GeoHi-GNN: Geometry-aware Hierarchical Graph Representation Learning for Normal Estimation

Submission ID (1020);

Abstract

Normal estimation has been one of the key tasks in point cloud analysis, while it is challenging when facing with severe noises or complex regions. The challenges mainly come from the selection of supporting points for estimation, that is, improper selections of points and points' scale will lead to insufficient information, loss of details, etc. To this end, this paper proposes one feature-centric fitting scheme, GeoHi-GNN, by learning geometry-aware hierarchical graph representation for fitting weights estimation. The main functional module is the continuously conducted Hierarchically Geometric-aware (HG) module, consisting of two core operations, namely, the graph node construction (GNC) and the geometric-aware dynamic graph convolution (GDGC). GNC aims to aggregate the feature information onto a smaller number of nodes, providing global-to-local information while avoiding the interferences from noises in larger scales. With these nodes distributed in different scales, GDGC dynamically updates the node features regarding to both intrinsic feature and extrinsic geometric information. Finally, the hierarchical graphical features are cascaded to estimate the weights for supporting points in the surface fitting. Through the extensive experiments and comprehensive comparisons with the state-of-the-arts, our scheme has exhibited many attractive advantages such as being geometry-aware and robust, empowering further applications like more accurate surface reconstruction.

Keywords: Normal estimation, Dynamic graph convolution, Hierarchical graphs, Surface fitting, 3D deep learning

1. Introduction

With the rise of autonomous driving technology, 3D data acquisition technology has gradually developed. Since point cloud data can better represent models and complex environments, the analysis of 3D point clouds has been the focal points of humans in recent decades, and normal estimation with raw point cloud as input is one of the key tasks. Accurate normals can help empower surface reconstruction Guerrero et al. (2018); Kazhdan et al. (2006), classification Qi et al. (2017b); Li et al. (2020), segmentation Ben-Shabat et al. (2018); Hu et al. (2020), etc. However, due to the presence of noise, complex structures, density variations, etc., accurate and robust normal estimation on point cloud remains challenging.

With the rapid growth of deep convolutional neural networks, recent researches generally conduct normal estimation via specifically-designed networks Cao et al. (2021); Guerrero et al. (2018); Ben-Shabat et al. (2019); Zhou et al. (2020b); Hashimoto and Saito (2019); Pistilli et al. (2020); Zhou et al. (2020a); Ben Shabat (2020); Holland and Welsch (1977); Lenssen et al. (2020); Zhu et al. (2021). Existing deep learning-based methods could be roughly divided into two groups, one of which is regression-based Guerrero et al. (2018); Ben-Shabat et al. (2019); Zhou et al. (2020b); Hashimoto and Saito (2019); Pistilli et al. (2020); Zhou et al. (2020a) and the other is surface fitting-based Ben Shabat (2020); Holland and Welsch (1977); Lenssen et al. (2020); Zhu et al. (2021). The regression-based methods force the network to learn the normal vectors with the knowledge of a few neighboring points. However, due to the disorder of 3D point clouds, the lack of topological structure, and the existence of noise points and outliers, these methods generally lack the generalization ability. In comparison, the fitting-based methods Ben Shabat (2020); Lenssen et al. (2020); Zhu et al. (2021) can achieve more accurate results. While, as for the fitting process, challenges mainly come

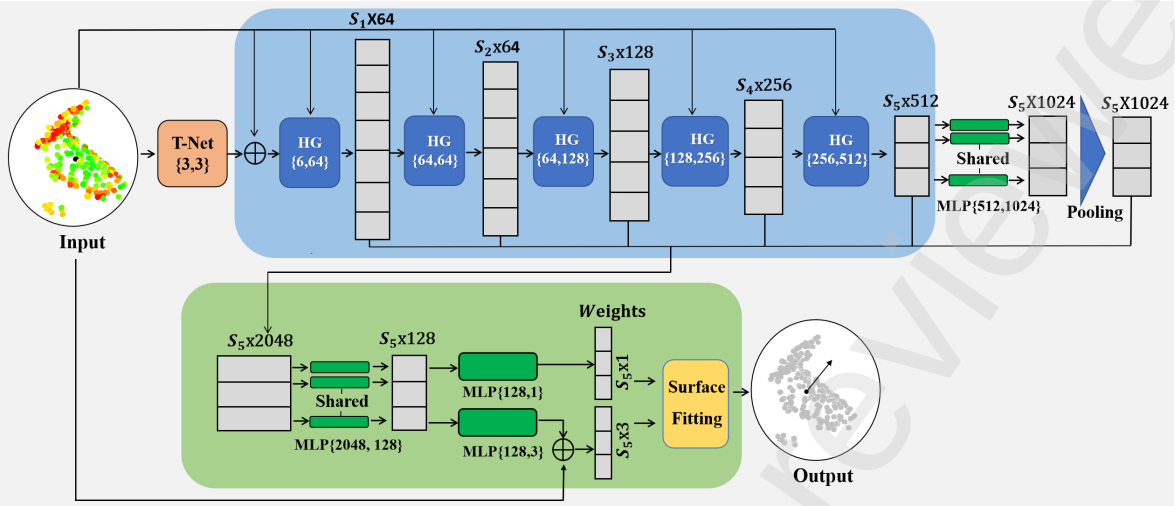


Figure 1: The functional pipeline of GeoHi-GNN, which mainly consists of five HG Blocks, cascading the hierarchical and geometric-aware graph features for weights estimation.

from the selection of supporting points. Too large or small scale of neighborhood will lead to over-smooth or unstable normal estimation results.

In recent years, a series of works Zhang et al. (2019); Valsesia et al. (2018); Wang et al. (2018); Simonovsky and Komodakis (2017); Li et al. (2019); Landrieu and Boussaha (2019) are proposed to improve the performance by utilizing the information contained in the supporting points. In these works, graph convolution is exploited to update neighborhood features, and build a local graph structure for each neighborhood so that the topological information is involved. However, noises and outliers are also involved in the constructed graph, which will affect the features. Besides, existed works simply concatenate local geometric features with global features, somewhat losing the multi-scale information. To remove the effects of noisy points and outliers, several fitting methods Ben Shabat (2020); Zhu et al. (2021) are proposed to predict the point-wise weights to control the contribution of supporting points. These works focus on setting more accurate weights for the supporting points either by selecting the offsets or proposing various feature representations. However, the estimated normals are still not accurate in the complex regions due to insufficient information or improper selection of supporting points.

To this end, this paper proposes one feature-centric fitting scheme based on hierarchically geometric-aware dynamic graph convolutions to abstract features of the point patch and conduct a soft selection of supporting points for robust normal estimation as shown in Fig. 1. In this new architecture, a HG module is introduced, consisting of two main operations, namely, GNC and GDGC. GNC aims to aggregate the current scale of feature information onto a smaller number of nodes. Then GDGC is incorporated to update the nodes' features considering both intrinsic feature and extrinsic spatial information. With HG modles implemented continuously, the whole network cascades the hierarchical graph features onto the final small number of graph nodes. By adopting the final graph nodes for least squares surface fitting, our scheme could achieve accurate and stable normal estimation results. The primary contributions of this paper can be summarized as follows:

- We propose a hierarchically geometric-aware fitting scheme, GeoHi-GNN, for point cloud normal estimation, which is stable and accurate with soft supporting points selection.
- We propose the HG module with two core operations, GNC and GDGC, for aggregating features when shrinking graph scale and dynamically update features considering both intrinsic feature and extrinsic

spatial information.

- We introduce one patch-perceptual loss to infer the patch-wise noise level and re-weight the supporting points for more accurate normal estimation.
- We demonstrate the superiority of our scheme over the state-of-the-arts by conducting qualitative and quantitative experiments on challenging data sets with different defects (noises, strips, gradients, etc), and also by facilitating application like surface reconstruction.

2. Related work

This section will briefly review prior researches related to normal estimation and the latest progresses on graph convolution.

Traditional Normal Estimation Methods. Normal is an important geometric attribute of 3D point cloud. In the traditional normal estimation methods, to estimate the normal vector of a point, it is necessary to estimate a plane through a local patch, and then calculate the normal vector of the point through the plane. In Hoppe et al. (1992), PCA-based point cloud normal estimation method is derived by the least squares method. Its core idea is to solve the eigenvalue and eigenvector through the covariance matrix for a local neighborhood, and then take the eigenvector corresponding to the smallest eigenvalue as estimated normal. Then, Cazals et al. Cazals and Pouget (2005) performs normal estimation by fitting a higher-level surface. However, in order to reduce the influence of noises and outliers, a larger neighborhood is usually used to calculate the normal, while the method performs poorly in models with large surface variations. Similarly, the method demonstrated in Klasing et al. (2009) is also too sensitive to noises and outliers. When the scale is improperly selected or the noise level is too high, these methods often perform unsatisfactory. As a further improvement, some methods Alliez et al. (2007); Dey and Goswami (2006); Mérigot et al. (2010) relying on point cloud Voronoi cells are proposed to improve the normal estimation results on feature points.

Deep Learning based Methods. In view of the success of deep learning methods in the 2D field, researchers started to explore the tools of neural networks in normal estimation. One of the main trend is to convert 3D point cloud data into 2D data as done in Su et al. (2015), which converts the point cloud data into 2D perspectives, and then processes it through 2D neural network. Another trend is to represent the point cloud as a volume occupancy grid Maturana and Scherer (2015), and then use a convolutional neural network for supervised learning. This method is faster for preprocessing 3D data, and achieves better results in the task of feature analysis of point clouds for different types of model data. Since data type conversion oftentimes leads to data loss, researchers started to pay attention to normal estimation methods that directly take point clouds as input. Recent researches mainly focus on two aspects, one is based on regression, and the other is based on surface fitting.

Regression-based Methods. Boulch et al. Boulch and Marlet (2016) first propose to use a convolutional neural network to regress the normal with the characteristics of Hong conversion of point cloud data, while the conversion process leads to information loss and make the final result unsatisfactory. Then, the proposal of the PointNet series of work has greatly improved the accuracy. PointNet Qi et al. (2017a) proposes to use T-NET to perform two alignment operations on the point cloud, corresponding to the geometric space and the feature space, and then performs MaxPooling on the learned features. It obtains the global features of the model, but lacks the local information. PointNet++ Qi et al. (2017b) was proposed to solve this problem by introducing the multi-level feature extraction structure. PointNet++ Qi et al. (2017b) not only solves the problem of how to deal with uneven sampling, but also considers the distance measurement between points in spatial space, and uses local patch information to learn features through the hierarchical structure, making the network structure more robust. Inspired by this, PCPNet Guerrero et al. (2018) proposes to directly use the local point cloud blocks as input to extract local feature information, which is more suitable for estimating local geometric properties. This method introduces a multi-scale network structure, and regresses to normal through multi-scale feature fusion. Ben et al. Ben-Shabat et al. (2019) incorporate the architecture of Mix of Experts (MOE) introduced in Jacobs et al. (1991) to select an optimal patch scale to regress to normal information.

Surface Fitting-based Methods. Surface fitting-based methods focus on how to better fit surfaces for local supporting points and then calculate the final normal. Ben et al. Ben Shabat (2020) propose the DeepFit architecture to learn the point-wise weights for weighted least squares polynomial surface fitting. This work uses the learned weights as the soft selection of adjacent points, avoiding the scale selection operation. Zhu et al. Zhu et al. (2021) focus on how to better fit local surfaces and propose an improvement of DeepFit, which returns the feature information learned by the neural network to the offset of each point in the local neighborhood, and then corrects the position of the point to better fit the surface.

Graph Convolution based Methods. Due to the recent success of graph convolution in 2D applications, more and more works have applied graph convolution to feature analysis and normal estimation on point cloud data. In order to better extract the local feature information, DGCNN Wang et al. (2019) is introduced by proposing the EdgeConv module to extract local information and build a local graph for each patch. A convolution-like operation is applied on edges connecting adjacent points to explore local geometric information. The network can not only better extract local neighborhood feature information, but also capture long-distance semantically similar information with the deepening of the network, achieving excellent results in classification and segmentation tasks. However, since the local geometric information is not considered, the results obtained by the network are unsatisfactory for some geometric feature analysis tasks. Therefore, researchers try to consider geometrically related information in the feature extraction process. Cui et al. Cui et al. (2021) improve the EdgeConv module by calculating the weight of different neighboring points by taking geometric relationship into consideration. And the weights are used to perform the final weighted feature update process. AdaptConv proposed in Cortes et al. (2017) is different from conventional methods that simply apply weights to feature information. It does not rely on predefined weights, but adaptively establishes the relationship between point pairs according to the learned features, so as to assign a learnable weight to each neighboring point.

3. Overview and Preparation

3.1. Overview

As for the task of normal estimation, the challenges mainly come from the disordered and unstructured point representation, especially when there exists serve noises and complex regions. We aim to preserve the low-frequency geometric structure meanwhile recover the high-frequency details robustly. For fitting-based methods, the core is to calculate the fitting weights for supporting points. Undoubtedly, the points that hold the intrinsic feature of the potential surface should contribute more in the fitting process with larger weights. Therefore, feature analysis and abstraction on the potential supporting points is of great importance.

Inspired by this, we propose to abstract the hierarchical features of neighboring points from both intrinsic and extrinsic aspects. As for the point-based convolutional architectures, previous methods treat all points equally in the convolutions, ignoring the geometric relationship when layers go deeper. Thus, we explore the input patch of points by introducing hierarchical graphs to describe the patch from global-to-local scales. To achieve robustness to noises and outliers, the smallest scale of points with multi-scale features should be selected for surface fitting. To this end, we firstly aggregate the previous features into smaller number of nodes via GNC. Then, to explore the intrinsic feature information and the extrinsic spatial information, the GDGC is introduced to update the nodes' features to further preserve the geometric details.

With the above two core operations, we set up the HG module. By gradually incorporating HG modules in feature abstraction, the dynamic graphs constructed capturing features from local neighbors to the whole patch. That is to say, our scheme endows the final smallest number graph nodes with the receptive field of the entire patch. Then the features learned from different scales are aggregated in a cascaded way Liu et al. (2019). Finally, the weight is obtained through the fully connected layer.

3.2. Preparation

In order to simplify the description of the problem, some definitions are given as below. For any given point $\mathbf{p}_i$ from a given 3D point cloud $\mathbf{X} = \{\mathbf{p}_1, \mathbf{p}_2, \cdots \mathbf{p}_N\} \in R^{N\times 3}$, our aim is to estimate the normal $N_{\mathbf{p}_i}$ of point $\mathbf{p}_i$. We use K Nearest Neighbor (KNN) search to define a local patch $\mathbf{P}_i = \{\mathbf{p}_{ij} | \mathbf{p}_{ij} \in KNN(\mathbf{p}_i)\}$,

here $j = 1, 2, \cdots, s$ and s is the number of the neighbors (we call it the scale of nodes in this paper). The corresponding features are denoted as $\mathcal{F} = \{\mathbf{f}_{i1}, \mathbf{f}_{i2}, \cdots \mathbf{f}_{ij}\} \in R^{s \times D}$. A directed graph of this patch can be represented as $\mathcal{G}(\nu, \mathcal{E})$ where ν consists of the feature vectors from $\mathcal{F}$ and $\mathcal{E} \subseteq \nu \times \nu$ represents the edge set. Notice that here the self-loop is included. Then we use PCA Hoppe et al. (1992) to align the patch as the input of our network. And the T-Net Qi et al. (2017a) is introduced to achieve the invariance to permutation.

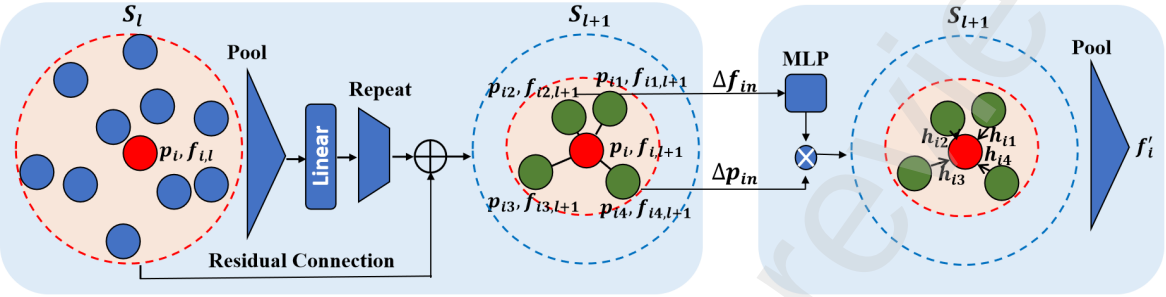


Figure 2: The HG module, consisting of GNC (left) and GDGC (right).

4. Method

The proposed architecture mainly includes the HG modules and the Jet fitting module as will be detailed in this section.

4.1. Hierarchically Geometric-aware Modules

The HG modules are the cores of our approach and each of them consists of the construction of graph nodes and the geometric-aware dynamic graph convolution, as shown in Fig. 2. Being cascaded in five times, the HG modules could hierarchically obtain the geometric-aware information in five scales on the concerned patch.

Graph Node Construction (GNC). Given the nodes and the corresponding features from the front layer (l-th layer), we propose to construct graph nodes for the current layer (($l+1$)-th layer) by aggregating the features from the l-th layer and shrinking the number of the graph nodes. As for the first HG module, the input nodes are set as the original points from the local patch. For the l-th layer, we denote the scale of it as s_l, and we selectively reduce the scale to s_{l+1} by discarding the farther away nodes from the central point, namely, $s_{l+1} < s_l$. The features of the k-th node on the ($l+1$)-th layer is updated via the node-pooling as follows:

$$\mathbf{f}_{k,l+1} = \theta_l(\gamma_l(\Pi\{\mathbf{f}_{k1,l}, \mathbf{f}_{k2,l}, \cdots, \mathbf{f}_{ks_l,l}\}), \mathbf{f}_{k,l}), \tag{1}$$

where, Π is the asymmetric function of max pooling. θ_l, γ_l are both Multi-layer Perceptions (MLP). $\mathbf{f}_{k,l}$ is the feature of scale s_l and $\mathbf{f}_{k1,l}, \cdots, f_{ks_l,l}$ are its neighbors' features. Note that from the scale s_l to s_{l+1}, although the farther points are discarded, their information is retained when updating the new nodes' features. Meanwhile the way to update the nodes and their features could reduce the effects of noises, that is, if noise points are selected as neighbors for the next graph layer, it will influence the final results. The scale updating process of several patches (with various geometric features) are visualized in Fig. 3.

Geometric-aware Dynamic Graph Convolution (GDGC). With the graph nodes constructed, we specifically propose one geometric-aware dynamic graph, which is equipped with the EdgeConv (as in DGCNN Wang et al. (2019)) as the basis to aggregate similar properties in the feature space. Given a graph node v_k, we define its n KNN neighbors as $v_{k1}, v_{k2}, \ldots, v_{kn}$ with features $f_{k1}, f_{k2}, \ldots, f_{kn}$. DGCNN constructs a local graph based on nodes and edges, and update the feature f'_k of node v_k by:

$$f'_k = \Pi(\eta(h(f_k, f_{kn}))), \tag{2}$$

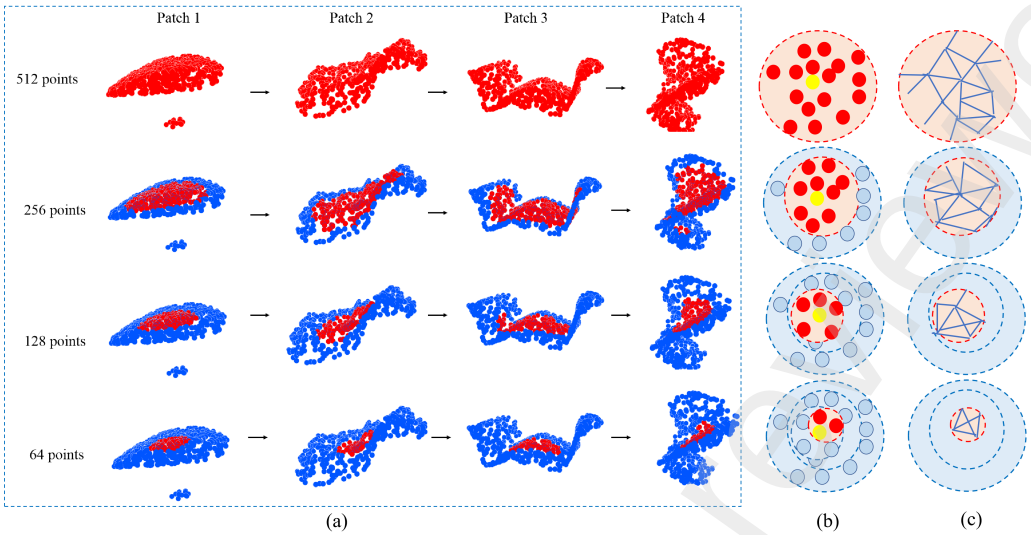


Figure 3: Visualization of hierarchical graphs in our GeoHi-GNN on several examples. (a) shows the change of patch scales with the node-pooling operation, where the red points represent the nodes selected to construct local graph by the current layer, and the blue points represent the discarded points. The number of nodes in our experiment changes as: 512-384-256-128-64 (here we just show four scales for visual clearness).(b) and (c) visualize the scale updating process of our HG models regarding to the receptive field of feature abstraction and graph convolution respectively.

where h calculates the relationship between v_k and v_{kn} in the feature space. η is MLP. Π is a symmetric function like max pooling. Note that the edges here are computed only by the features. Therefore it can only describe the intrinsic feature relation. But actually from the original point cloud data, extrinsic geometric information among vertices is also meaningful for the feature-learning Matveev et al. (2020). So we combine both of them to compute the edges and then update the features. The input feature of one dynamic graph could be represented as:

$$\Delta f_{kn} = \rho(f_k, f_{kn} - f_k), \tag{3}$$

where ρ is the concatenation operation. f_{kn} is the neighbors' features. We further define the geometric input as follows:

$$\Delta p_{kn} = \rho(p_k, p_{kn} - p_k), \tag{4}$$

where, p represents the xyz coordinates of the points. After that we can get h_{kn} which represents the edge feature of point p_k and p_n:

$$h_{kn} = \sigma(\vartheta(\lambda(\Delta f_{kn}), \Delta p_{kn})), \tag{5}$$

where λ is a feature mapping function, and we use MLP in our work. ϑ is the inner product of the vectors. σ is a nonlinear activation function, and we use LeakyReLU. Finally the output f'_k can be computed by:

$$f'_k = \Pi \{h_{k1}, h_{k2}, \cdots, h_{kn}\}. \tag{6}$$

With this settings, we consider the relationship not only in the feature space, but also in the spatial space. When the network goes deeper, the neighbors' features will be quite similar, causing different neighbors to play similar roles in the feature updating. However different neighbors should be treat differently regarding to the local geometry and typology, esp. for the normal estimation task. By conducting the graph convolution in this dynamic and geometric-aware way, the hierarchically located graphs (as shown in Fig. 3(c)) could capture the global-to-local features of the concerned patch, providing sufficient information for estimating the fitting weights.

Table 1: RMSE Angle Error For Different Methods

Aug.	Ours	DeepFit	Lenssen	Nesti-Net	PCPNet	PCA	Jet	AdaFit
No Noise	**4.49**	6.51	6.72	6.99	9.66	12.29	12.23	5.19
Low Noise	**8.77**	9.21	9.95	10.11	11.46	12.87	12.84	9.05
Med Noise	**16.44**	16.72	17.18	17.63	18.26	18.38	18.33	16.44
High Noise	**21.89**	23.12	21.96	22.28	22.8	27.5	27.68	21.94
Strips	**5.39**	7.92	7.73	8.47	11.74	13.66	13.39	6.01
Gradients	**5.03**	7.31	7.51	9.00	13.42	12.81	13.13	5.90
Average	**10.33**	11.8	11.84	12.41	14.56	16.25	16.29	10.76

4.2. Jet Fitting

To fit the surface for normal estimation, we adopt the truncated Taylor expansion surface fitting using weighted least-squares (WLS) Cazals and Pouget (2005) as well as the offset prediction strategy proposed in Adafit Zhu et al. (2021). To simplify the description of the problem, we call it m-jet below. The surface embedding process can be written as a height function as below:

$$Z = J_{\beta,m}(x,y) = \sum_{k=0}^{m}\sum_{j=0}^{k}\beta_{k-j,j}x^{k-j}y^{j}, \tag{7}$$

where β is the jet coefficients vector consisting of terms. Since we aim to perform local surface fitting on local patch, Vandermonde matrix $\boldsymbol{M} = \left(1, x_i, y_i, \ldots, x_i{y_i}^{m-1}, {y_i}^{m}\right)$ is specified and the height function can be denoted as $Z = \left(z_1, z_2, \ldots, z_{N_p}\right)$ (N_p is the number of supporting points). Then a system of linear equation can be obtained as:

$$Z = \boldsymbol{M}\beta. \tag{8}$$

Then we use an WLS approximation to minimize the sum of square errors between the value of the jet and the height function. Moreover, we define a diagonal weight matrix $\boldsymbol{W} = diag\left(w_1, w_2, \cdots, w_{N_p}\right)$ to control the impact of noise and outliers. Then we could get:

$$\beta = \arg\min_{z\in R^{N_p}} \left\| \boldsymbol{W}^{1/2}\left(\boldsymbol{M}_z - Z\right)\right\|^2. \tag{9}$$

After further simplification, the final solution could be represented as:

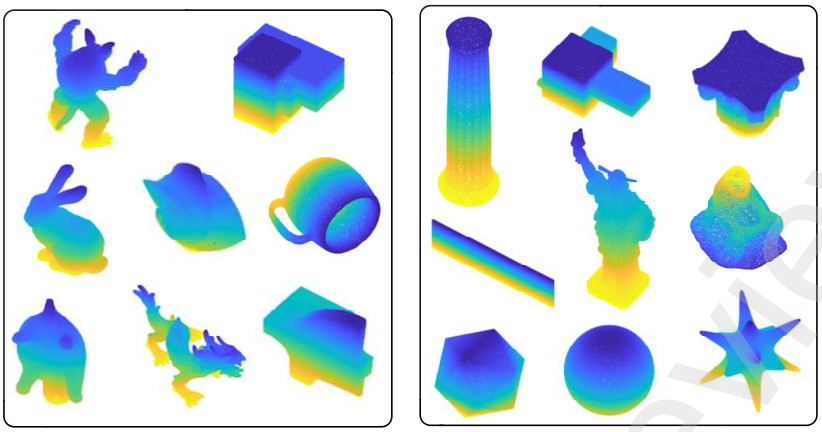
$$\beta = \left(\boldsymbol{M}^T\boldsymbol{W}\boldsymbol{M}\right)^{-1}\boldsymbol{M}^T\boldsymbol{W}Z. \tag{10}$$

When the surface is fitted, the normal can be computed by:

$$N_i = \frac{(-\beta_1, -\beta_2, 1)}{\|(-\beta_1, -\beta_2, 1)\|_2}, \tag{11}$$

whereβ_1 and β_2 are defined as $\beta_{1,0}$ and $\beta_{0,1}$ in Eq. 7, respectively. And for the neighbors $p_{ij} \in P_i$, the normal can be computed by:

$$N_j = \frac{\nabla F}{\|\nabla F\|}|_{p_{ij}} = \frac{\left(-\beta_r\frac{\partial \mathrm{M}^T}{\partial x}, \beta_r\frac{\partial \mathrm{M}^T}{\partial y}, 1\right)}{\|\nabla F\|}|_{p_{ij}}. \tag{12}$$



(a) training data (b) testing data

Figure 4: The main shapes in PCPNet Guerrero et al. (2018) dataset. (a) shows the training data, which mainly consists of 8 shapes. (b) is the testing data consisting of 19 shapes.

Table 2: Accuracy under Different Angle Tolerance (%)

	1°	3°	5°	10°	15°	20°	25°
PCPNet	3.5	20.15	30.23	**40.80**	**45.18**	**47.75**	**49.42**
Nesti-Net	10.32	22.89	29.64	36.82	39.71	41.25	42.35
DeepFit	11.55	21.73	26.79	32.62	35.13	36.48	37.37
Ours	**18.21**	**29.95**	**35.12**	40.36	42.53	43.80	44.61

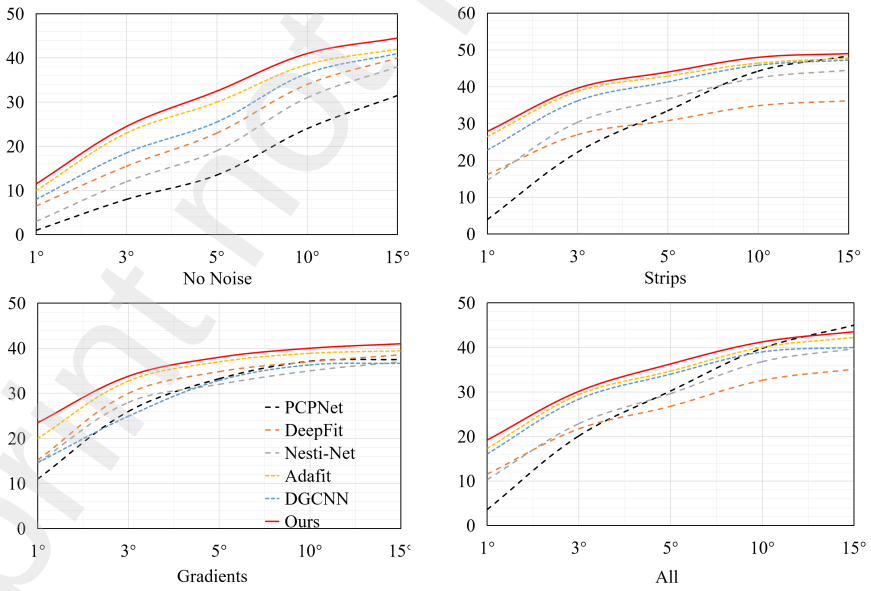


Figure 5: Accuracy of our method and other baseline methods, X-axis shows the threshold in degree and Y-axis shows the ratio of correct estimated under a given threshold.

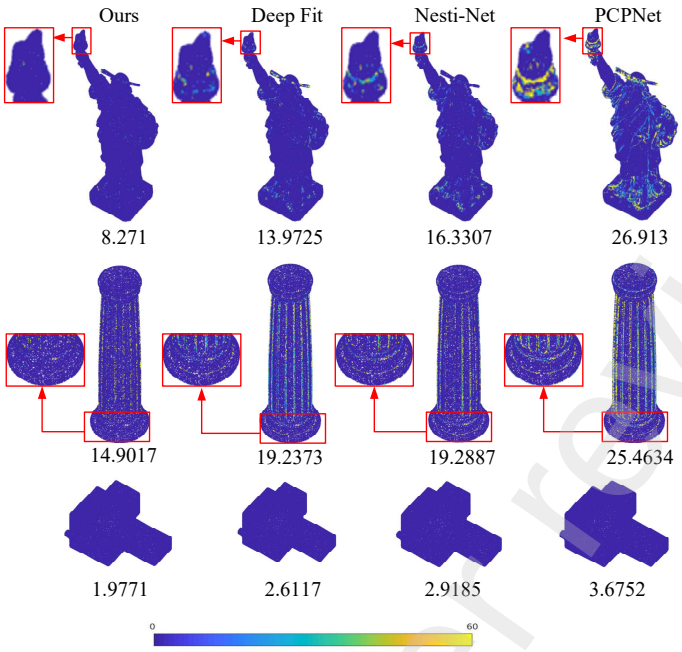
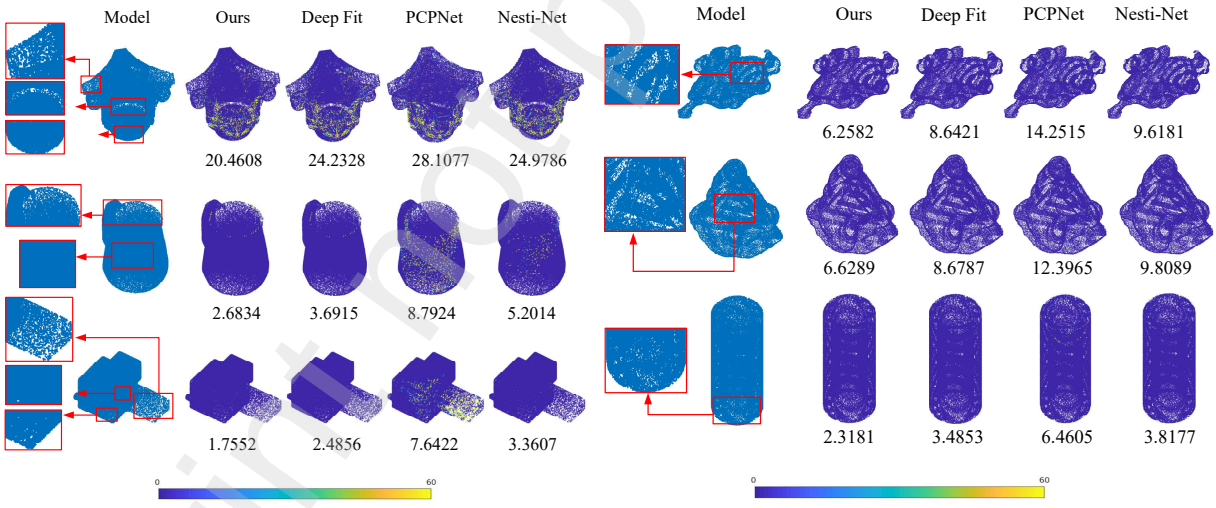


Figure 6: Visualization of normal angle error on models without noise. The colors of the points correspond to angular difference, mapped to a heatmap ranging from 0-60 degrees. And the number represents the RMSE error.



(a) Gradients Sampling

(b) Strips Sampling

Figure 7: RMSE error on models with different sampling schemes.

4.3. Loss Function

In order to estimate the fitting weights more accurately, we introduce a patch-perceptual loss L_{pp}, which consists of the weighted consistency term L_{cen} and the patch self-constraint term L_{pat}. Here, we denote N_{GT} as the ground truth normal of the central point p_i, and N_j as the predicted normal of the neighbor point and utilize both of them to constrain the weight of the points in the local patch, ensuring that neighbor

points with larger deviations will be given lower weights. To this end, the consistency loss is set as:

$$L_{con} = \frac{1}{N_{p_i}}(-\sum_{j=1}^{N_{p_i}} \log(w_j) + \sum_{j=1}^{N_{p_i}} w_j \left|N_j \cdot N_{GT}\right|). \tag{13}$$

Then we further use the difference between N_{GT}^j and N_j to infer the noise level and define L_{patch} as:

$$L_{patch} = \frac{1}{N_{p_i}} \sum_{j=1}^{N_{p_i}} w_j \left|N_j \cdot {N_{GT}}^j\right|. \tag{14}$$

Then L_{pp} can be represented as:

$$L_{pp} = L_{con} + L_{patch}. \tag{15}$$

And we also adopt the sin loss:

$$L_{\sin} = \left|N_i \cdot N_{GT}\right|, \tag{16}$$

where N_i is the predicted normal of the central point in the patch. Above all, the final loss L_{tot} can be represented as:

$$L_{tot} = L_{sin} + \mu_1 L_{pp}, \tag{17}$$

where μ_1 is a weighting factor, and we empirically set it as 0.25 in our work.

5. Experimental Results and Discussions

In this section, we demonstrate the performance of our approach by conducting experiments in various aspects. All the experiments are conducted on Nvidia RTX3090 GPU with 24G memory.

5.1. Dataset

In order to evaluate all the methods, we use the PCPNet Guerrero et al. (2018) dataset for network training and testing. We use 8 models to train our network and 19 models to test. The data is augmented by introducing i.i.d. Gaussian noise for each point's spacial location with a standard deviation of 0.012, 0.006, 0.00125 w.r.t the bounding box size. These models are visualized in Fig. 4. A subset of 5000 points per shape is used for evaluation. The batch size is set as 128 and the learning rate is set as 0.1. The network is trained with the Adam optimizer.

5.2. Performance Evaluation and Comparisons

Firstly, we introduce the RMSE error between the estimated and the ground-truth normal to evaluate the accuracy of the normal direction estimation, as shown in Table 1. It can be seen that under different noise levels and point cloud densities, our method has achieved the state-of-the-art results. In comparison with other methods, our method has better anti-noise and feature-preserving ability. Then the correct proportion of normals obtained by different methods are counted under different angle tolerance Hyeon et al. (2019) on the PCPNet dataset as shown in Table 2. Meanwhile, the accuracy of various methods at different noise levels is visualized by line graphs in Fig. 5. It's clear that compared with DeepFit Ben Shabat (2020) our method achieves better results in the absence of noises. Although Adafit Zhu et al. (2021) uses the offset to project points to the neighboring regions and achieve better result than other methods, our scheme performs better, esp. when there exist density variations, since it sufficiently explores the local-to-global features.

In Fig. 6, the RMSE error is introduced for further comparions. For clearer visualization, we utilize a heat map to highlight the angular difference. Note that other methods may need scale selection, that is, their results are not consistent at different scales. For fair comparison, we select their best results for comparison. As shown in Fig. 6, we get a better performance in sharp edges which benefits from our hierarchically geometry-aware graph convolution that take into account the spatial and multi-scale features. Fig. 7(a) and (b) show the results for models with gradients and strips sampling schemes. It's clear that our method performs better than others methods specially on the sharp features.

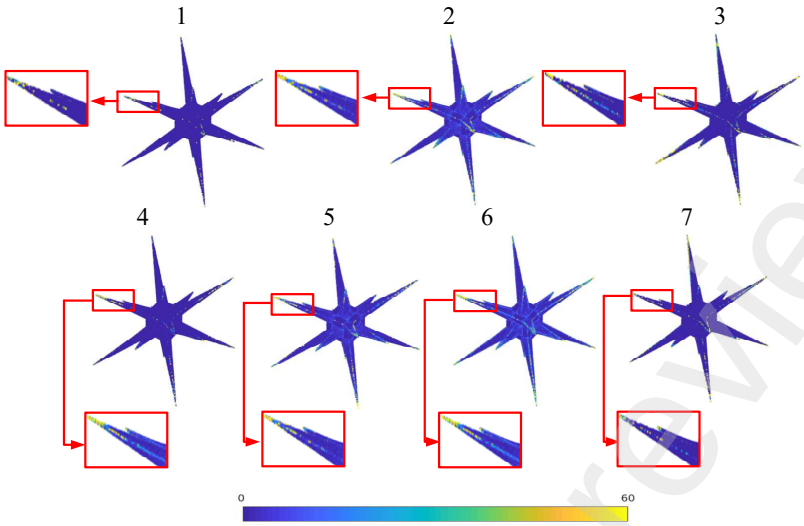


Figure 8: The visualization of RMSE value of different experimental combinations, the numbers correspond to the experimental serial numbers in Table 3 (the first row).

5.3. Ablation Study

This section will explicitly explore the separate parts of our scheme, including the HG module, including its two main components (GNC and GDGC), and our feature patch-perceptual loss. The main metric we utilize is the RMSE metric, as shown in Table 3. At the same time, in order to more clearly see the performance of different settings in Table 3, we select some models to visualize the RMSE results in Fig. 8.

Table 3: Ablation Test Results

Experiment	1	2	3	4	5	6	7
Our Conv	✓			✓	✓	✓	
EdgeConv		✓					✓
Conv1d			✓				
Node-pooling	✓	✓	✓		✓		
Our Loss	✓	✓	✓	✓			
No Noise	4.49	5.88	6.54	5.20	**4.48**	5.80	6.13
Low Noise	**8.77**	9.11	9.33	9.08	8.89	9.08	9.29
Med Noise	**16.44**	16.51	16.48	17.39	16.69	16.54	17.40
High Noise	**21.89**	21.91	22.03	25.43	22.95	22.78	25.48
Strips	5.39	7.08	8.47	6.25	**5.31**	6.83	7.25
Gradients	**5.03**	6.481	9.00	5.83	5.08	6.37	6.67
Average	**10.33**	11.16	12.41	11.53	10.58	11.23	12.03

Graph Node Construction. To demonstrate the effectiveness of the way we construct the graph nodes, we remove the GNC in HG module and get the results as shown in column 1 and column 4 in the Table 3. It's clear that GNC can help improve the normal estimation results by a large margin, esp. when the noise level is high. This demonstrates that GNC could help improve the anti-noise ability of our approach by discarding the far away points continuously.

Geometry-aware Dynamic Graph Convolution. Here we use DGCNN Wang et al. (2019) as a baseline to compare traditional graph convolution and our GDGC. We specifically replace our convolution

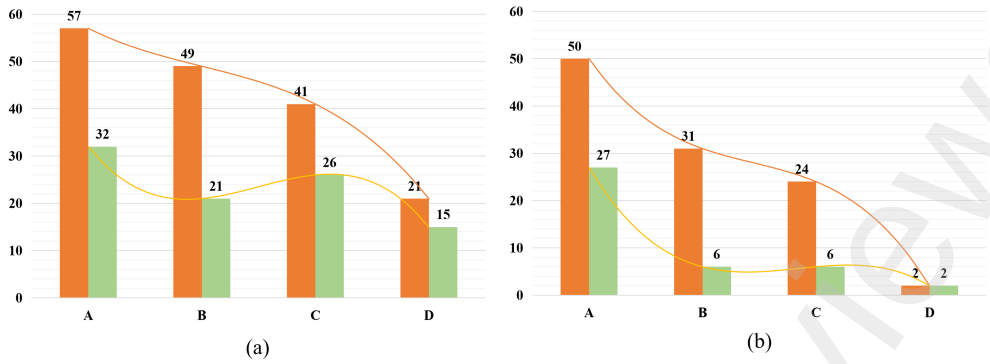


Figure 9: Visualization of improvement ratio. (a) shows different module combinations relative to the baseline, where orange and green represents the icoahedron and netsuke model respectively. (b) shows the average improvement rate of all complex models (orange) and simple models (green) in the PCPNet dataset.

Table 4: Effectiveness of Hierarchical Scales

Scale	256-256	512-64	256-32
No Noise	5.80	4.49	**4.30**
Low Noise	9.08	**8.77**	8.92
Med Noise	16.54	**16.44**	16.68
High Noise	22.87	**21.89**	22.95
Stripes	6.83	5.39	**5.27**
Gradients	6.37	**5.03**	5.09
Average	11.23	**10.33**	10.55

with the EdgeConv Wang et al. (2019) and get the results as shown in column 1 and column 2 in Table 3. In comparison, our convolution has seen better performance, esp. when inferring by gradient or strip sampling scheme. Furthermore, we also introduce the Conv1d in Deepfit Ben Shabat (2020) for comparison as shown in column 3 in Table 3. In this comparison, both methods use 512 neighboring points as input and 64 points to fit the final surface. The results demonstrate the effectiveness of our convolution in capturing geometric details.

Patch-perceptual Loss. In the first and fifth columns of Table 3, we show the results of two experiments with and without our patch-perceptual loss. It's clear that, in case of all levels of noise, the proposed loss could help the whole scheme increase the robustness thanks to its functionality in inferring the patch-based noise for weighting the supporting points.

HG Module. To evaluate the role of the HG module, we specifically design four schemes for comparisons. Before that, we set the scheme with Conv2d replacing HG in our scheme as the **Baseline**. Then, we denote **A**: **Baseline** + GNC+GDGC (our scheme), **B**: **Baseline** + GNC + EdgeConv, **C**: **Baseline** +GDGC, **D**: **Baseline** + GNC. And we show the improvement ratio of the above schemes compared with the baseline in Fig. 9. Fig. 9(a) shows the improvement ratios on the model netsuke and the model icoahedron, and (b) shows the average improvement ratio of all complex models and simple models.

Furthermore, we demonstrate the effectiveness of our hierarchically geometric-aware graph feature abstraction by fixing the scales for comparison. As shown in Table 4, when we set the scale 512 and 256 as initial input respectively, and the number of graph nodes is changing as 512-384-256-192-128-64 and 256-192-128-96-64-32. It's clear that compared with the fixed-scale experiment, multiple scale setting can obtain a better performance, indicating that hierarchical feature abstraction could explore the patch's intrinsic and

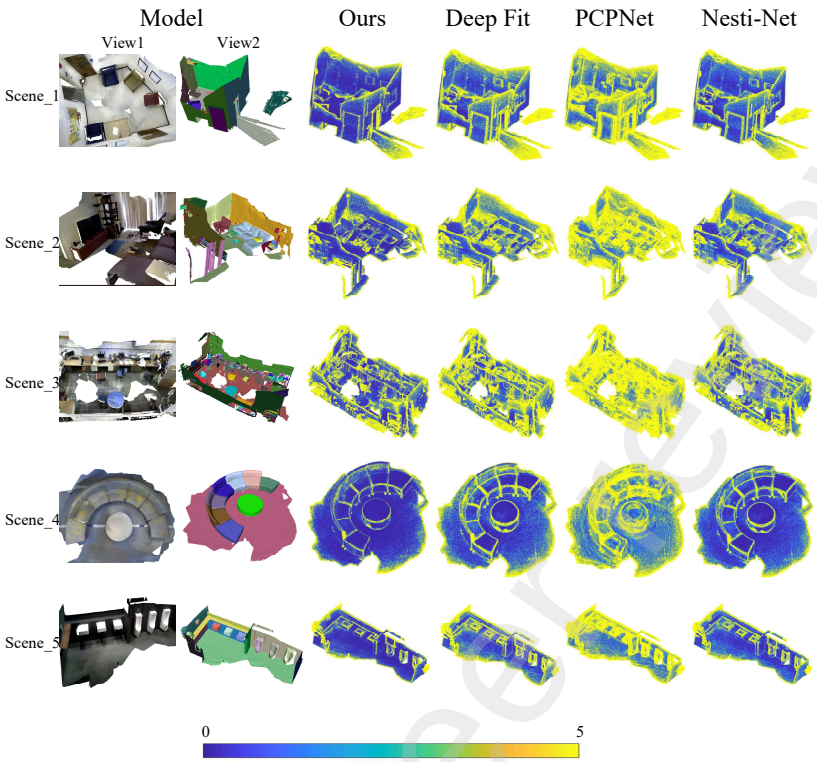


Figure 10: Error maps of estimated normals for different methods on the real SceneNN dataset. Colors correspond to angular difference and it is mapped to a heatmap ranging from 0°-5°.

extrinsic features more comprehensively.

5.4. Experiments on Real Dataset

In order to test the expansibility of our method to real scenes, we carry out the test on real indoor SceneNN Zapata-Impata et al. (2019) dataset without going through any training process. The SceneNN dataset contains a large number of indoor scenes, each consisting of tens of millions of points. We use the real reconstruction network provided by it to obtain point cloud data, and then use the model trained by our method to predict the corresponding normal. Fig. 10 shows the RMSE error for part of the scenes, where the colors of the points correspond to angular difference, mapped to a heatmap ranging from 0-5 degree. It can be seen that our method also achieves higher accuracy and better performance in real indoor scenes.

5.5. Application to Surface Reconstruction

Accurate normal estimation can undoubtedly facilitate the application of surface reconstruction. Thus we conduct the surface reconstruction experiments based on the normals estimated with the related methods and ours. Fig. 11 visualizes the reconstruction results for comparisons. It is not difficult to see that in general, our method has achieved better results, especially in the regions with sharp features and complex geometric variations or details.

6. Conclusion and Future Work

This paper explores a novel way of abstracting hierarchically geometric-aware features for the task of normal estimation on point cloud based on dynamic graph convolution and surface fitting scheme. Specifically,

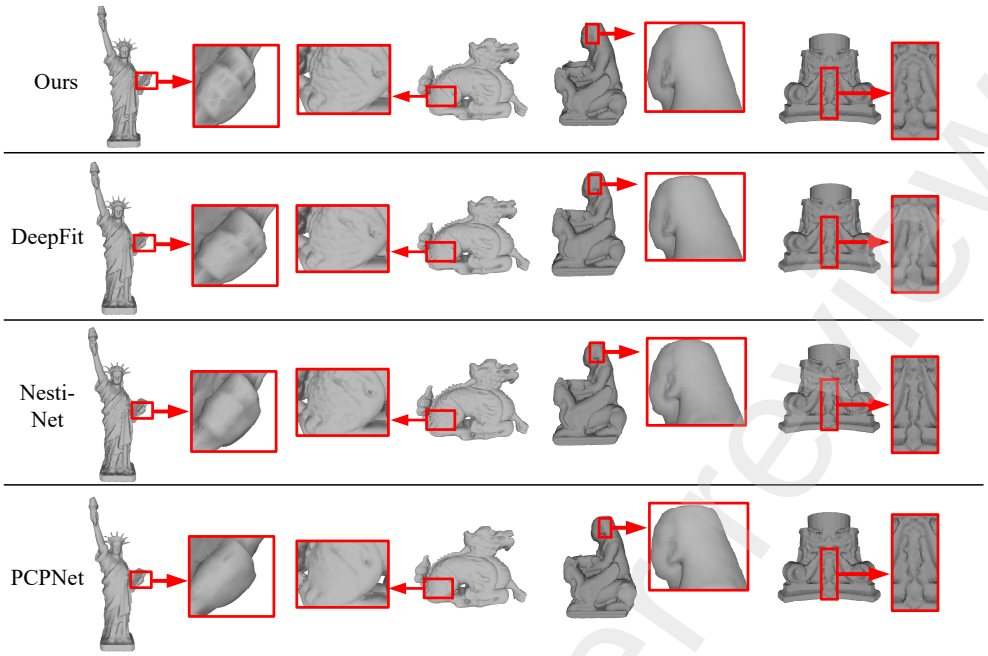


Figure 11: Surface reconstruction based on normals estimated by different methods. The red box represents the details after zooming in.

we introduce the HG module to construct graph nodes and conduct geometric-aware dynamic convolutions with decreasing number but with increasingly multi-scale information. Then by cascading the features obtained by HG modules, our scheme provides more accurate weights of the supporting nodes for the final surface fitting. The extensive experiments have demonstrated that compared with the existed methods, our method performs better on models with different noise levels and complex cases, such as uneven sampling with gradients and strips. Our future work will focus on dealing with severely noisy models, which are more common in the actual production and life due to the current instrument limitations.

References

Alliez, P., Cohen-Steiner, D., Tong, Y., Desbrun, M., 2007. Voronoi-based variational reconstruction of unoriented point sets, in: Symposium on Geometry processing, pp. 39–48.

Ben Shabat, Yizhak Gould, S., 2020. Deepfit: 3d surface fitting via neural network weighted least squares, in: European Conference on Computer Vision, Springer. pp. 20–34.

Ben-Shabat, Y., Avraham, T., Lindenbaum, M., Fischer, A., 2018. Graph based over-segmentation methods for 3d point clouds. Computer Vision and Image Understanding 174, 12–23.

Ben-Shabat, Y., Lindenbaum, M., Fischer, A., 2019. Nesti-net: Normal estimation for unstructured 3d point clouds using convolutional neural networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10112–10120.

Boulch, A., Marlet, R., 2016. Deep learning for robust normal estimation in unstructured point clouds, in: Computer Graphics Forum, Wiley Online Library. pp. 281–290.

Cao, J., Zhu, H., Bai, Y., Zhou, J., Pan, J., Su, Z., 2021. Latent tangent space representation for normal estimation. IEEE Transactions on Industrial Electronics 69, 921–929.

Cazals, F., Pouget, M., 2005. Estimating differential quantities using polynomial fitting of osculating jets. Computer Aided Geometric Design 22, 121–146.

Cortes, C., Gonzalvo, X., Kuznetsov, V., Mohri, M., Yang, S., 2017. Adanet: Adaptive structural learning of artificial neural networks, in: International Conference on Machine Learning, PMLR. pp. 874–883.

Cui, Y., Liu, X., Liu, H., Zhang, J., Zare, A., Fan, B., 2021. Geometric attentional dynamic graph convolutional neural networks for point cloud analysis. Neurocomputing 432, 300–310.

Dey, T.K., Goswami, S., 2006. Provable surface reconstruction from noisy samples. Computational Geometry 35, 124–141.

Guerrero, P., Kleiman, Y., Ovsjanikov, M., Mitra, N.J., 2018. Pcpnet learning local shape properties from raw point clouds, in: Computer Graphics Forum, Wiley Online Library. pp. 75–85.
Hashimoto, T., Saito, M., 2019. Normal estimation for accurate 3d mesh reconstruction with point cloud model incorporating spatial structure., in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.
Holland, P.W., Welsch, R.E., 1977. Robust regression using iteratively reweighted least-squares. Communications in Statistics-theory and Methods 6, 813–827.
Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W., 1992. Surface reconstruction from unorganized points, in: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, pp. 71–78.
Hu, Z., Zhang, D., Li, S., Qin, H., 2020. Attention-based relation and context modeling for point cloud semantic segmentation. Computers & Graphics 90, 126–134.
Hyeon, J., Lee, W., Kim, J.H., Doh, N., 2019. Normnet: Point-wise normal estimation network for three-dimensional point cloud data. International Journal of Advanced Robotic Systems 16, 1729881419857532.
Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E., 1991. Adaptive mixtures of local experts. Neural computation 3, 79–87.
Kazhdan, M., Bolitho, M., Hoppe, H., 2006. Poisson surface reconstruction, in: Proceedings of the Fourth Eurographics Symposium on Geometry Processing.
Klasing, K., Althoff, D., Wollherr, D., Buss, M., 2009. Comparison of surface normal estimation methods for range sensing applications, in: 2009 IEEE international Conference on Robotics and Automation, IEEE. pp. 3206–3211.
Landrieu, L., Boussaha, M., 2019. Point cloud oversegmentation with graph-structured deep metric learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7440–7449.
Lenssen, J.E., Osendorfer, C., Masci, J., 2020. Deep iterative surface normal estimation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11247–11256.
Li, G., Muller, M., Thabet, A., Ghanem, B., 2019. Deepgcns: Can gcns go as deep as cnns?, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9267–9276.
Li, R., Li, X., Heng, P.A., Fu, C.W., 2020. Pointaugment: an auto-augmentation framework for point cloud classification, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6378–6387.
Liu, Y., Fan, B., Meng, G., Lu, J., Xiang, S., Pan, C., 2019. Densepoint: Learning densely contextual representation for efficient point cloud processing, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5239–5248.
Maturana, D., Scherer, S., 2015. Voxnet: A 3d convolutional neural network for real-time object recognition, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE. pp. 922–928.
Matveev, A., Artemov, A., Zorin, D., Burnaev, E., 2020. Geometric attention for prediction of differential properties in 3d point clouds, in: IAPR Workshop on Artificial Neural Networks in Pattern Recognition, Springer. pp. 113–124.
Mérigot, Q., Ovsjanikov, M., Guibas, L.J., 2010. Voronoi-based curvature and feature estimation from point clouds. IEEE Transactions on Visualization and Computer Graphics 17, 743–756.
Pistilli, F., Fracastoro, G., Valsesia, D., Magli, E., 2020. Point cloud normal estimation with graph-convolutional neural networks, in: 2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), IEEE. pp. 1–6.
Qi, C.R., Su, H., Mo, K., Guibas, L.J., 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation, in: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 652–660.
Qi, C.R., Yi, L., Su, H., Guibas, L.J., 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems 30.
Simonovsky, M., Komodakis, N., 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3693–3702.
Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E., 2015. Multi-view convolutional neural networks for 3d shape recognition, in: Proceedings of the IEEE International Conference on Computer Vision, pp. 945–953.
Valsesia, D., Fracastoro, G., Magli, E., 2018. Learning localized generative models for 3d point clouds via graph convolution, in: International conference on learning representations.
Wang, C., Samari, B., Siddiqi, K., 2018. Local spectral graph convolution for point set feature learning, in: Proceedings of the European Conference on Computer Vision, pp. 52–66.
Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M., 2019. Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (TOG) 38, 1–12.
Zapata-Impata, B.S., Gil, P., Pomares, J., Torres, F., 2019. Fast geometry-based computation of grasping points on three-dimensional point clouds. International Journal of Advanced Robotic Systems 16, 1729881419831846.
Zhang, K., Hao, M., Wang, J., de Silva, C.W., Fu, C., 2019. Linked dynamic graph cnn: Learning on point cloud via linking hierarchical features. arXiv preprint arXiv:1904.10014 .
Zhou, H., Chen, H., Feng, Y., Wang, Q., Qin, J., Xie, H., Wang, F.L., Wei, M., Wang, J., 2020a. Geometry and learning co-supported normal estimation for unstructured point cloud, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13238–13247.
Zhou, J., Huang, H., Liu, B., Liu, X., 2020b. Normal estimation for 3d point clouds via local plane constraint and multi-scale selection. Computer Aided Design 129, 102916.
Zhu, R., Liu, Y., Dong, Z., Wang, Y., Jiang, T., Wang, W., Yang, B., 2021. Adafit: Rethinking learning-based normal estimation on point clouds, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6118–6127.