(c) and (d)). Based on this pipeline, we define a classification baseline as shown in Fig. 3. The baseline first calculates the new representation with a 1D convolution layer, a Batch Normalization (BN) layer and a ReLU layer. Then, the representation is fed into pooling layers and fully connected layers to get classification scores.

B. Adaptive Contribution Weighting

When the network training is finished, the learned global reference surfaces have been fixed. They are static for the testing data, which limits the richness and the descriptive capability of the reference surfaces for various shapes. To address this problem, we propose an adaptive contribution weighting optimization to weight the initial representation, equivalent to weighting the original reference surfaces. Then, we can use these dynamic reference surfaces to describe different shapes accordingly.

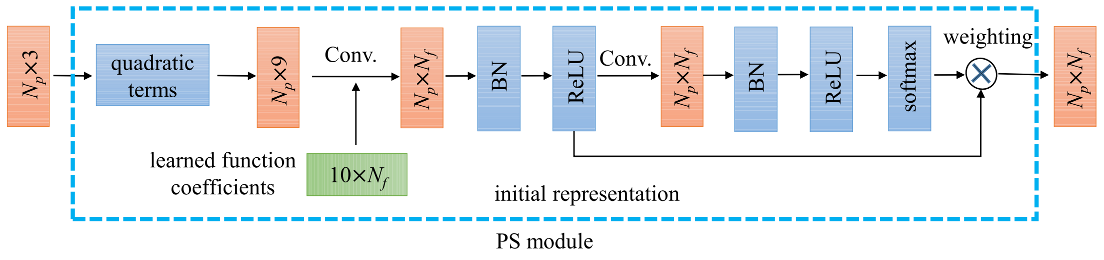


Fig. 4. The structure of PS module. The PS module first calculates the initial point-to-surface representations as the baseline. Then, the initial representations are used to calculate the contribution weights. Finally, the initial representations are weighted by the calculated weights to get the final point-to-surface representation.

Specifically, we convert the initial point-to-surface representation into contribution weights through the convolution layers. This is because the initial point-to-surface function value is related to the distance from the point to the surface, implying the relationship between the point and the reference surface. In addition, this information is different from each point for every surface, so this point-wise information can be used for adaptively learning the contribution of each reference surface to each point. The contribution learning model includes a convolution layer with the kernel size of 1, a BN layer, a ReLU layer and a softmax layer. The size of the learned contribution weights is the same as the initial representation. In this way, we use the learned weights to weight the initial representation to get the final point-to-surface representation. This method is equivalent to changing the static and global surfaces into dynamic global surfaces. Since the learned weights are different for each point, this dynamic representation can further enhance the local characteristics.

C. Point-to-Surface Module

To learn the reference surface functions, all point-wise quadratic terms are fed into a 1D convolution with the kernel size of 1. The weights and bias in 1D convolution are surface coefficients and constant. Therefore, the input channel of the 1D convolution is the number of function terms, i.e., 9. We estimate a set of N_f surface functions. Accordingly, the output channel of the 1D convolution is set to be N_f. Then, for a point $\mathbf{p}$, the outputs of the 1D convolution are N_f values, corresponding to N_f surface function values. For a point cloud with N_p points, after 1D convolution, the initial representation is extracted based on the quadratic terms and reference surfaces, yielding a feature of size $N_p \times N_f$. Considering that $f(x, y, z)$ may have a domain, to fit the boundary, an activation function is used for activating the output of the convolution.

For adaptive contribution weighting, the initial representation is fed into a new 1D convolution layer with the kernel size of 1. Both the input and output channels are set to N_f. Then a BN layer, a ReLU layer and a softmax layer are followed to get the contribution weights. Finally, the initial representation is weighted by these weights to get the final dynamic representation.

We call the above implementation a Point to Surface (PS) module as shown in Fig. 4. The input of the PS module is point cloud, and the output is the new representation. Note that, the PS module is plug-and-play, which can be inserted into the existing 3D learning pipelines seamlessly, where the reference surface function coefficients in the module are learned with its binding pipeline during the learning of specific 3D tasks. Different tasks with different pipelines may result in different surface function parameters for their own optimization.

D. 3D Classification and Segmentation

Extracting the local and global shape information of point cloud is important for 3D classification and segmentation. However, the traditional method extracts the point-wise feature solely based on point coordinates (x, y, z), which may under-fitting for 3D surfaces. Here, we use our point-to-surface representation.

We plug our PS module into two representative networks for point cloud classification and segmentation. The two networks are dynamic graph network DGCNN [14] and RSCNN [3]. DGCNN [14] is based on dynamic graph structure for extracting topology information. RSCNN [3] inherits the PointNet++ [12] hierarchical structure while learning the geometric information of the neighborhood at a high-level. The modified DGCNN is shown in Fig. 5. The classification and segmentation networks modified by RSCNN are shown in Fig. 6 and Fig. 7, respectively.

As for DGCNN [14] and RSCNN [3], they both extract k neighborhoods for each point as local geometrical information by k-nearest neighbors searching or ball-query searching. As shown in Fig. 5, if the neighborhood is obtained in feature space, we first calculate the quadratic terms. Then, these terms are used for neighbor searching (Fig. 5(a)). If the neighborhood is obtained by point coordinates, the quadratic terms calculation follows the neighbor searching (Fig. 5(b)). After the neighbor searching, we get a feature with the size of $k \times N_p \times N_f$. Therefore, the function coefficients should be learned by 2D convolution in PS modules.

In the modified DGCNN, the point cloud is first transformed into the point-to-surface representation with our PS module for the classification. For the segmentation, the PS module follows the spatial transformation. After PS module, the new representation is fed into the dynamic graph network (Fig. 5(c)) for classification and segmentation. For the details of DGCNN, please refer to [14]. The modified RSCNN for classification is shown in Fig. 6. The hierarchical architectures input different number of points. These point sets are first represented by our point-to-surface representation through a

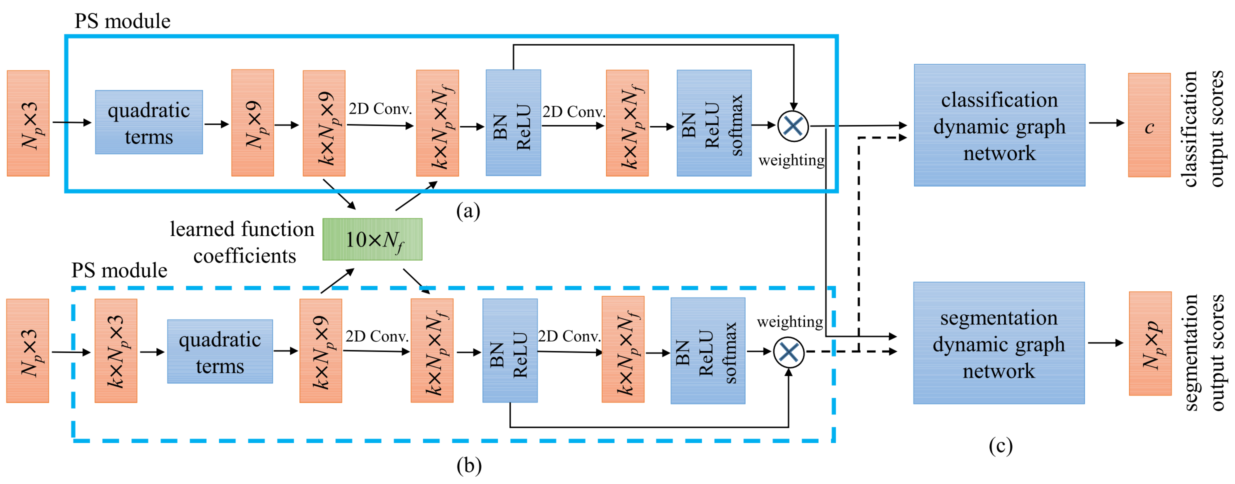


Fig. 5. The modified DGCNN [14] with the proposed point-to-surface representation for classification and segmentation. For classification, the input of the PS module is the original point cloud. While for segmentation, the input is the transformed point cloud with non-Euclidean coordinates. The nearest-neighbor searching can be based on point coordinates or features, so the PS module has two forms as shown in (a) and (b). (N_p: the number of the input points. N_f: the number of the reference surfaces. c: the number of the categories. p: the number of the segmentation parts.).

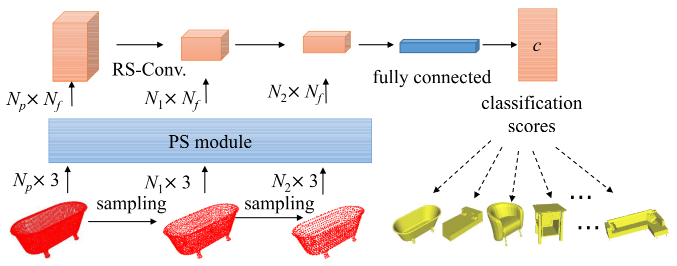


Fig. 6. The modified RSCNN [3] with the proposed point-to-surface representation for classification. Different from the original structure, the modified network no longer extracts the feature of point coordinates directly, but extracts the feature of the new representation obtained through PS module.

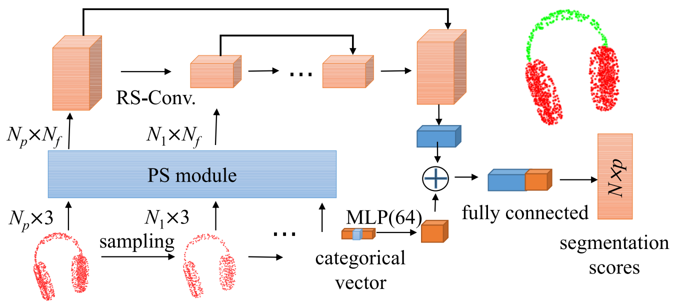


Fig. 7. The modified RSCNN [3] with the proposed point-to-surface representation for segmentation. Different from the original structure, the point Euclidean coordinates are first transformed to new representation through PS module for further feature extraction.

PS module as illustrated in Fig. 5(b). Then, this representation is used for classification features learning with RS-Conv. [3]. Finally, the fully connected layers follows the network to obtain the classification scores. The modified RSCNN for segmentation is shown in Fig. 7. Sampled multiple point sets are also fed into our PS module to get the new representation. Then, this representation is used for extracting local and global features through RS-Conv. [3] and skip connection. The features concatenate the category feature. This category feature is obtained by extending the one-hot categorical vector with a MLP layer as the strategy in [14]. Finally, the concatenated features are used for learning the segmentation scores.

E. Implementation Details

The network is trained on a 1080 Ti GPU device. For the modified DGCNN, we use Stochastic Gradient Descent (SGD) optimizer to minimize the classification and segmentation loss function [14]. For the modified RSCNN, the optimizer is Adam. In classification experiments, the epoch is set to 400 for full convergence. When the input shape has $2,048$ sampling points, the batch size is $B = 8$. While for $1,024$ points input, the batch size is $B = 16$. In segmentation experiments, the epoch is set to 200 for full convergence. The batch size is set to $B = 16$. The number of the reference surface functions $F(x, y, z)$ is set to 64. Our method is implemented by Pytorch. The modified networks are implemented based on the public codes of DGCNN [14] and RSCNN [3].

IV. Experiments

In this section, we first introduce the experimental datasets. Then, the baseline of our method is analyzed in terms of effectiveness, classification performance and complexity. Next, we evaluate the point-to-surface representation on 3D classification, part segmentation and semantic segmentation. After that, we analyze the proposed method through ablation study. Finally, we visualize the learned reference surfaces and the point-to-surface representation.

A. Experimental Datasets

We evaluate our method on popular 3D classification benchmark dataset, *i.e.* ModelNet10 (MN10) and ModelNet40 (MN40) [8]. Both datasets are CAD models. In the experiment, we sample $1,024$ or $2,048$ points uniformly. ModelNet40 dataset has 40 categories with $9,843$ training samples and $2,468$ testing samples. While ModelNet10 has 10 categories with $3,991$ training samples and 908 testing samples. For shape part segmentation, we use the popular ShapeNet part benchmark [42] to evaluate our method. This dataset has

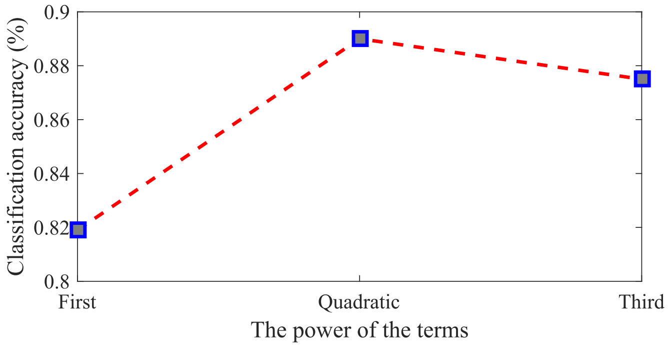


Fig. 8. The classification accuracy on different power of terms. The first and third order terms results are obtained by replacing the quadratic terms of the baseline model by the first and third order terms, respectively.

16 categories with 50 parts. In the experiment, 2, 048 points are selected as inputs. For semantic segmentation, we use the popular ScanNet [43] benchmark for evaluation.

B. Point to Surface Representation Evaluation

We first illustrate the effectiveness of our point-to-surface representation by comparing quadratic terms with the first and third order terms based on our baseline. Then, we extend our baseline with popular network structures such as DGCNN [14] and RSCNN [3] for 3D classification and shape part segmentation. Finally, we extend the PS module to FPConv [44] for semantic segmentation. The performance of our method is analyzed by comparing the results that before and after modification.

1) Analysis of Baseline: We first validate effectiveness of the proposed quadratic terms based baseline as shown in Fig. 3. We compare them with the first and third order terms. The first order terms include x, y and z. The third order terms include all the first and quadratic terms and the terms $x^3, xyz, xy^2, \dots,$ totally 19 terms. The quadratic terms in Fig. 3 are respectively replaced by the first and third order terms to obtain the classification accuracy as shown in Fig. 8. The result shows that quadratic terms based method outperforms the other two methods. In particular, the quadratic terms have a significant performance improvement over the first order terms. The classical methods such as PointNet [1], PointNet++ [12] and DGCNN [14] all belong to the first-order-terms based methods. This indicates that the quadratic terms are more descriptive than the traditional first order terms. This also proves that the physical implication of learning based on quadratic terms is more reasonable than that of first order terms. The learning based on first order terms can be considered as point cloud description on reference plane.

As shown in Fig. 3, our classification baseline has only one convolution layer to learn the function coefficients. Nevertheless, it has a strong ability to classify the 3D objects. Table I lists the classification accuracy of our method and the other state-of-the-art methods. The results show that the accuracy of our baseline is 89.5% on MN40, which is higher than the 89.2% of multiple-layers PointNet [1]. On MN10, our baseline achieves a accuracy of 94.0%, higher than PointNet++'s accuracy of 93.3%.

TABLE I

3D OBJECT CLASSIFICATION RESULT (%) ON MODELNET40 (MN40) AND MODELNET10 (MN10). THE BEST ONES ARE MARKED IN RED, AND THE SECOND BEST ONES ARE IN BLUE ("-": UNKNOWN)

Method	points	MN40	MN10
Pointwise-CNN [45]	1k	86.0	-
Deep Sets [46]	1k	87.1	-
ECC [47]	1k	87.4	90.0
PointNet [1]	1k	89.2	91.9
Baseline (Ours)	1k	89.5	94.0
SCN [48]	1k	90.0	-
Flex-Conv [49]	1k	90.2	-
Kd-Net(depth=10) [50]	1k	90.6	-
PointNet++ [12]	1k	90.7	93.3
KCNet [51]	1k	91.0	94.4
MRTNet [52]	1k	91.2	-
Spec-GCN [53]	1k	91.5	-
PointCNN [54]	1k	91.7	-
PCNN [55]	1k	92.3	-
RSCNN (single-scale) [3]	1k	92.3	94.2
RSCNN+PA [56]	1k	92.7	96.0
KPConv [57]	1k	92.7	-
RSCNN+PS (Ours)	1k	92.9	96.1
PointASNL [58]	1k	92.9	95.7
DGCNN [14]	1k	92.9	94.8
DensePoint [59]	1k	32.2	96.6
DGCNN+PA [56]	1k	93.4	96.7
DGCNN+PS (Ours)	1k	93.5	96.4
SO-Net [22]	2k	90.9	94.1
PointNet++ [12]	5k	91.9	-
SpiderCNN [60]	5k	92.4	-
DGCNN [14]	2k	93.5	95.5
DGCNN+PS (Ours)	2k	94.2	96.5

TABLE II

THE COMPARISON OF COMPLEXITY OF DIFFERENT METHODS

Method	params	FLOPs/sample
PointNet [1]	3.5 M	440 M
PointNet++ [12]	1.48 M	1684 M
Baseline	0.4 M	3.2 M
DGCNN+PS	1.818 M	2586 M
DGCNN [14]	1.813 M	2484 M
PS	4992	102 M

Meanwhile, the baseline is very lightweight and fast. As recorded in Table II, our baseline has approximately 0.4 M parameters, an order of magnitude lower than PointNet [1]. The flops of our baseline for one sample is 3.2 M, two orders of magnitude lower than PointNet [1] and approximately 0.2% of PointNet++ [12]. Therefore, this baseline is suitable for massive point clouds or a low computing power device. With regard to our PS module, we use DGCNN [14] network to calculate the additional parameters and flops. As shown in Table II, the original DGCNN has 1.813 M parameters and 2484 M flops. After adding our PS module, the increase is 4992 and 102 M, respectively, which is an increase of 0.28% and 4.11% over the original DGCNN. This indicates that our PS module also has lower calculation cost.

2) 3D Classification: To further verify the effectiveness of our method, we expand the baseline with DGCNN [14] (Fig. 5) and RSCNN [3] (Fig. 6) networks for classification. We compare our method with the state-of-the-art methods on MN40 and MN10 datasets. Table I shows the classification

TABLE III

The Comparison Result (%) of Shape Segmentation on ShapeNet [42]. Best Ones Are Marked in Red, and the Second Best Ones Are in Blue.DGCNN-T: Spatial Transform Is Removed From DGCNN [14]

Method	class mIoU	instance mIoU	air plane	bag	cap	car	chair	ear phone	guitar	knife	lamp	laptop	motor bike	mug	pistol	rocket	skate board	table
Kd-Net [50]	77.4	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3
PointNet [1]	80.4	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
RS-Net [61]	81.4	84.9	82.7	86.4	84.1	78.2	90.4	69.3	91.4	87.0	83.5	95.4	66.0	92.6	81.8	56.1	75.8	82.2
SCN [48]	81.8	84.6	83.8	80.8	83.5	79.3	90.5	69.8	91.7	86.5	82.9	96.0	69.2	93.8	82.5	62.9	74.4	80.8
PCNN [55]	81.8	85.1	82.4	80.1	85.5	79.5	90.8	73.2	91.3	86.0	85.0	95.7	73.2	94.8	83.3	51.0	75.0	81.8
SPLATNet [62]	82.0	84.6	81.9	83.9	88.6	79.5	90.1	73.5	91.3	84.7	84.5	96.3	69.7	95.0	81.7	59.2	70.4	81.3
KCNet [51]	82.2	84.7	82.8	81.5	86.4	77.6	90.3	76.8	91.0	87.2	84.5	95.5	69.2	94.4	81.6	60.1	75.2	81.3
PointASNL [58]	83.4	86.1	84.1	84.7	87.9	79.7	92.2	73.7	91.0	87.2	84.2	95.8	74.4	95.2	81.0	63.0	76.3	83.2
KPConv [57]	85.1	86.4	84.6	86.3	87.2	81.1	91.1	77.8	92.6	88.4	82.7	96.2	78.1	95.8	85.4	69.0	82.0	83.6
DensePoint [59]	84.2	86.4	84.0	85.4	90.0	79.2	91.1	81.6	91.5	87.5	84.7	95.9	74.3	94.6	82.9	64.6	76.8	83.7
DGCNN [14]	82.3	85.2	84.0	83.4	86.7	77.8	90.6	74.7	91.2	87.5	82.8	95.7	66.3	94.9	81.1	63.5	74.5	82.6
DGCNN+PS	80.9	85.5	80.9	81.9	88.0	81.3	90.7	68.2	91.7	87.1	84.9	91.6	64.0	93.7	78.0	53.3	74.2	84.6
DGCNN-T	80.0	85.1	82.1	77.9	80.3	77.2	90.9	73.6	90.9	88.5	85.2	95.6	56.2	93.8	78.4	50.4	74.6	83.5
DGCNN-T+PS	80.7	85.4	81.0	84.9	94.5	83.6	90.1	56.6	91.6	87.4	83.3	96.4	67.7	91.8	74.1	56.2	68.0	84.4
RSCNN [3]	84.0	86.2	83.5	84.8	88.8	79.6	91.2	81.1	91.6	88.4	86.0	96.0	73.7	94.1	83.4	65.5	77.7	83.6
RSCNN+PS	84.5	86.5	84.3	84.5	95.5	82.2	91.0	68.5	90.6	83.4	85.7	91.7	77.5	94.7	85.3	73.3	79.7	84.7

accuracy of popular methods and the modified versions equipped with our representation. As shown in Table I, compared with RSCNN [3], our representation improves the accuracy by 0.6% and 1.9% on MN40 and MN10, respectively, higher than that of PointAugment (PA) [56]. On the basis of the performance of DGCNN [14], our method improves by 0.6% and 1.6% on MN40 and MN10, respectively, outperforming the other methods on MN40. The improvement is close to that of PA [56]. When the input number of points is 2 k, our method has an accuracy of 94.2% on MN40 and 96.5% on MN10, superior to the state-of-the-art methods on both datasets. Compared with other methods, our performance improvement is significant. These results indicate that the point-to-surface representation plays a significant and positive role in improving the accuracy of 3D classification.

3) Shape Part Segmentation: In order to verify the effectiveness of our PS module on point cloud segmentation, we also modify the DGCNN [14] and RSCNN [3] networks with our representation for comparison. The modified DGCNN is shown in Fig. 5. The modified RSCNN is shown in Fig. 7. We use the popular ShapeNet part benchmark [42] to evaluate our method and the state-of-the-art methods. Referring to [3], the number of the input points is $2,048$. The methods are evaluated by mean Inter-over-Union (mIoU). Table III records the mean IoU, class IoU and every category IoU of different methods.

When we directly add the PS module into the DGCNN [14], the results in Table III show that the instance mIoU increases by 0.3% but the class mIoU decreases by 1.4%. One possible reason is that the spatial transformation of DGCNN [14] changes the original Euclidean space, causing the proposed representation meaningless. In order to verify this point, we remove the spatial transformation from the above comparative experiments, so that the coordinates are in Euclidean space. In Table III, the two experiments are denoted by DGCNN-T+PS and DGCNN-T, respectively. The results show that the modified DGCNN improves the class mIoU and instance mIoU by 0.7% and 0.3%, respectively, which is significant. This result indicates that our PS module is effective for the traditional Euclidean spatial point cloud but not for the transformed non-Euclidean space. For the traditional networks based on Euclidean coordinates, such as RSCNN [3], after embedding our PS module, the class mIoU and instance mIoU are 84.5% and 86.5%, respectively. The instance mIoU is superior to the other methods. Compared with RSCNN [3], our method improves the class and instance mIoU by 0.5% and 0.3%, respectively. Meanwhile, our method outperforms the others on multiple objects part segmentation as shown in Table III. In particular, the results also show that the performance of air plane, cap, car, motor bike, pistol, rocket, skateboard and table is improved significantly by using our

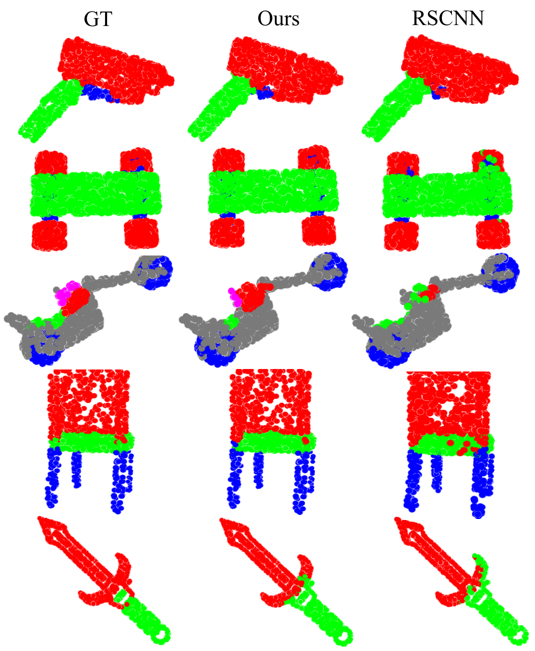


Fig. 9. The comparison of segmentation results between our method and RSCNN [3]. Different colors represent the segmentation results of different parts.

TABLE IV

THE COMPARISON RESULTS OF THE SEMANTIC SEGMENTATION ON SCANNET DATASET BETWEEN THE FPCONV [44] AND THE MODIFIED NETWORK WITH OUR PS MODULE. THE RESULTS INCLUDE SEGMENTATION MIOU, MEAN CLASS ACCURACY (MA) AND OVER ALL CLASS ACCURACY (OA) (%)

Model	mIoU	mA	oA
FPConv [44]	62.2	74.3	84.6
FPConv+PS	62.6	75.6	84.6

TABLE V

THE ACCURACY (%) OF 3D CLASSIFICATION ON MN40 WITH INITIAL POINT-TO-SURFACE REPRESENTATION (INIT.), ADAPTIVE CONTRIBUTION WEIGHTING (ACW)

Model	INIT.	ACW	MPOINTS	Acc.
DGCNN [14]				92.9
DGCNN+	✓			93.1
DGCNN+	✓	✓		93.5
DGCNN+	✓	✓	✓	94.2

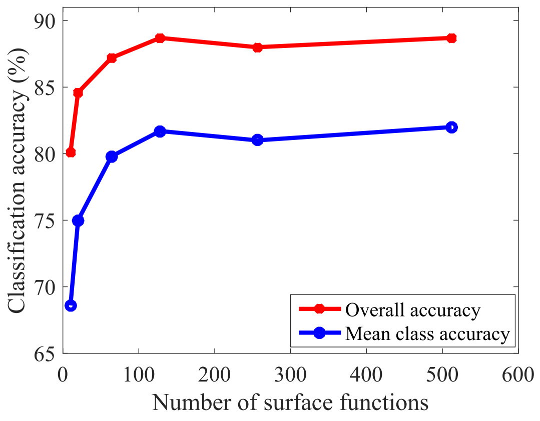


Fig. 10. Results of our baseline with different numbers of surface functions.

PS module. We also compare some segmentation results with RSCNN [3] as shown in Fig. 9. These results show that our method has better result for small parts segmentation. More results are in Appendix A. The above experimental results indicate that our point-to-surface representation can enhance the ability of network feature description in segmentation.

4) Semantic Segmentation: We also investigate the ability of the proposed point-to-surface representation for complex scenarios such as the semantic segmentation benchmark ScanNet [43]. For the comparison, we add the PS module to the network structure of FPConv [44] as the method of the modified RSCNN. The setting for the comparison experiments are the same except for the PS module. The quantitative results are recorded in Table IV. The results show that our PS module improves the mIoU and mean class accuracy of FPConv by 0.4% and 1.3%, respectively. This means that the point-to-surface representation is also suitable for feature extraction in complex scenario.

C. Ablation Study

In this section, our method is analyzed by ablation study in the application of 3D classification. The ablation study is performed on the network structure of DGCNN [14]. The results are recorded in Table V. The original DGCNN has

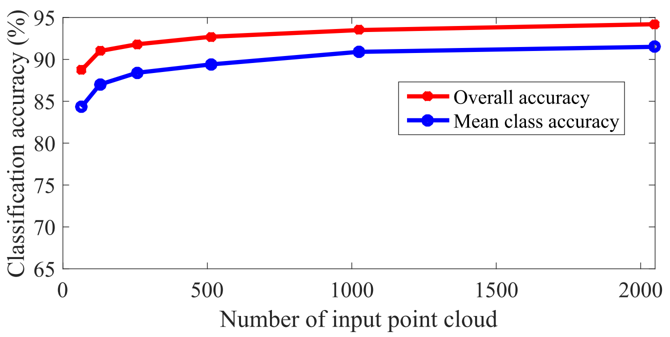


Fig. 11. Results of classification trained by 64, 128, 256, 512, 1,024 and 2,048 points with our method.

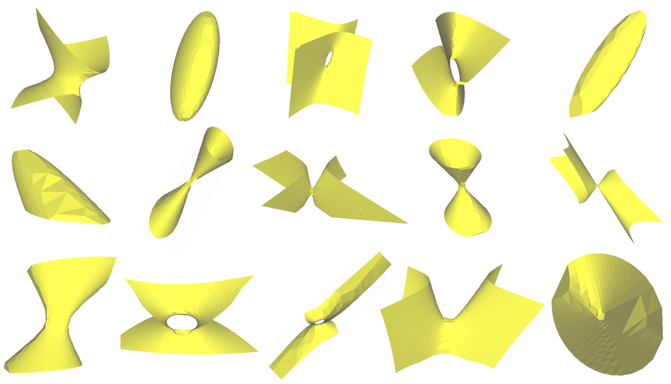


Fig. 12. Some learned surfaces with ModelNet40 benchmark. According to the surface function equation, the sampling point cloud can be obtained by giving x and y coordinates and solving the z coordinate. Then, the point cloud is used for mesh reconstruction with MeshLab [63] software.

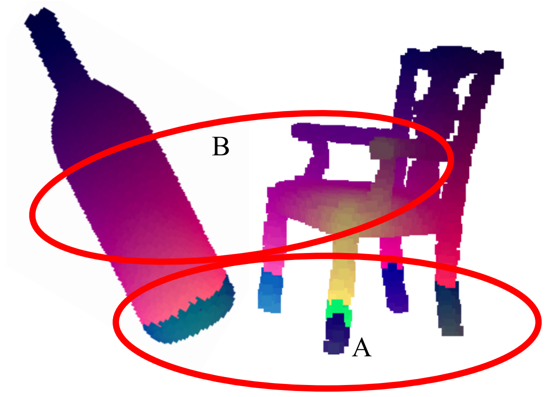


Fig. 13. Contribution visualization of a certain reference surface function. Region B shows the continuity of the surface function. Region A illustrates that different surface functions contribute differently to different parts of a 3D shape.

an accuracy of 92.9%. After embedding our initial point-to-surface representation without adaptive contribution weighting, the accuracy is 93.1%, improved by 0.2%. When we add the adaptive contribution weighting to DGCNN, the accuracy is improved by another 0.4%. Finally, in order to study the impact of the number of input points on performance, we increase the number of input points to 2 k for comparison. The accuracy has improved significantly, reaching 94.2%.

We also analyze the effect of the surface function number on performance. Fig. 10 shows the comparison results with different surface function number in baseline. The experimental

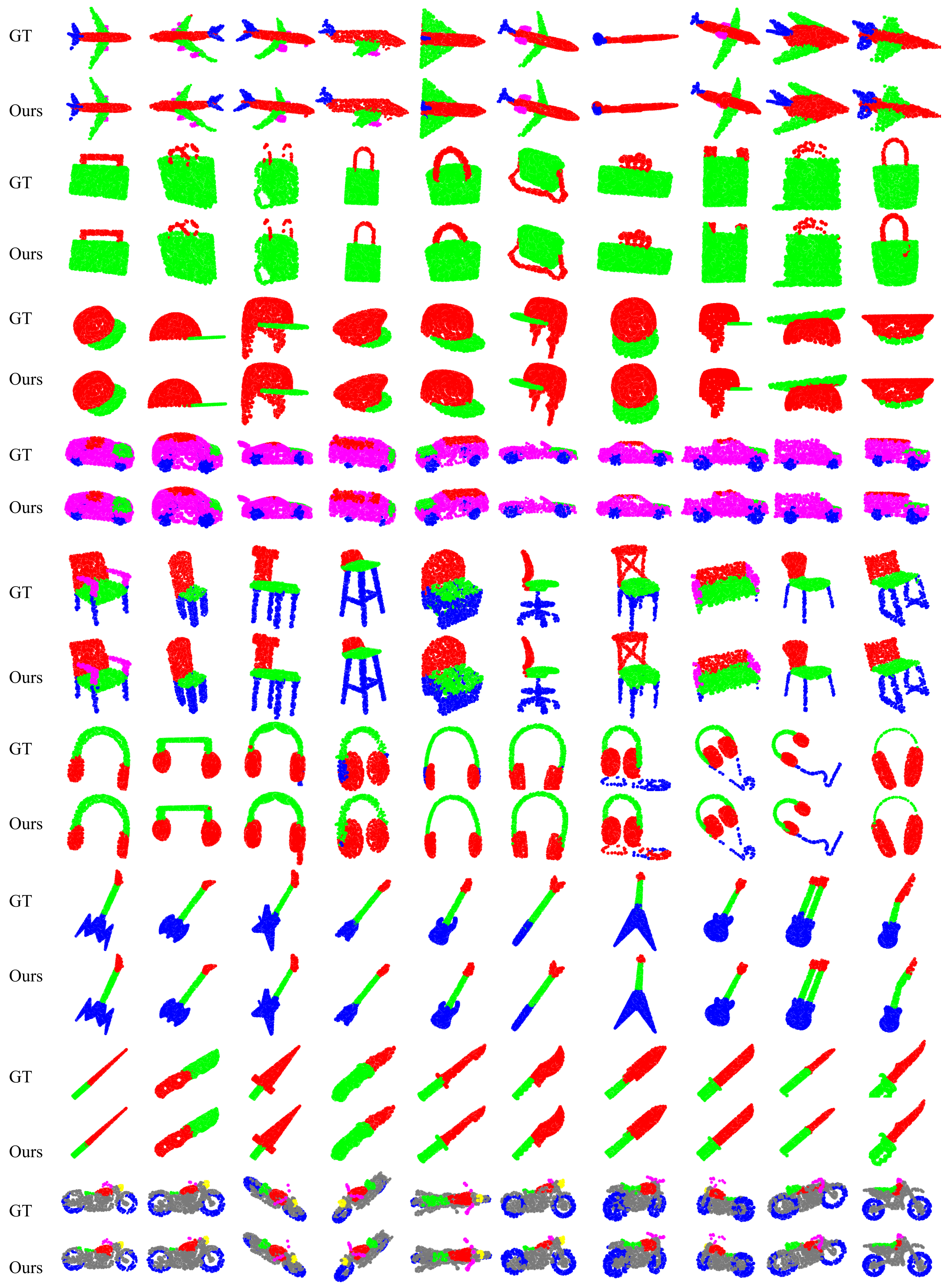


Fig. 14. More part segmentation results on ShapeNet dataset. The results are obtained by the modified RSCNN with our PS module.

results indicate that increasing the number of surface functions can improve the ability of shape representation. As the number of functions increases to a certain extent, the improvement gradually flattens out. Therefore, a better performance

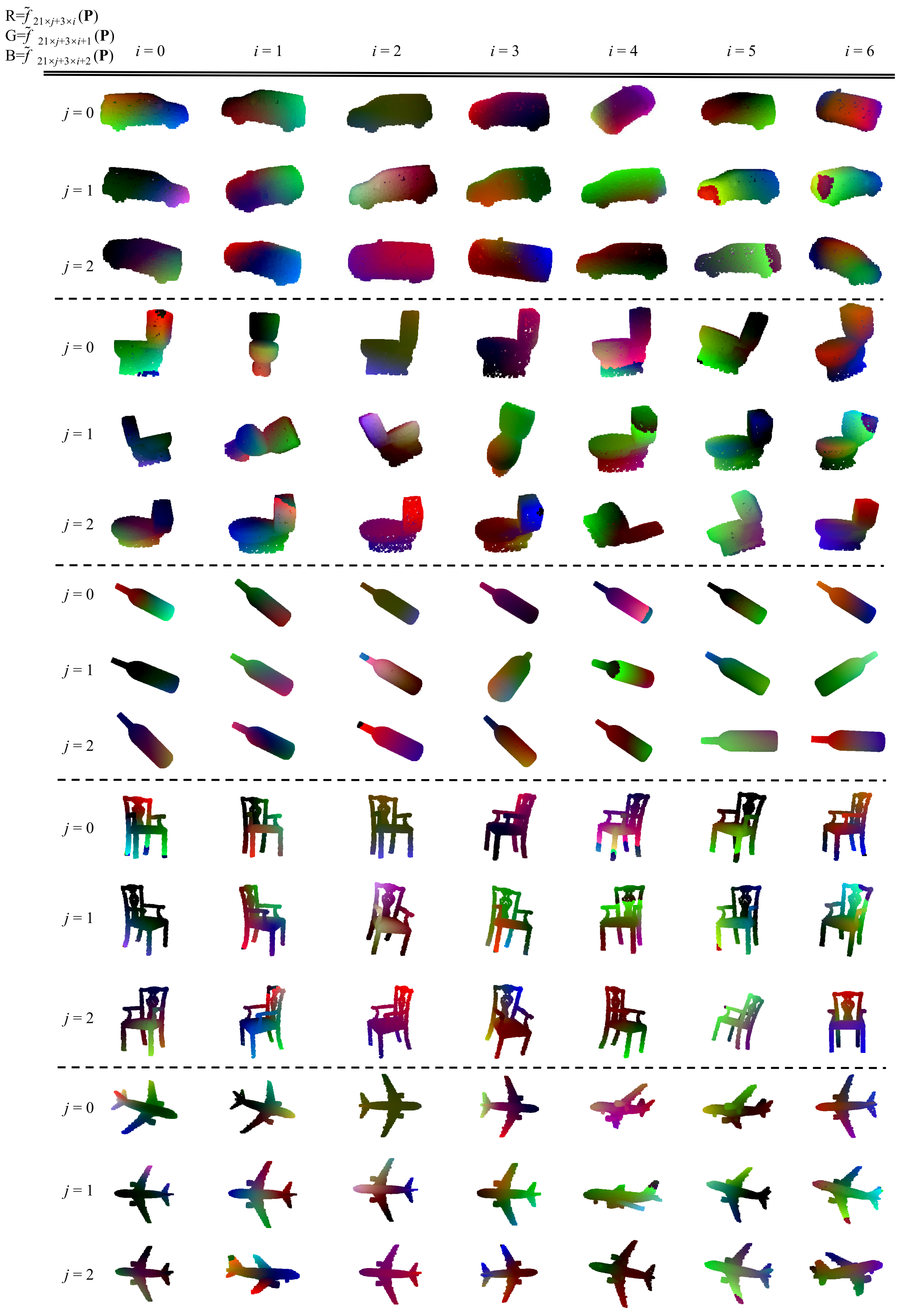

R=$\tilde{f}_{21\times j+3\times i}(\mathbf{P})$
G=$\tilde{f}_{21\times j+3\times i+1}(\mathbf{P})$
B=$\tilde{f}_{21\times j+3\times i+2}(\mathbf{P})$
$i=0$
$i=1$
$i=2$
$i=3$
$i=4$
$i=5$
$i=6$
$j=0$
$j=1$
$j=2$

Fig. 15. The visualization of point-to-surface representation. The three channels R, G and B of the point cloud color respectively represent the normalized functional values of the point cloud to three reference surfaces. For each shape, we have visualized $3 \times 3 \times 7 = 63$ point-to-surface representations.

does not require a large number of parameters and computation.

Finally, we test the robustness of the point-to-surface representation to the number of the input points. We trained our model with 64, 128, 256, 512, 1,024 and 2,048 points, respectively. The test classification accuracy is shown in Fig. 11. The result shows that our method is robust for different number of input points. Even with 64 input points, the model still keeps a higher classification ability. This is of great significance for the lightweight and real-time applications.

D. Visualization

1) Learned Reference Surfaces: In the training step, we learn the underlying surfaces through unsupervised learning. In order to visualize these underlying surfaces, we first utilize the learned surface function coefficients to get the shape point cloud by giving the x and y coordinates and solving the z coordinate. The mesh surface is reconstructed by transforming the point cloud to meshes with MeshLab [63] software. Some reconstructed surfaces are shown in Fig. 12. These surfaces are not obtained by fitting certain shapes, but are learned during the task to represent all 3D shapes. Since there are quadratic terms in the surface function, these surfaces are symmetric. Moreover, they differ from each other in structure, which ensure a strong representation capability.

2) Point-to-Surface Representation: Here, we want to visualize the contribution of a certain surface function to the point cloud. To this end, we set the R, G, B values of the point cloud to three surface function values. Then, we can visualize the contribution of 3 surface functions at a time. For example, in order to visualize the representation of function $f_1(x, y, z)$, $f_2(x, y, z)$, $f_3(x, y, z)$, the color of the point cloud $\mathbf{P}$ are defined as:

$$\begin{aligned} R &= \tilde{f}_1(\mathbf{P}), \\ G &= \tilde{f}_2(\mathbf{P}), \\ B &= \tilde{f}_3(\mathbf{P}), \end{aligned} \tag{5}$$

where the notation $\tilde{f}_i$ denotes the normalized value. Then, all representation values are visualized by a group of colorized point cloud. Fig. 13 gives two visualization examples. As shown in Fig. 13, the circled B region is overly flat in colour, indicating that the output values of these surface functions are continuous at these regions. We also notice some discontinuous regions, such as the green part on the chair leg in region A. One possible reason is due to the ReLU activation during the learning. Another reason is that the training makes different surface functions contribute differently to different parts of a 3D shape in a task. More point-to-surface representations are shown in Appendix B. This visualization is non-trivial for understanding and analyzing the point-to-surface representation.

V. Conclusion

In this paper, a quadratic terms based point-to-surface representation has been proposed to transform the point coordinates to the relationship between the local point and the global surfaces. The static representation can also become dynamic by adaptive contribution weighting. The proposed PS module can be inserted to the existing 3D learning networks. The experimental results show that the learning based on quadratic terms is more descriptive than the first order terms. The proposed point-to-surface representation is easy to implement and lightweight. Meanwhile, the modified networks with our representation outperform the state-of-the-art methods on 3D classification and segmentation. The experimental results also indicate that the representation is robust to sparse point cloud classification.

In the future, we will focus on the rotation invariance of the point-to-surface representation. In addition, the problem of over-fitting in 3D vision tasks is another problem worthy of further study.

Appendix A
Part Segmentation Results

More part segmentation results are displayed in Fig. 14. We compare the results of our method with the Ground Truth (GT). Please zoom-in for clearer visualization.

Appendix B
The Visualization of the Point to Surface Representation

More surface function representations are shown in Fig. 15. As we can only visualize 3 functions at a time by setting RGB values. Hence, we visualize the contributions of 63 surface functions by 21 point clouds for one shape. In Fig. 15, these 21 point clouds are divided into 3 rows, with 7 columns in each row. The row number is indexed by i, the column number is indexed by j, the color of different point cloud represents different surface function as follows:

$$\begin{aligned} R &= \tilde{f}_{21\times j+3\times i}(\mathbf{P}), \\ G &= \tilde{f}_{21\times j+3\times i+1}(\mathbf{P}), \\ B &= \tilde{f}_{21\times j+3\times i+2}(\mathbf{P}). \end{aligned} \tag{6}$$

References

[1] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.

[2] J. Huang, W. Yan, G. Li, T. Li, and S. Liu, "Learning disentangled representation for multi-view 3D object recognition," *IEEE Trans. Circuits Syst. Video Technol.*, early access, Feb. 24, 2021, doi: 10.1109/TCSVT.2021.3062190.

[3] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8895–8904.

[4] D. Li, G. Shi, Y. Wu, Y. Yang, and M. Zhao, "Multi-scale neighborhood feature extraction and aggregation for point cloud segmentation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 6, pp. 2175–2191, Jun. 2021.

[5] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 165–174.

[6] Y. Xiong, M. Ren, R. Liao, K. Wong, and R. Urtasun, "Deformable filter convolution for point cloud reasoning," 2019, *arXiv:1907.13079*. [Online]. Available: http://arxiv.org/abs/1907.13079

[7] H. Ren, M. El-Khamy, and J. Lee, "Deep robust single image depth estimation neural network using scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 37–45.

[8] Z. Wu *et al.*, "3D shapenets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920.

[9] A. Sharma, O. Grau, and M. Fritz, “VConv-DAE: Deep volumetric shape learning without object labels,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Amsterdam, The Netherlands: Springer, 2016, pp. 236–250.

[10] T. Yu, J. Meng, and J. Yuan, “Multi-view harmonized bilinear network for 3D object recognition,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 186–194.

[11] Y. Feng, Z. Zhang, X. Zhao, R. Ji, and Y. Gao, “GVCNN: Group-view convolutional neural networks for 3D shape recognition,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 264–272.

[12] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep hierarchical feature learning on point sets in a metric space,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.

[13] L. Landrieu and M. Simonovsky, “Large-scale point cloud semantic segmentation with superpoint graphs,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4558–4567.

[14] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph CNN for learning on point clouds,” *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, Nov. 2019.

[15] J.-R. Chang and Y.-S. Chen, “Pyramid stereo matching network,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5410–5418.

[16] S. Chen, Z. Pu, X. Fan, and B. Zou, “Fixing defect of photometric loss for self-supervised monocular depth estimation,” *IEEE Trans. Circuits Syst. Video Technol.*, early access, Mar. 24, 2021, doi: 10.1109/TCSVT.2021.3068834.

[17] S. Izadi *et al.*, “KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera,” in *Proc. 24th Annu. ACM Symp. User Interface Softw. Technol.*, Oct. 2011, pp. 559–568.

[18] A. A. Soltani, H. Huang, J. Wu, T. D. Kulkarni, and J. B. Tenenbaum, “Synthesizing 3D shapes via modeling multi-view depth maps and silhouettes with deep generative networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1511–1519.

[19] A. Haque, B. Peng, Z. Luo, A. Alahi, S. Yeung, and L. Fei-Fei, “Towards viewpoint invariant 3D human pose estimation,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Amsterdam, The Netherlands: Springer, 2016, pp. 160–177.

[20] D. Goudie and A. Galata, “3D hand-object pose estimation from depth with convolutional neural networks,” in *Proc. 12th IEEE Int. Conf. Autom. Face Gesture Recognit. (FG)*, May 2017, pp. 406–413.

[21] Y. Chen, Z. Tu, L. Ge, D. Zhang, R. Chen, and J. Yuan, “SO-HandNet: Self-organizing network for 3D hand pose estimation with semi-supervised learning,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6961–6970.

[22] J. Li, B. M. Chen, and G. H. Lee, “SO-Net: Self-organizing network for point cloud analysis,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9397–9406.

[23] M. Yavartanoo, E. Y. Kim, and K. M. Lee, “SPNet: Deep 3D object classification and retrieval using stereographic projection,” in *Proc. Asian Conf. Comput. Vis. (ACCV)*. Perth, WA, Australia: Springer, 2018, pp. 691–706.

[24] Z. Han, Z. Liu, J. Han, C.-M. Vong, S. Bu, and C. L. P. Chen, “Mesh convolutional restricted Boltzmann machines for unsupervised learning of features with structure preservation on 3-D meshes,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2268–2281, Oct. 2017.

[25] T. Le, G. Bui, and Y. Duan, “A multi-view recurrent neural network for 3D mesh segmentation,” *Comput. Graph.*, vol. 66, pp. 103–112, Aug. 2017.

[26] Y. Feng, Y. Feng, H. You, X. Zhao, and Y. Gao, “MeshNet: Mesh neural network for 3D shape representation,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 8279–8286.

[27] B. Taha, M. Hayat, S. Berretti, D. Hatzinakos, and N. Werghi, “Learned 3D shape representations using fused geometrically augmented images: Application to facial expression and action unit detection,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 9, pp. 2900–2916, Sep. 2020.

[28] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, “Multi-view convolutional neural networks for 3D shape recognition,” in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 945–953.

[29] A.-A. Liu, W.-Z. Nie, and Y.-T. Su, “3D object retrieval based on multi-view latent variable model,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 3, pp. 868–880, Mar. 2019.

[30] Y. Su, Y. Li, W. Nie, D. Song, and A.-A. Liu, “Joint heterogeneous feature learning and distribution alignment for 2D image-based 3D object retrieval,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 10, pp. 3765–3776, Oct. 2020.

[31] M. Gadelha, S. Maji, and R. Wang, “3D shape induction from 2D views of multiple objects,” in *Proc. Int. Conf. 3D Vis. (3DV)*, Oct. 2017, pp. 402–411.

[32] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, “Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 82–90.

[33] Y. Xu, Y. Wu, and H. Zhou, “Multi-scale voxel hashing and efficient 3D representation for mobile augmented reality,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 1505–1512.

[34] Y. Zhou and O. Tuzel, “VoxelNet: End-to-end learning for point cloud based 3D object detection,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4490–4499.

[35] T. Le and Y. Duan, “PointGrid: A deep network for 3D shape understanding,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9204–9214.

[36] A. Dai, C. R. Qi, and M. NieBner, “Shape completion using 3D-encoder-predictor CNNs and shape synthesis,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5868–5877.

[37] D. Stutz and A. Geiger, “Learning 3D shape completion from laser scan data with weak supervision,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1955–1964.

[38] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, “Learning representations and generative models for 3D point clouds,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 40–49.

[39] S. Wang, S. Suo, W.-C. Ma, A. Pokrovsky, and R. Urtasun, “Deep parametric continuous convolutional neural networks,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2589–2597.

[40] Y. Wang and J. Solomon, “Deep closest point: Learning representations for point cloud registration,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3523–3532.

[41] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[42] L. Yi *et al.*, “A scalable active framework for region annotation in 3D shape collections,” *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–12, Nov. 2016.

[43] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Niessner, “ScanNet: Richly-annotated 3D reconstructions of indoor scenes,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5828–5839.

[44] Y. Lin *et al.*, “FPConv: Learning local flattening for point convolution,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4293–4302.

[45] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, “Pointwise convolutional neural networks,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 984–993.

[46] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, “Deep sets,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3391–3401.

[47] M. Simonovsky and N. Komodakis, “Dynamic edge-conditioned filters in convolutional neural networks on graphs,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3693–3702.

[48] S. Xie, S. Liu, Z. Chen, and Z. Tu, “Attentional ShapeContextNet for point cloud recognition,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4606–4615.

[49] F. Groh, P. Wieschollek, and H. P. A. Lensch, “Flex-convolution (Million-scale point-cloud learning beyond grid-Worlds),” 2018, *arXiv:1803.07289*. [Online]. Available: http://arxiv.org/abs/1803.07289

[50] R. Klokov and V. Lempitsky, “Escape from cells: Deep kd-networks for the recognition of 3D point cloud models,” in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 863–872.

[51] Y. Shen, C. Feng, Y. Yang, and D. Tian, “Mining point cloud local structures by kernel correlation and graph pooling,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4548–4557.

[52] M. Gadelha, R. Wang, and S. Maji, “Multiresolution tree networks for 3D point cloud processing,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 103–118.

[53] C. Wang, B. Samari, and K. Siddiqi, “Local spectral graph convolution for point set feature learning,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 52–66.

[54] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, “PointCNN: Convolution on X-transformed points,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 820–830.

[55] M. Atzmon, H. Maron, and Y. Lipman, “Point convolutional neural networks by extension operators,” *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–12, Aug. 2018.

[56] R. Li, X. Li, P.-A. Heng, and C.-W. Fu, "PointAugment: An auto-augmentation framework for point cloud classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 6378–6387.

[57] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6410–6419.

[58] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui, "PointASNL: Robust point clouds processing using nonlocal neural networks with adaptive sampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 5589–5598.

[59] Y. Liu, B. Fan, G. Meng, J. Lu, S. Xiang, and C. Pan, "Dense-Point: Learning densely contextual representation for efficient point cloud processing," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 5239–5248.

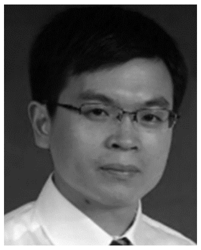
[60] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "SpiderCNN: Deep learning on point sets with parameterized convolutional filters," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 87–102.

[61] Q. Huang, W. Wang, and U. Neumann, "Recurrent slice networks for 3D segmentation of point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2626–2635.

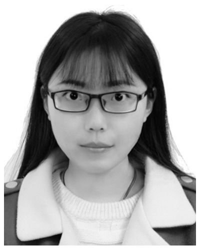
[62] H. Su *et al.*, "SPLATNet: Sparse lattice networks for point cloud processing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2530–2539.

[63] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "Meshlab: An open-source mesh processing tool," in *Proc. Eurographics Italian Chapter Conf.*, Salerno, Italy, 2008, pp. 129–136.

Tiecheng Sun received the B.E. degree in electronic information engineering from the University of Electronic and Technology of China in 2014, where he is currently pursuing the Ph.D. degree with the School of Information and Communication Engineering. His research interests include 3D descriptor, 3D reconstruction, and scene understanding. He served as a Reviewer for two international conferences, including International Conference on Intelligent Robots and Systems (IROS) and Pacific Graphics (PG).



Guanghui Liu (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in electronic engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2002 and 2005, respectively. In 2005, he joined Samsung Electronics, Seoul, South Korea, as a Senior Engineer. In 2009, he became an Associate Professor with the School of Electronics Engineering, UESTC, where he has been a Full Professor since 2014. He is with the School of Information and Communication Engineering, UESTC. His general research interests include multimedia, remote sensing, and wireless communication. In these areas, he has authored over ten articles in refereed journals or papers in conferences, and holds more than 60 patents (six U.S. granted patents). In 2015, he received the Natural Science Award and the Science and Technology Progress Award from the Ministry of Education of China.



Ru Li (Student Member, IEEE) received the B.E. degree in electronic information engineering from the China University of Petroleum, Qingdao, China, in 2016. She is currently pursuing the Ph.D. degree with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China. She visited University of Oxford from July 2019 to October 2019 and worked on image enhancement. Her research interests include image processing and computer vision.

Shuaicheng Liu (Member, IEEE) received the B.E. degree from Sichuan University, Chengdu, China, in 2008, and the M.S. and Ph.D. degrees from the National University of Singapore, Singapore, in 2010 and 2014, respectively. In 2014, he joined the University of Electronic Science and Technology of China, Chengdu, where he is currently an Associate Professor with the School of Information and Communication Engineering, Institute of Image Processing. His research interests include computer vision and computer graphics.

Shuyuan Zhu (Member, IEEE) received the Ph.D. degree from The Hong Kong University of Science and Technology (HKUST), Hong Kong, in 2010. From 2010 to 2012, he worked at HKUST and Hong Kong Applied Science and Technology Research Institute Company Ltd., respectively. In 2013, he joined the University of Electronic Science and Technology of China, where he is currently a Professor with the School of Information and Communication Engineering. His research interests include image/video compression and image processing. He is a member of IEEE CAS Society. He received the Top 10% Paper Award at IEEE ICIP-2014 and the Top 10% Paper Award at VCIP-2016. He served as the Committee Member for IEEE ICME-2014, VCIP-2016, and PCM-2017. He was the Special Session Chair of Image Super-Resolution at IEEE DSP- 2015.

Bing Zeng (Fellow, IEEE) received the B.E. and M.Sc. degrees in electronic engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 1983 and 1986, respectively, and the Ph.D. degree in electrical engineering from the Tampere University of Technology, Tampere, Finland, in 1991. He worked as a Post-Doctoral Fellow with the University of Toronto from September 1991 to July 1992, and as a Researcher with Concordia University from August 1992 to January 1993. Then, he joined The Hong Kong University of Science and Technology (HKUST). After 20 years of service, he returned to UESTC in Summer 2013, through Chinas 1000-Talent-Scheme. At UESTC, he leads the Institute of Image Processing to work on image and video processing, 3D and multi-view video technology, and visual big data. During his tenure with HKUST and UESTC, he has supervised more than 30 master's and Ph.D. students, received over 20 research grants, filed eight international patents, and published more than 250 articles. He was a recipient of the 2nd Class Natural Science Award (the first recipient) from the Ministry of Education of China in 2014 for his contributions to image and video coding. He received the Best Associate Editor Award in 2011. He was the General Co-Chair of the IEEE VCIP-2016, Chengdu, in November 2016. He is currently on the Editorial Board of *Journal of Visual Communication and Image Representation* and serves as the General Co-Chair for PCM-2017. He served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (TCSVT) for eight years.