PFF-Net: Patch Feature Fitting for Point Cloud Normal Estimation

Qing Li *Member, IEEE*, Huifang Feng, Kanle Shi, Yue Gao *Senior Member, IEEE*, Yi Fang, Yu-Shen Liu *Member, IEEE*, Zhizhong Han

***Abstract*—Estimating the normal of a point requires constructing a local patch to provide center-surrounding context, but determining the appropriate neighborhood size is difficult when dealing with different data or geometries. Existing methods commonly employ various parameter-heavy strategies to extract a full feature description from the input patch. However, they still have difficulties in accurately and efficiently predicting normals for various point clouds. In this work, we present a new idea of feature extraction for robust normal estimation of point clouds. We use the fusion of multi-scale features from different neighborhood sizes to address the issue of selecting reasonable patch sizes for various data or geometries. We seek to model a patch feature fitting (PFF) based on multi-scale features to approximate the optimal geometric description for normal estimation and implement the approximation process via multi-scale feature aggregation and cross-scale feature compensation. The feature aggregation module progressively aggregates the patch features of different scales to the center of the patch and shrinks the patch size by removing points far from the center. It not only enables the network to precisely capture the structure characteristic in a wide range, but also describes highly detailed geometries. The feature compensation module ensures the reusability of features from earlier layers of large scales and reveals associated information in different patch sizes. Our approximation strategy based on aggregating the features of multiple scales enables the model to achieve scale adaptation of varying local patches and deliver the optimal feature description. Extensive experiments demonstrate that our method achieves state-of-the-art performance on both synthetic and real-world datasets with fewer network parameters and running time.**

***Index Terms*—Point clouds, normal estimation, 3D deep learning, feature extraction, surface fitting.**

I. Introduction

POINT cloud normal estimation is one of the basic tasks in 3D computer vision. It has a very wide range of applications and is a prerequisite for many downstream tasks or algorithms, such as surface reconstruction [1], graphics rendering [2]–[4], point cloud denoising [5]–[9] and so on.

Qing Li is with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, China. E-mail: qingli@swjtu.edu.cn

Huifang Feng is with the School of Computer and Software Engineering, Xihua University, Chengdu, China. E-mail: fhf@xhu.edu.cn

Yue Gao and Yu-Shen Liu are with the School of Software, Tsinghua University, Beijing, China. E-mail: {gaoyue, liuyushen}@tsinghua.edu.cn

Kanle Shi is with Kuaishou Technology, Beijing, China. E-mail: shikanle@kuaishou.com

Yi Fang is with the Center for Artificial Intelligence and Robotics, New York University Abu Dhabi, Abu Dhabi, UAE. E-mail: yfang@nyu.edu

Zhizhong Han is with the Department of Computer Science, Wayne State University, Detroit, USA. E-mail: h312h@wayne.edu

The corresponding author is Huifang Feng and Yu-Shen Liu. The main work of this paper was completed at Tsinghua University. The source code, data and pretrained models will be available at https://github.com/LeoQLi/PFF-Net.

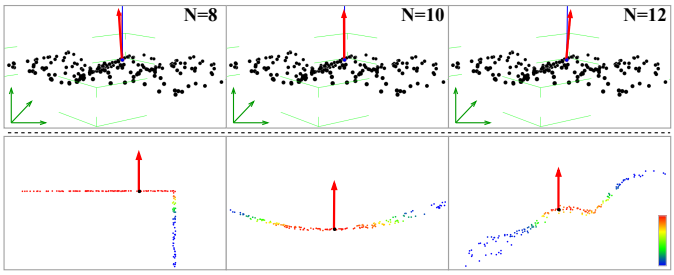


Fig. 1. Normal estimation for point cloud patches. *Top row*: on a 3D surface, the normals estimated by different neighborhood sizes N have different directions. *Bottom row*: the number of neighboring points that can be used to accurately estimate the query point normal varies in different structures. The red points contribute more for normal estimation, and the blue points contribute less.

Although normal estimation has been extensively studied with the development of 3D point cloud processing, it is still challenging under varying noise levels, non-uniform sampling densities, and various complex geometries.

A standard procedure for estimating a query point normal is to build a fixed-scale local patch and analyze its geometry using various techniques [10]–[12]. However, as shown in Fig. 1, it is difficult to choose an appropriate patch size for different data or geometries. A too small patch size can not provide enough neighboring points to capture the local spatial geometric information, while a too large patch size will bring redundancy or dehighlight sharp geometry, degenerating accuracy and efficiency. We provide an experimental analysis in Sec. III. Specifically, (1) for point clouds with noise, a relatively larger size is often a better choice. (2) For structures with smooth planes, it is suitable to select a smaller patch size, and a larger size may not bring performance improvement but redundant points and additional computational burden. (3) For structures with high curvature, the points that are favorable for normal estimation are basically distributed in a small range around the query point, while the points far from the center are mostly irrelevant. Therefore, existing learning-based methods [13]–[19] adopt various techniques to extract features from point cloud patches to fully capture the local geometries.

The key insight of this work is that aggregating features from different neighborhood sizes can address the issue of selecting reasonable patch sizes for various data or geometries. For noisy point clouds, having the central points acquire information in a large neighborhood can make the estimation more robust. For smooth planes or large curvature structures, letting the model focus on points near the center can lead to more efficient and accurate normal estimation. However,

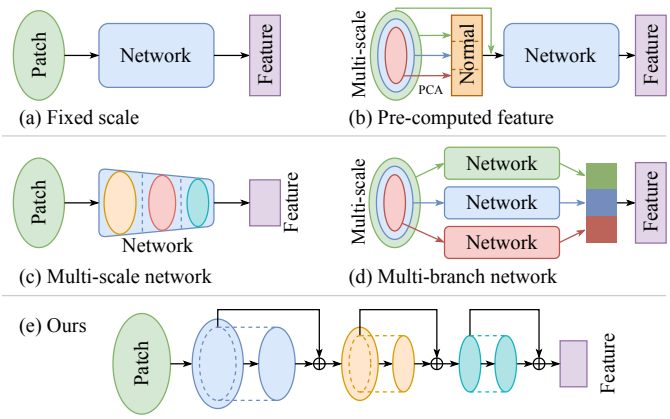


Fig. 2. Previous methods use (a) fixed-scale patches (DeepFit [12], GraphFit [20]), (b) pre-computed features (Refine-Net [15], Zhang *et al*. [21]), (c) multi-scale networks (AdaFit [14], NeAF [17], HSurf-Net [16]) or (d) multi-branch networks (PCPNet [11], Nesti-Net [13], SHS-Net [18], [22]). (e) Our patch feature fitting method for normal estimation.

how to efficiently learn multi-scale features from input patches and effectively make these features reveal the geometry is not properly solved. Existing methods shown in Fig. 2 estimate point normals by first establishing local patches, and then utilizing neural networks to learn to directly or indirectly map their extracted features to 3D normal vectors. However, the methods in Fig. 2(a) do not generalize well to various data. The methods in Fig. 2 (b) and (d) suffer from a large number of network parameters or high computational complexity. The methods in Fig. 2(c) have limited feature learning capabilities with reduced points.

As shown in Fig. 2(e), our method attempts to model a patch feature fitting based on multi-scale features to approximate the optimal geometric description for point cloud normal estimation. We implement an approximation process via multi-scale feature aggregation and cross-scale feature compensation. Unlike surface fitting models [12], [14], [20], [23], [24] that use truncated Taylor expansion to fit 3D surfaces for point cloud patches, we view the feature extraction as a higher-order approximation process in feature space using neural networks. The approximation process is built using multi-scale features from different patch sizes, and the approximation error is considered via cross-scale feature compensation based on attention. Instead of treating all input points equally, we collect more features from points closer to the center of the patch, and transfer features from large scales to small scales for multi-scale feature aggregation. Thus, the model can not only efficiently capture a wide range of spatial information, but also perpetually focus on central points. Our approximation strategy based on weighing the features from multiple scales enables the model to achieve scale adaptation for various geometries. Experimental results on the shape dataset, the real-world indoor and outdoor scene datasets show that our method is robust to domain shift (training on shapes, testing on scenes) and has good generalization capability on real-world LiDAR data. We also demonstrate the extensibility of our method and its ability to improve the performance of other methods. Our main contributions can be summarized as follows.

- We propose a strategy that exploits the idea of patch feature fitting based on multi-scale features to approximate the optimal features for normal estimation.
- We design an effective network architecture that includes multi-scale feature aggregation and cross-scale feature compensation to implement the patch feature fitting and the error in approximation process.
- Extensive evaluation shows that our strategy brings significant performance improvements with fewer parameters and runtime than baseline methods.

II. Related Work

Traditional approaches. The most widely used normal estimation method is based on the classic Principle Component Analysis (PCA) [10], which analyzes the variance in a patch around a query point and defines its normal as the direction of minimal variance [25]. Later, many improvements [26]–[28] have been proposed for PCA. Mitra *et al*. [29] costly investigate the effect of local curvature and point density of the underlying surface to determine the patch size. To preserve more detailed features, other methods use Voronoi cells [30]–[33], Hough transform [34] and edge-aware sampling [35]. Variants that are based on complex surfaces have also been proposed, such as moving least squares [36], jet fitting [23], spherical fitting [37], multi-scale kernel [38], local kernel regression [39] and winding-number field [40]. The data-specific parameters of the above methods often do not generalize well to different data.

Learning-based approaches. (1) *Regression-based methods*. In recent years, learning-based methods have been proposed to directly predict normals from point clouds in a data-driven manner. Some methods [41]–[43] try to map the unstructured point cloud data into a regular domain to extract features. Meanwhile, some alternative approaches learn from raw point clouds. PCPNet [11] and Zhou *et al*. [44] adopt the PointNet architecture [45] to extract patch features from multiple scales. Hashimoto *et al*. [46] use a two-branch network to extract local and spatial features. Nesti-Net [13] tries to search the optimal neighborhood scale but suffers from computational inefficiency. Refine-Net [15] employs a refinement network to optimize the initial normals using the learned local features. HSurf-Net [16] introduces a hyper surface that is parameterized by MLP layers and optimized in high dimensional feature space for normal estimation. NeAF [17] proposes to implicitly learn the angle distance field of points and predict the angle offsets of query vectors. SHS-Net [18], [22] introduces signed hyper surface to learn unoriented and oriented normals. NGLO [47] first predicts coarse normals with global consistency from the global point cloud by learning implicit functions, and then refines the normals based on local information to improve their accuracy. NeuralGF [48] estimates normals in an unsupervised manner by learning gradients of implicit functions. MSECNet [19] improves normal estimation in areas with drastic normal changes by introducing edge detection technology. CMG-Net [49] proposes a metric of Chamfer Normal Distance to address the issue of normal direction inconsistency in noisy point clouds. (2) *Fitting-based methods*. Lenssen *et al*. [50] propose to iteratively parameterize an

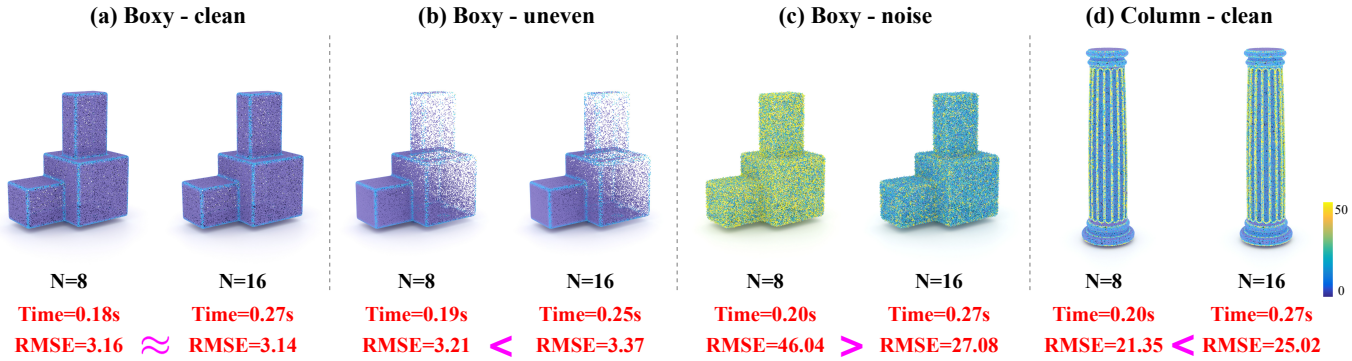


Fig. 3. Normal estimation results of PCA on different point clouds using two different neighbor sizes N. The execution time and average normal angle RMSE are provided under each point cloud. The point color is the normal angle RMSE mapped to a heatmap ranging from $0°$ to $50°$.

adaptive anisotropic kernel to learn weights for a least squares plane fitting. MTRNet [51] uses a differentiable RANSAC to fit a latent tangent plane. DeepFit [12], AdaFit [14], GraphFit [20], Zhou *et al*. [52], Zhang *et al*. [21] and Du *et al*. [24] use a PointNet-like or graph convolutional network to predict point weights and apply a weighted polynomial surface fitting to fit a local surface and calculate its normal. They try to use the weight to provide more reliable inlier points for the fitting. The above methods do not fully explore multi-scale features and effectively filter out redundant information, often have parameter-heavy networks and run inefficiently. Compared to AdaFit, which reduces patch size to simplify explicit surface fitting, our center-based downsampling is designed to enrich geometric representation across scales for direct normal regression. By integrating point-wise weighting, multi-mode feature fusion, and cross-scale compensation, our method achieves higher accuracy with fewer parameters and significantly better efficiency.

III. Preliminary

A. Effect of patch size on normal estimation

To analyze the effect of different patch sizes on point cloud normal estimation, we employ the classic PCA algorithm [10] with k-nearest neighbor sizes $N = 8$ and $N = 16$ to conduct normal calculations on several different point clouds, which cover the common situations including clean (noise-free), noise, non-uniform sampling, simple and complex structures. The results are shown in Fig. 3, we can observe several phenomenons that are in harmony with the intuition: (1) Smaller patch size takes less time, while larger one takes more time. (2) For simple flat structures (boxy shape), the results of different patch sizes are similar, but a larger size leads to redundant information and costs much more runtime. (3) For complex structures (column shape), the useful information for the query point normal estimation is relatively concentrated. A smaller size brings better results, while a larger size gives invalid information or even distractions. (4) For non-uniformly sampled point clouds, a smaller patch size can provide more reasonable structural information from points that are far apart. (5) For noisy point clouds, a larger patch size can effectively suppress the interference caused by noise. This experiment shows that using a large or small patch size has advantages and disadvantages in different situations, and the optimal method should take the advantages of both and avoid the disadvantages of both. To this end, we propose a strategy that exploits the idea of patch feature fitting based on multi-scale features to approximate the optimal features and enables the model to achieve scale adaptation for various geometries in normal estimation. We use a relatively large patch size to extract reliable structural features and suppress noise, and gradually reduce the patch size to focus on the center of the patch and get faster runtime. Meanwhile, the feature aggregation during scale reduction filters out redundant or invalid information to ensure accurate normal estimation.

B. Theoretical motivation from Taylor expansion

Given a point cloud patch $P = \{p_i\}_{i=1}^{N}$ consisting of N points around a query point q, our method aims to estimate the unoriented normal $\mathbf{n}_q$ at point q. Our network design is inspired by the Taylor expansion of an implicit surface function. Let $f : \mathbb{R}^3 \to \mathbb{R}$ be a smooth scalar function that defines a surface by $f(p) = 0$, where $\nabla f(p)$ is aligned with the surface normal at p. For any point p_i in the neighborhood of q, the function can be locally approximated via a second-order Taylor expansion:

$$f(q+X) = f(q)+\nabla f(q)^\top X+\frac{1}{2}X^\top H_f(q)X+R_3(X)\ , \quad (1)$$

where $X = p_i - q$ denotes the local offset, $H_f(q)$ is the Hessian matrix capturing curvature, and $R_3(X)$ represents the third- and higher-order residual. The first two terms $\nabla f(q)^\top X$ and $X^\top H_f(q)X$ correspond to local planar and quadratic geometry, while the residual term accounts for finer surface details. In practice, the surface fitting based methods [12], [14], [20] first approximate a 3D surface by a binary polynomial and then compute the normal of the fitted local surface using the coefficient of the solved polynomial.

In order to directly regress 3D point normals in an end-to-end manner, we approximate the surface function in a feature space and define a learnable feature transformation $F(X)$ based on the offset X. We expect $F(X)$ to capture geometric information such as normals and curvatures. As a result, we view $F(X)$ as a feature-space analogue of a polynomial expansion:

$$F(X) \approx \theta_1^\top X + X^\top \theta_2 X + \cdots\ , \quad (2)$$

where θ_1 and θ_2 are learnable parameters of MLPs approximating surface derivatives, and the terms correspond to surface differential properties. As introduced in Sec. IV-B, we interpret our network as a functional approximation of this expansion, with each residual block modeling a distinct component of the underlying polynomial that characterizes the local surface geometry. Specifically, block $\mathcal{F}_1$ is designed to progressively extract and aggregate multi-scale features $\mathcal{X}$ corresponding to the linear and quadratic terms. Each residual block in $\mathcal{F}_1$ updates the representation as

$$\mathcal{X}_{k+1} = \mathcal{X}_k + \mathrm{MLP}_k\left(\mathcal{X}_k, X_k\right) \ , \tag{3}$$

which can be viewed as fitting the dominant geometric components $\theta_1^\top X$ and $X^\top \theta_2 X$ through stage-wise feature refinement. X_k is also used to denote offset encoding. Each $\mathrm{MLP}_k(\cdot)$ is responsible for modeling one stage of the expansion, and the residual addition naturally mimics term-wise accumulation. In effect, $\mathcal{F}_1$ fits a local planar approximation plus curvature, *i.e.*, the first two low-order terms of the Taylor expansion. This is analogous to classical geometry-fitting: estimating the tangent plane (first-order) and principal curvatures (second-order) via a truncated Taylor expansion (called n-jet) [23].

Subsequently, block $\mathcal{F}_2$ models the remaining high-order variation using finer-scale patches:

$$\mathcal{X}_{\text{refine}} = \mathcal{X}_{\text{coarse}} + \mathrm{MLP}_{\text{refine}}(\mathcal{X}_{\text{coarse}}) \ , \tag{4}$$

serving as a learned residual corrector to capture detailed geometry in complex regions. Additionally, our attention-based cross-scale compensation module in Sec. IV-C reuses coarse-scale features in a weighted manner, further enhancing the fidelity of the approximation.

In summary, our network mimics the progressive accumulation of Taylor terms: block $\mathcal{F}_1$ fits the low-order structure, block $\mathcal{F}_2$ corrects the residual, and cross-scale attention enables explicit compensation. The formulation not only grounds our architecture in geometric theory but also explains the effectiveness of our residual-based multi-scale feature fusion.

IV. Method

Overview. In this work, we introduce a novel multi-scale feature extraction method in normal estimation. Rather than explicitly fitting 3D surfaces using point coordinates, we implement a feature extraction and fusion mechanism based on multiple size scales to allow the network to adaptively find optimal geometric descriptions for input patches with fixed scales from their fused features. Our goal is to use the aggregation of multi-scale features to obtain $F(X)$, which is implicitly defined by the ground truth normal. To this end, we design two kinds of layers with different learning strategies, which are further used to build two kinds of blocks, to extract features at different scales. Moreover, we employ an attention block to provide compensation concerning the approximation error. Fig. 5 shows an overview of the proposed algorithm, which mainly consists of a per-point feature extraction module, two blocks for multi-scale feature aggregation and a cross-scale compensation module using attention mechanism. As the number of layers in the network increases, we decrease the patch size in the latter layers by reducing the nearest neighbors of the query q. The reduction in points effectively reduces the burden on the algorithm and brings higher running efficiency.

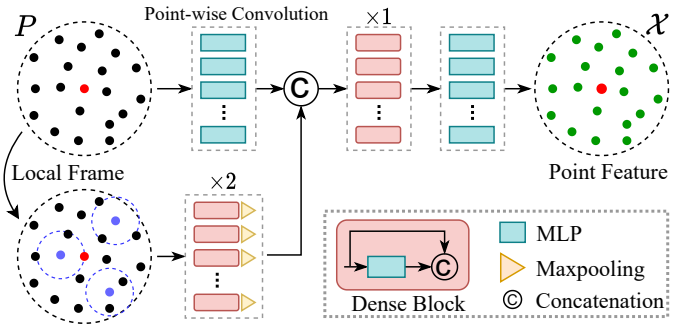


Fig. 4. The architecture of our per-point feature extraction module.

A. Per-point Feature Extraction

We first learn a point-wise feature set $\mathcal{X} = \{x_i\}_{i=1}^N$ for all points of the input patch P. Previous methods [11], [12], [14], [17] employ the PointNet-like structures [45] for feature extraction. However, their network is insufficient to capture local structure information since it does not encode the connections of each point to its neighborhoods. As shown in Fig. 4, we provide a novel feature extraction unit, which is formulated as

$$x_i = \psi\left(\phi(p_i),\ \mathrm{MAX}\left\{\varphi(p_i^j | j = 1, \cdots, n_k)\right\}\right),\ i = 1, \cdots, N, \tag{5}$$

where ψ and ϕ are MLPs, and φ is a stack of densely connected graph convolution layers. $p_i^j \in \mathrm{kNN}(p_i)$ denotes the n_k-nearest neighboring points of p_i. $\mathrm{MAX}\{\cdot\}$ means maxpooling. The function $\psi(\cdot,\cdot)$ fuses the point-wise features and local features for each point by concatenation. The graph convolution extracts local features for each point in the patch, while the dense connection delivers features with richer contextual information [53], [54].

B. Multi-scale Feature Aggregation

A template. To construct the feature-based polynomial, we learn the multi-scale feature based on the per-point feature $\mathcal{X}$. Specifically, we extract features from different patch sizes and then fuse the features to realize feature aggregation. During this process, we gradually reduce the patch size around the query point q by removing neighboring points that are far away from it, and aggregate the information from the removed far points to the remaining points in the patch, thus realizing focalization and concentration. Meanwhile, the reduction in points can improve running efficiency. In summary, the basic convolutional layer $\mathcal{P}$ is formulated as

$$y_i = \alpha\left(\beta\left(\mathrm{MAX}\left\{\gamma(w_j \cdot x_j)\right\}_{j=1}^{N_k}\right),\ x_i\right), i = 1, \cdots, N_{k+1} \ , \tag{6}$$

where α, β and γ are MLPs, x_i and y_i are per-point features. Note that the features are sorted in ascending order according to the distances between the underlying points and the query, so that the index can be used to efficiently remove points

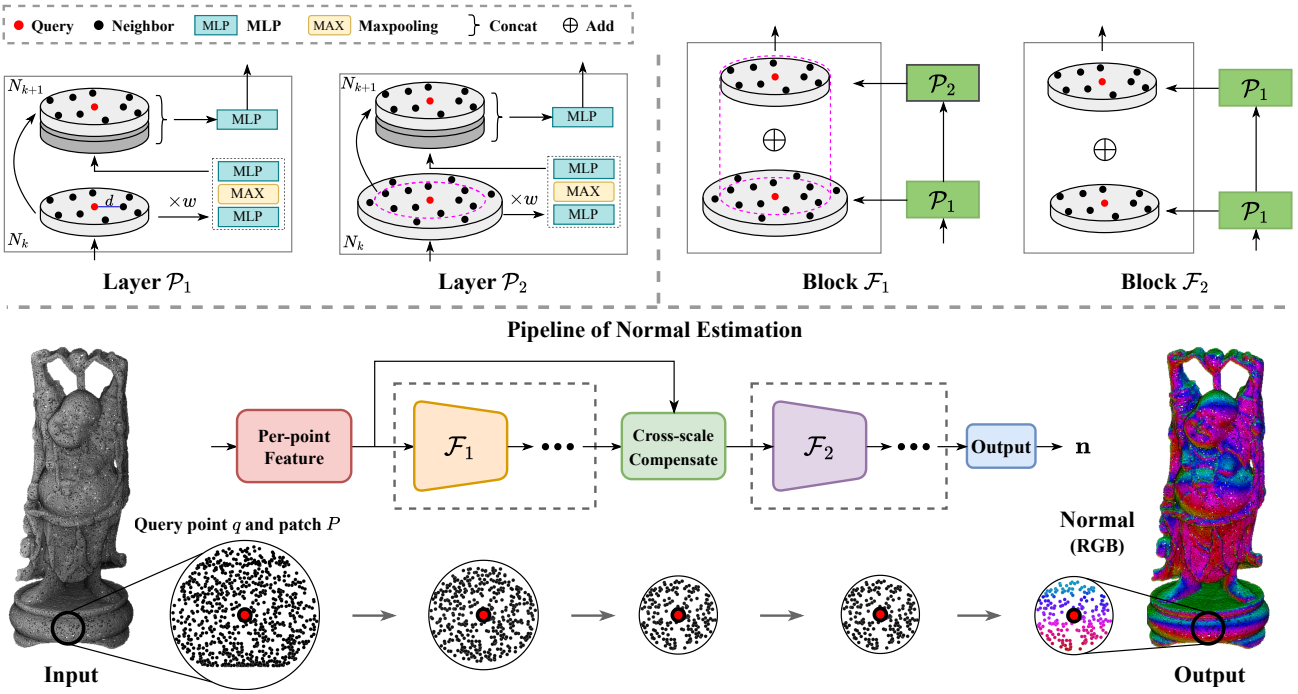


Fig. 5. An overview of our normal estimation method. The blocks $\mathcal{F}_1$ and $\mathcal{F}_2$ are built by using different layers $\mathcal{P}_1$ and $\mathcal{P}_2$. Two blocks are stacked recursively to form the normal estimation pipeline. The color of the shape point cloud indicates the predicted normals.

and their features that are far away. k is the size index and $N_{k+1} \leqslant N_k \leqslant N$. MAX$\{\cdot\}$ represents the feature maxpooling over N_k points in the patch. w is a distance-based weight [18], [22], and is calculated by

$$w_j = \frac{d_j}{\sum_{i=1}^{N} d_i}, \ d_i = \sigma\big(a - b||p_i - q||_2\big), \tag{7}$$

where σ is the sigmoid function, a and b are learnable parameters with the initial value set to 1. We use the weight w to let the network focus more on the points p_i that are closer to the query point q, thereby extracting reliable features in areas where the geometry changes drastically and improving the robustness of the algorithm. Although our feature aggregation framework shares conceptual similarities with SHS-Net [18], it is fundamentally different in motivation and scope of feature encoding. SHS-Net adopts a simplified and lightweight fusion of global and local features, while our approach is rooted in Taylor expansion theory and performs hierarchical residual fusion solely within local neighborhoods, enabling finer geometric approximation.

Two variants. Next, we will build two different layers and two different blocks based on Eq. (6). As shown in Fig. 5, we provide two types of layers $\mathcal{P}_1$ and $\mathcal{P}_2$ depending on whether the patch size is reduced or not, *i.e.*, $N_{k+1} = N_k$ or $N_{k+1} < N_k$. These two layers $\mathcal{P}_1$ and $\mathcal{P}_2$ can be stacked alternately, enabling the model to find increasingly rich representations of the point cloud patch. In our normal estimation pipeline, we further build two different blocks $\mathcal{F}_1$ and $\mathcal{F}_2$ using the different combinations of layers $\mathcal{P}_1$ and $\mathcal{P}_2$. For block $\mathcal{F}_1$, the features from two layers $\mathcal{P}_1$ and $\mathcal{P}_2$ are aggregated by addition operation and passed to the next layer, *i.e.*,

$$\mathcal{X}_{k+1} = [\mathcal{X}_k]_{N_{k+1}} + \mathcal{P}_2(\mathcal{X}_k), \quad \mathcal{X}_k = \mathcal{P}_1(\mathcal{X}_{k-1}), \tag{8}$$

where $[\cdot]_{N_{k+1}}$ means taking the features of the nearest neighbor point of q whose neighborhood size is N_{k+1}. Note that we sort the input points and their features according to their 3D distance from q and keep the order unchanged during processing, thus achieving fast indexing. For block $\mathcal{F}_2$, the features from two $\mathcal{P}_1$ layers are aggregated by addition operation and then passed to the next layer, *i.e.*,

$$\mathcal{X}_{k+1} = \mathcal{X}_k + \mathcal{P}_1(\mathcal{X}_k), \quad \mathcal{X}_k = \mathcal{P}_1(\mathcal{X}_{k-1}), \tag{9}$$

The block $\mathcal{F}_1$ extracts wide-range neighborhood features and gradually reduces the patch size to filter out redundant or irrelevant features, while the block $\mathcal{F}_2$ further refines these features. With the recursive utilization of our blocks, the large patch sizes of earlier layers provide more information about the underlying geometries, while the small patch sizes of the latter layers result in a more accurate description of the central details. By utilizing their combination, we can use fewer network parameters to extract information useful for normal estimation, and the reduction in patch size can also improve the algorithm's running efficiency.

C. Cross-scale Compensation

Generally, there is information loss during the feature extraction process, which leads to description errors and affects feature approximation. The residual connection [56] that reuses the features from earlier layers can stabilize training and convergence. In this work, we introduce an alternative method and propose to weigh features at different scales via distance-weighted attention, and generate compensation in geometric

TABLE I
NORMAL RMSE ON THE DATASETS PCPNET AND FAMOUSSHAPE. THE LOWER THE BETTER.

Category	PCPNet Dataset							FamousShape Dataset						
	Noise				Density		Average	Noise				Density		Average
	None	Low	Medium	High	Stripe	Gradient		None	Low	Medium	High	Stripe	Gradient	
PCV [55]	12.50	13.99	18.90	28.51	13.08	13.59	16.76	21.82	22.20	31.61	46.13	20.49	19.88	27.02
Jet [23]	12.35	12.84	18.33	27.68	13.39	13.13	16.29	20.11	20.57	31.34	45.19	18.82	18.69	25.79
PCA [10]	12.29	12.87	18.38	27.52	13.66	12.81	16.25	19.90	20.60	31.33	45.00	19.84	18.54	25.87
PCPNet [11]	9.64	11.51	18.27	22.84	11.73	13.46	14.58	18.47	21.07	32.60	39.93	18.14	19.50	24.95
Zhou *et al.* [44]	8.67	10.49	17.62	24.14	10.29	10.66	13.62	-	-	-	-	-	-	-
NeuralGF [48]	7.89	9.85	18.62	24.89	9.21	9.29	13.29	13.74	16.51	31.05	40.68	13.95	13.17	21.52
Nesti-Net [13]	7.06	10.24	17.77	22.31	8.64	8.95	12.49	11.60	16.80	31.61	39.22	12.33	11.77	20.55
Lenssen *et al.* [50]	6.72	9.95	17.18	21.96	7.73	7.51	11.84	11.62	16.97	30.62	39.43	11.21	10.76	20.10
DeepFit [12]	6.51	9.21	16.73	23.12	7.92	7.31	11.80	11.21	16.39	29.84	39.95	11.84	10.54	19.96
MTRNet [51]	6.43	9.69	17.08	22.23	8.39	6.89	11.78	-	-	-	-	-	-	-
Refine-Net [15]	5.92	9.04	16.52	22.19	7.70	7.20	11.43	-	-	-	-	-	-	-
Zhang *et al.* [21]	5.65	9.19	16.78	22.93	6.68	6.29	11.25	9.83	16.13	29.81	39.81	9.72	9.19	19.08
Zhou *et al.* [52]	5.90	9.10	16.50	22.08	6.79	6.40	11.13	-	-	-	-	-	-	-
AdaFit [14]	5.19	9.05	16.45	21.94	6.01	5.90	10.76	9.09	15.78	29.78	38.74	8.52	8.57	18.41
GraphFit [20]	5.21	8.96	16.12	21.71	6.30	5.86	10.69	8.91	15.73	29.37	38.67	9.10	8.62	18.40
NeAF [17]	4.20	9.25	16.35	21.74	4.89	4.88	10.22	7.67	15.67	29.75	38.76	7.22	7.47	17.76
HSurf-Net [16]	4.17	8.78	16.25	21.61	4.98	4.86	10.11	7.59	15.64	29.43	38.54	7.63	7.40	17.70
NGLO [47]	4.06	8.70	16.12	21.65	4.80	4.56	9.98	7.25	15.60	29.35	38.74	7.60	7.20	17.62
Du *et al.* [24]	3.85	8.67	16.11	21.75	4.78	4.63	9.96	6.92	15.05	29.49	38.73	7.19	6.92	17.38
SHS-Net [18]	3.95	8.55	16.13	21.53	4.91	4.67	9.96	7.41	15.34	29.33	38.56	7.74	7.28	17.61
CMG-Net [49]	3.87	8.45	16.08	21.89	4.85	4.45	9.93	7.07	14.83	29.04	38.93	7.43	7.03	17.39
MSECNet [19]	3.84	8.74	16.10	21.05	4.34	4.51	9.76	6.85	15.60	29.22	**38.13**	**6.64**	6.65	17.18
Ours	**3.32**	**8.34**	**15.63**	**20.94**	**4.10**	**3.92**	**9.38**	**6.60**	**14.68**	**28.86**	38.27	6.86	**6.41**	**16.95**

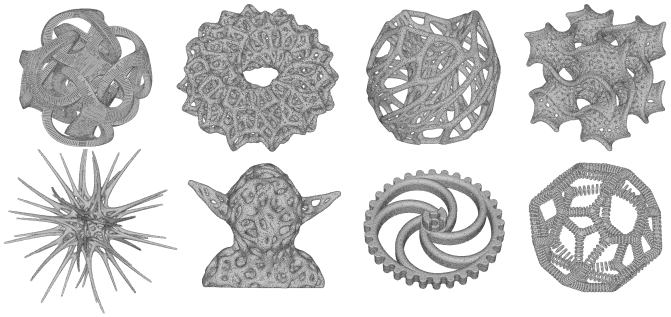


Fig. 6. Examples of point cloud shapes from our NestPC dataset.

description. First, we generate several feature components through the following formulas

$$Q_i = \theta_Q(y_i), K_i = \theta_K(x_i), V_i = \theta_V(x_i), \Delta_i = \theta_\Delta(Q_i + K_i), \tag{10}$$

where θ is MLP, $x_i \in \mathcal{X}$ is the per-point feature before block $\mathcal{F}_1$ and $y_i \in \mathcal{Y}$ is the feature after block $\mathcal{F}_1$. Then, the cross-scale feature compensation is implemented by an attention block conditioned on $\mathcal{Y}$, that is

$$z_i = \eta\left(w_i \cdot V_i \cdot \mu(\Delta_i), \quad y_i\right), \ i = 1, \cdots, N_k \ , \tag{11}$$

where η and μ are MLPs. μ provides attention weights to modulate individual features, while $\eta(\cdot, \cdot)$ fuses two kinds of features through concatenation. w is the distance-based weight in Eq. (7). We do not use a *softmax* operation for normalization as in [57], [58] since it does not bring performance gains, which is verified by ablation experiments.

D. Normal Prediction and Training Loss

After obtaining the final output feature $\mathcal{X}_o$ of the remaining nearest N_o points using block $\mathcal{F}_2$, the unnormalized normal of query point $\mathbf{n}_q$ is predicted by a weighted maxpooling of its neighboring features $x_i \in \mathcal{X}_o$, *i.e.*,

$$\mathbf{n}_q = \delta\big(\text{MAX}\{w_i \cdot \tau_i \cdot x_i | i = 1, \cdots, N_o\}\big), \tag{12}$$

where $\tau_i = \text{sigmoid}(\xi(x_i))$ is a weight. δ and ξ are MLPs. The neighboring point normals $\mathbf{n}_i$ are predicted from $\mathcal{X}_o$ by another MLP. To train the network to predict accurate normal, we calculate the *sin* distance d_{sin} and squared Euclidean distance d_{euc} between the predicted normal $\mathbf{n}$ and the ground truth $\hat{\mathbf{n}}$, then we have

$$\mathcal{L}_\mathbf{n} = d_{\text{sin}} + d_{\text{euc}} = \|\mathbf{n} \times \hat{\mathbf{n}}\| + \min\left(\|\mathbf{n} - \hat{\mathbf{n}}\|^2, \|\mathbf{n} + \hat{\mathbf{n}}\|^2\right). \tag{13}$$

Thus, we obtain the loss $\mathcal{L}_\mathbf{n}^q$ for query point normal $\mathbf{n}_q$ and the mean loss $\mathcal{L}_\mathbf{n}^p$ for neighboring point normals $\mathbf{n}_i$. Moreover, to facilitate the learning of τ in Eq. (12), we adopt a weight loss based on coplanarity [21],

$$\mathcal{L}_\tau = \frac{1}{N}\sum_{i=1}^{N}(\tau_i - \hat{\tau}_i)^2, \quad \hat{\tau}_i = \exp\left(-\frac{(p_i \cdot \hat{\mathbf{n}}_q)^2}{\varepsilon^2}\right), \tag{14}$$

where $\varepsilon = \max(0.0025, \ 0.3\sum_{i=1}^{N}(p_i \cdot \hat{\mathbf{n}}_q)^2/N)$. In summary, the final training loss function is given by

$$\mathcal{L} = \lambda_1 \mathcal{L}_\mathbf{n}^q + \lambda_2 \mathcal{L}_\mathbf{n}^p + \lambda_3 \mathcal{L}_\tau, \tag{15}$$

where the weight factors $\lambda_1 = 0.1$, $\lambda_2 = 0.4$ and $\lambda_3 = 1.0$ are first set empirically, and then fine-tuned according to the experimental results.

V. EXPERIMENTS

Implementation. We follow the same experimental setup in [11], [16] and train our network only on the training set of the PCPNet dataset [11]. In this dataset, each shape corresponds to a clean point cloud and other point clouds

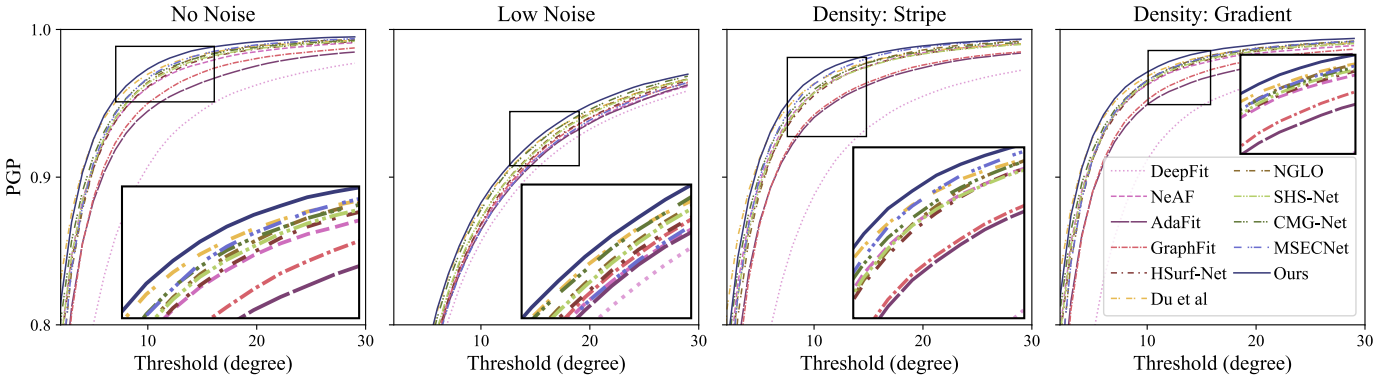


Fig. 7. Normal PGP curves on the PCPNet dataset. The Y-axis is the percentage of good point normals whose normal errors are smaller than the given angle threshold of the X-axis (in degrees). More graphs with a different axis scale are shown in Fig. 8.

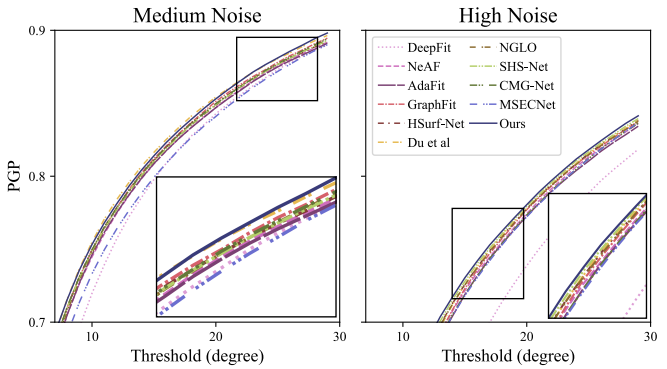


Fig. 8. Normal PGP curves on the PCPNet dataset. The scale of the axes is different from Fig. 7, so we show it separately for better presentation.

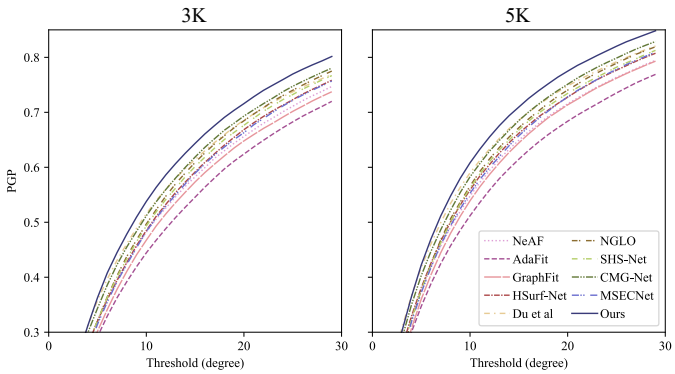


Fig. 9. Normal PGP curves on sparse point cloud data with 3000 and 5000 points, respectively.

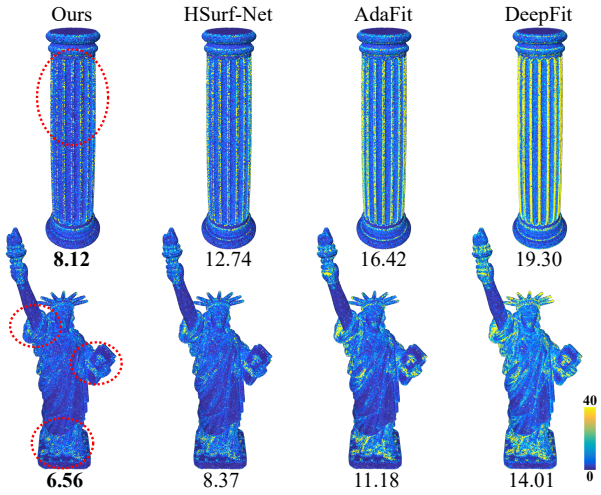


Fig. 10. Normal error visualization on complex shapes of the PCPNet dataset. The normal RMSE is mapped to a heatmap ($0^\circ - 40^\circ$), and the average value is provided under each shape.

generated by five different strategies, including three levels of Gaussian noise with standard deviations of 0.12% (low), 0.6% (medium) and 1.2% (high) of the shape bounding box diagonal and two types of non-uniform sampling (stripe and gradient). In each training epoch, we randomly sample 1000 query points from each shape as centers and build patches around them. Each patch is centered on the query point and normalized to a unit sphere. We set the input patch size $N = 800$, the size scale set $N_k = \{N/b^s\}_{s=0}^{L}$, where $L = 2$ (for the number of size scales) and $b = 2$ (for the size of each scale). The number of kNN points in the per-point feature extraction is $n_k = 16$. We train the model for 800 epochs, and adopt the AdamW optimizer with an initial learning rate of 0.001, which is decayed to $1/5$ of the latest value at epochs $\{400, 600\}$.

Metric. We use the normal angle Root Mean Squared Error (RMSE) as the evaluation metric and the Percentage of Good Points (PGP) to count points with qualified normals [11], [16].

NestPC dataset. In this study, we present a new dataset for 3D point cloud normal estimation evaluation. The number of datasets in this research field is very limited. The data in commonly used point cloud normal estimation datasets, such as PCPNet [11] and FamousShape [18], have relatively simple topological structures. To comprehensively evaluate the performance of the proposed algorithm, we collect shapes with complex topology and sample point clouds from the provided mesh data. The ground-truth normals of the point clouds are calculated from the mesh data and used for evaluation. As shown in Fig. 6, we call this new dataset *NestPC*, which will be publicly available along with our source code.

A. Baseline Methods

In our experiments, we compare our method with various baseline methods, which are mainly classified into the following three categories: (1) the traditional normal estimation methods, such as PCA [10], Jet [23] and GCNO [40]; (2) the learning-based methods using the surface fitting, such as

TABLE II
COMPARISON OF PGP-20° ON THE PCPNET AND FAMOUSSHAPE DATASETS UNDER THE HIGHEST NOISE. THE HIGHER THE BETTER.

(%)	AdaFit [14]	GraphFit [20]	NeAF [17]	HSurf-Net [16]	Du *et al.* [24]	NGLO [47]	SHS-Net [18]	CMG-Net [49]	MSECNet [19]	Ours
PCPNet	77.26	77.71	77.44	77.77	77.79	77.76	77.94	77.35	77.22	**78.05**
FamousShape	43.95	43.99	44.05	44.62	43.97	44.19	44.67	43.18	44.70	**44.99**

TABLE III
NORMAL RMSE ON SPARSE POINT CLOUD DATA WITH 3000 AND 5000 POINTS.

	AdaFit [14]	GraphFit [20]	NeAF [17]	HSurf-Net [16]	Du *et al.* [24]	NGLO [47]	SHS-Net [18]	CMG-Net [49]	MSECNet [19]	GCNO [40]	Ours
3K	30.46	29.56	28.64	27.74	27.18	26.78	27.22	26.13	27.32	27.87	**24.50**
5K	27.05	25.47	25.10	23.93	23.30	23.07	23.55	22.40	23.64	31.54	**20.91**

TABLE IV
OUR METHOD CAN IMPROVE THE PERFORMANCE OF OTHER METHODS ON THE PCPNET DATASET.

	PCPNet [11]	DeepFit [12]	NeAF [17]	HSurf-Net [16]	NGLO [47]	CMG-Net [49]
Original	14.58	11.80	10.22	10.11	9.98	9.93
w/ Ours-1	9.71 (↓4.87)	9.69 (↓2.11)	9.69 (↓0.53)	9.69 (↓0.42)	9.69 (↓0.29)	9.69 (↓0.24)
w/ Ours-2	10.30 (↓4.28)	10.75 (↓1.05)	10.06 (↓0.16)	9.82 (↓0.29)	9.71 (↓0.27)	9.87 (↓0.06)

'Ours-1' indicates normal optimization and 'Ours-2' indicates network module integration.

TABLE V
COMPARISON OF THE NETWORK PARAMETER (MILLION) AND THE AVERAGE INFERENCE TIME (SECONDS PER 100K POINTS).

	AdaFit [14]	GraphFit [20]	NeAF [17]	HSurf-Net [16]	Du *et al.* [24]	NGLO [47]	SHS-Net [18]	CMG-Net [49]	MSECNet [19]	Ours
Param.	4.87	4.26	6.74	2.16	4.46	0.46+1.92	3.27	2.70	10.40	**2.03**
Time	56.23	292.12	400.81	72.47	295.69	0.56+70.77	65.89	109.98	0.61*	**5.06 (0.11*)**

∗ represents patch-to-patch normal estimation for the entire shape [19].

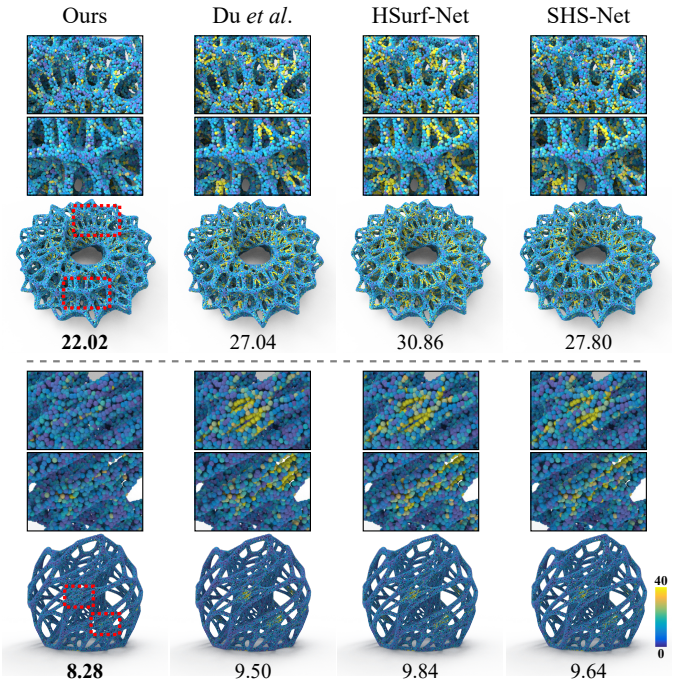


Fig. 11. Visual comparison of normal errors on two shapes of the NestPC dataset. The normal RMSE is mapped to a heatmap, and the average value is provided under each shape.

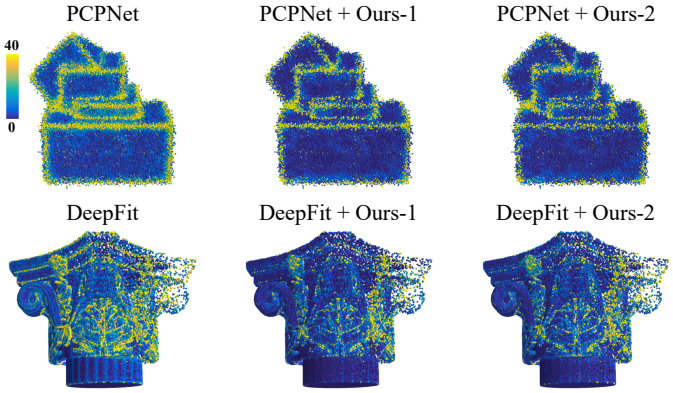


Fig. 12. Visual comparison of normal errors on noisy and non-uniform point clouds. Our algorithm can be used to improve other methods.

TABLE VI
NORMAL RMSE ON THE NESTPC DATASET.

GraphFit	HSurf-Net	Du *et al.*	NGLO	SHS-Net	CMG-Net	MSECNet	Ours
18.44	13.38	13.36	12.79	13.37	13.99	13.89	**11.62**

DeepFit [12], AdaFit [14], GraphFit [20] and Du *et al.* [24]; (3) the learning-based methods for normal regression, such as PCPNet [11], Nesti-Net [13], Refine-Net [15], HSurf-Net [16], SHS-Net [18], [22], NeuralGF [48] and MSECNet [19].

For quantitative comparisons on the PCPNet dataset [11], the numerical results of some baseline methods are taken from their papers due to the unavailable source codes, including Zhou *et al.* [44], MTRNet [51] and Zhou *et al.* [52]. For the method of Zhang *et al.* [21], we only feed the 3D point clouds into their network model to predict normals since their precomputed features are not available. For GraphFit [20], we train a neural network model from scratch using the official source code. For the algorithm of Du *et al.* [24], we use GraphFit [20] as the backbone network.

TABLE VII
NORMAL RMSE ON THE SCENENN DATASET.

	HSurf-Net	Du *et al.*	NGLO	SHS-Net	CMG-Net	MSECNet	Ours
Clean	7.55	7.68	7.73	7.93	7.64	**6.94**	7.28
Noise	12.23	11.72	12.25	12.40	11.82	11.66	**11.47**

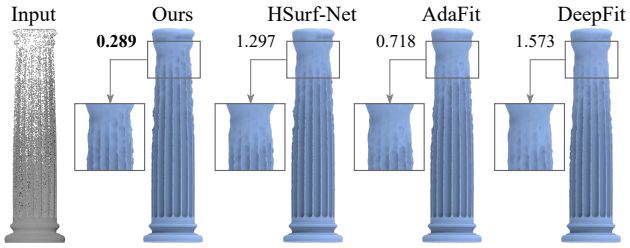


Fig. 13. Comparison of the reconstructed surfaces from a non-uniformly sampled point cloud using the estimated normals. The chamfer distance ($\times 10^{-4}$) is provided.

B. Results on Shape Data

The quantitative comparison results on the datasets PCPNet [11] and FamousShape [18] are reported in Table I. Our method achieves significant performance improvement, and provides more accurate normal results than baseline methods under most noise levels and density variations. We show the normal PGP of different data categories in Fig. 7 and Table II. We can see that our method has the best performance at almost all thresholds. In Fig. 8, we show more results of normal PGP on noisy point clouds of the PCPNet dataset. It can be seen that our method has a performance advantage compared to the baseline methods. Fig. 10 shows a visual comparison of the normal errors on different shapes, where the point clouds are rendered in RGB colors according to the error map. We provide the quantitative comparison results on our NestPC dataset in Table VI. Moreover, Fig. 11 shows a visual comparison of normal errors on two shapes with complex topology and geometry. These evaluation results all demonstrate the excellent performance of our algorithm.

Sparse data. We perform a quantitative evaluation on two sets of point clouds that have the same shapes as the FamousShape dataset [18], but each shape in these two sets contains only 3000 and 5000 points, respectively. As shown in Table III, we report quantitative comparison results of unoriented normals on these two data sets. We can see that our method has the lowest RMSE result. Fig. 9 shows the normal PGP of each evaluated method. These results demonstrate the good performance of our method on sparse point clouds.

C. Results on Scene Data

To evaluate the generalization ability of our method, we directly use the model trained on the PCPNet shape dataset to do testing on the real scene data. The SceneNN dataset [59] provides RGB-D data captured in various real-world room scenes and all scenes are reconstructed as triangle meshes. We use the preprocessed data from [16] for normal evaluation. The ground-truth normals of this dataset are calculated from the provided mesh data. We report the quantitative comparison results in Table VII, and our method achieves better performance than baseline methods in the data category of noise.

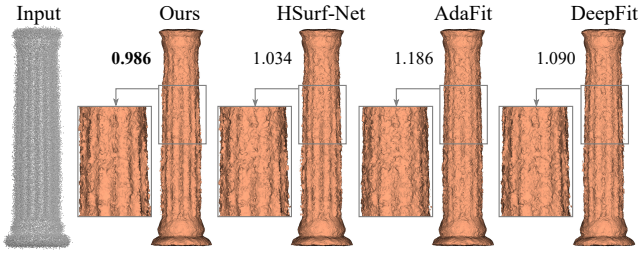


Fig. 14. Comparison of the reconstructed surfaces from the denoised point clouds. The noisy point cloud is first denoised using the estimated normals. The chamfer distance ($\times 10^{-4}$) is provided.

In Fig. 15, we visualize the normal RMSE on point clouds of several indoor scenes provided by the SceneNN dataset. Our method achieves the lowest normal estimation error in these scenes with complex geometries.

D. Applications

Improve other methods. We provide two types of experiments to illustrate that our algorithm can further improve the performance of other methods:

(1) *Ours-1*: We do not make any modifications or adjustments to other methods, but directly optimize their output results. Following the normal optimization in [47], we first modify our network and train it to predict the angle distance between an arbitrary input vector and the true normal, rather than predicting the normal. Then, we construct vector samples in spherical space based on the normal results of other methods. Finally, the vector samples are used as the input to predict the angle of each sample, and the sample with the smallest output angle is selected as the optimal normal.

(2) *Ours-2*: We replace the main structure of the feature extraction network of other methods with our designed modules, namely multi-scale feature aggregation layer and cross-scale compensation layer. We do not change the input data and training losses of other methods, and keep the parameters of the new network models as close as possible to the same order of magnitude as their original network models. The results of the above two experiments (*i.e.*, Ours-1 and Ours-2) are shown in Table IV and Fig. 12, which demonstrate the good scalability and portability of our approach.

Surface reconstruction. To investigate the effect of the estimated normals, we use Poisson reconstruction [1] to reconstruct object or scene surfaces from point clouds. The reconstructed object surfaces using the estimated normals are visualized in Fig. 13. More surface reconstruction results of scene data are shown in Figures 17 and 18. The results show that the reconstruction algorithm can benefit from the normals estimated by our method to recover more detail in sparse or sharp areas. We also provide qualitative comparison results on wireframe-type point clouds with less than 1000 points. As shown in Fig. 16, our method can handle sparse and non-uniformly sampled point clouds.

Point cloud denoising. We adopt a denoising method [7] with default parameters to filter the input noisy point cloud based on the normals estimated by different methods. The comparison of reconstructed surfaces from the denoised point clouds is

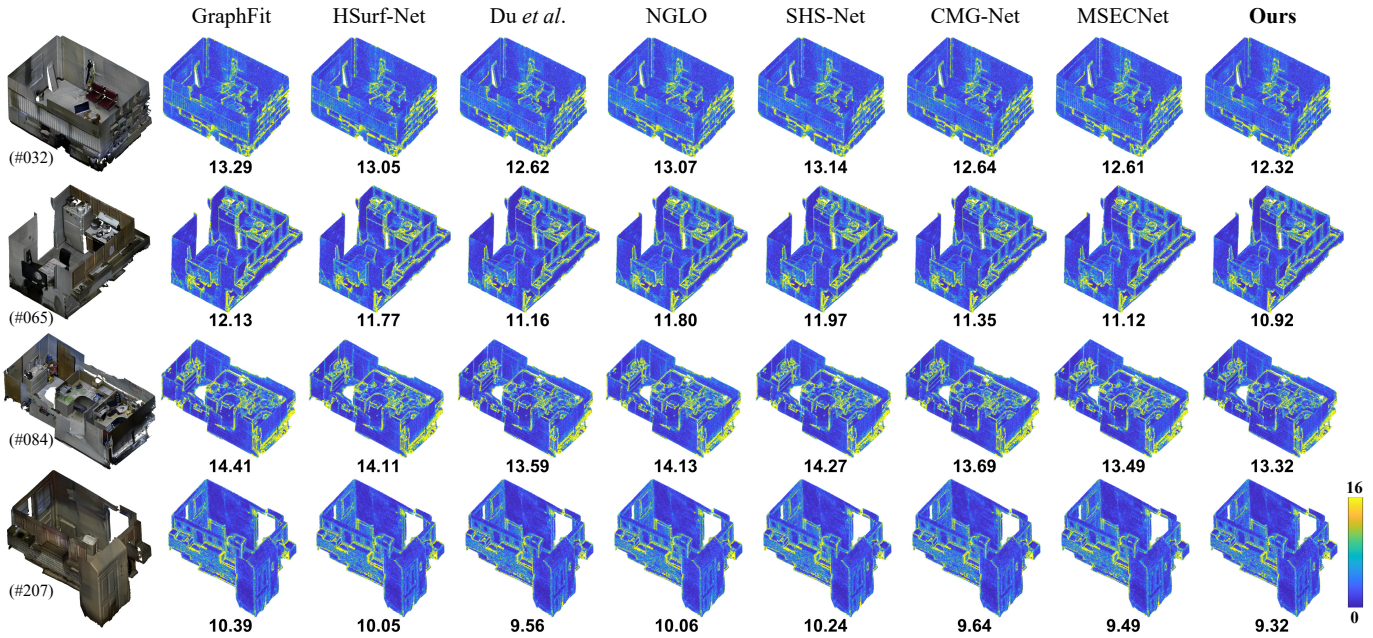


Fig. 15. Comparison of normal RMSE on noisy point clouds of the SceneNN dataset. The colors on point clouds denote the normal errors mapped to a heatmap ranging from 0° to 16°. The average RMSE of the entire point cloud is provided under each point cloud. The first column shows real-world indoor scenes with RGB colors.

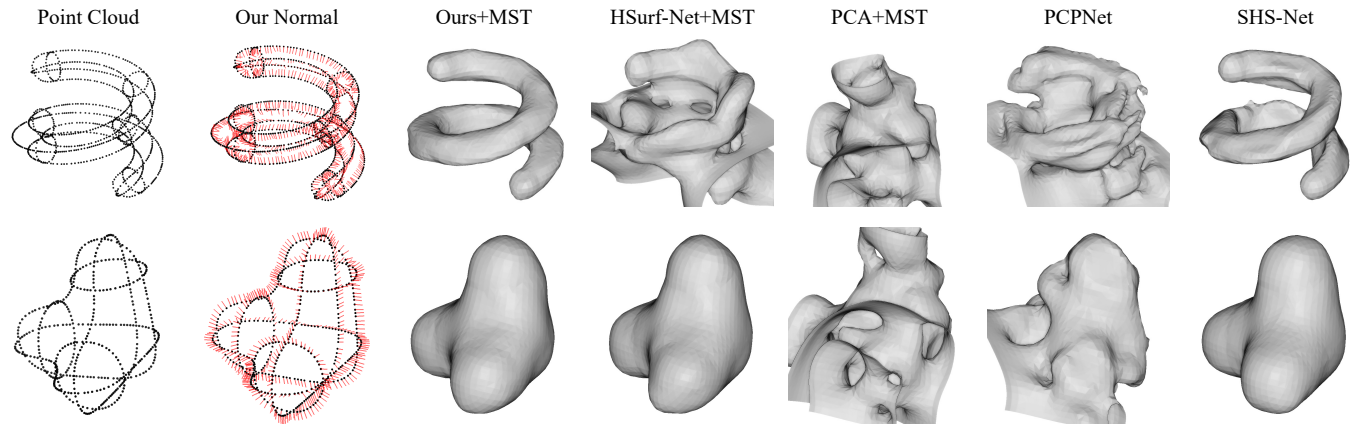


Fig. 16. Visual comparison on wireframe point clouds with sparse and non-uniform sampling. The input point clouds contain less than 1000 points. Our normals (red lines), as well as the normals of HSurf-Net and PCA, are reoriented via MST [10], and PCPNet and SHS-Net can estimate oriented normals.

shown in Fig. 14. The results indicate that our estimated normals can facilitate the denoising algorithm to keep more complete structures and details.

E. More Results on Different Data

In this section, we will show that our method (only trained on the PCPNet shape data) can generalize well to new and different types of point cloud data, such as RGB-D data of indoor scenes and LiDAR data of outdoor scenes.

Kinect data. We use the RGB-D Scenes Dataset [60] to test the generalization ability of our method to real-world scene data. This dataset is captured in indoor room scenes and the ground-truth normal is unavailable. We adopt ODP algorithm [61] to make the estimated normals have a consistent orientation and employ Poisson reconstruction algorithm [62] to reconstruct surfaces. As shown in Fig. 17, we visualize the surface reconstruction results based on the normals estimated by different methods. The results show that our method gives better shapes of objects in the room.

KITTI dataset. The KITTI dataset [63] is captured from real-world street scenes using a laser scanner, but its point cloud data is sparser in points compared to the Paris-rue-Madame dataset. Since the ground-truth normal is unavailable, we use ODP algorithm [61] and Poisson reconstruction algorithm [62] to reconstruct surfaces from the estimated normals. As shown in Fig. 19, we show a visual comparison of the estimated normals. In Fig. 18, we provide a visual comparison of the reconstructed surfaces based on the normals.

Semantic3D dataset. Different from the above datasets, the Semantic3D dataset [64] is obtained by scanning different scenic spots or buildings with LiDAR. It has a larger scene scale and contains more points in a single point cloud. As the ground truth normal of the Semantic3D dataset is not available, we show a visual comparison in Fig. 21. We can

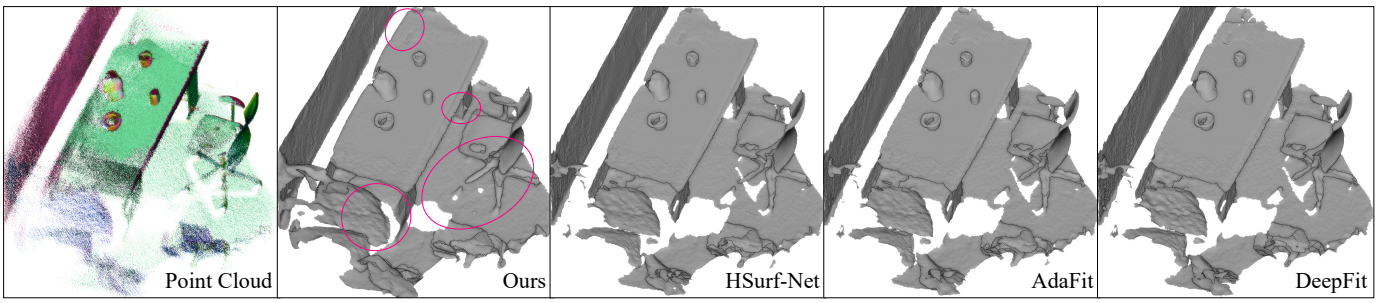


Fig. 17. Surfaces reconstructed using normals estimated by different methods on the RGB-D Scenes Dataset. The generated surfaces are cropped using the raw point cloud, leaving holes in the wrong surfaces.

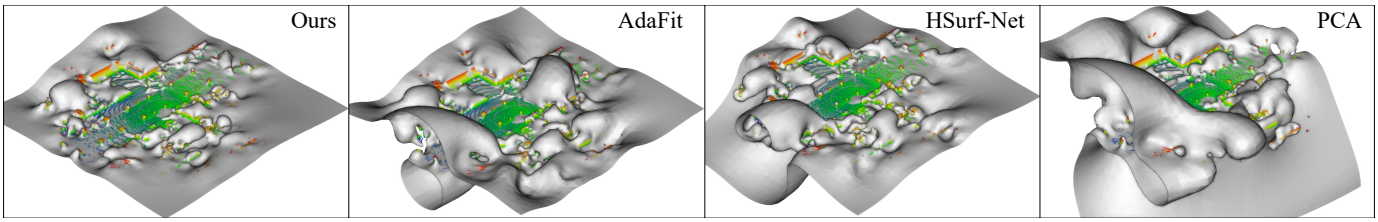


Fig. 18. The reconstructed surfaces using normals estimated by different methods on the KITTI dataset. Points are colored by their height values.

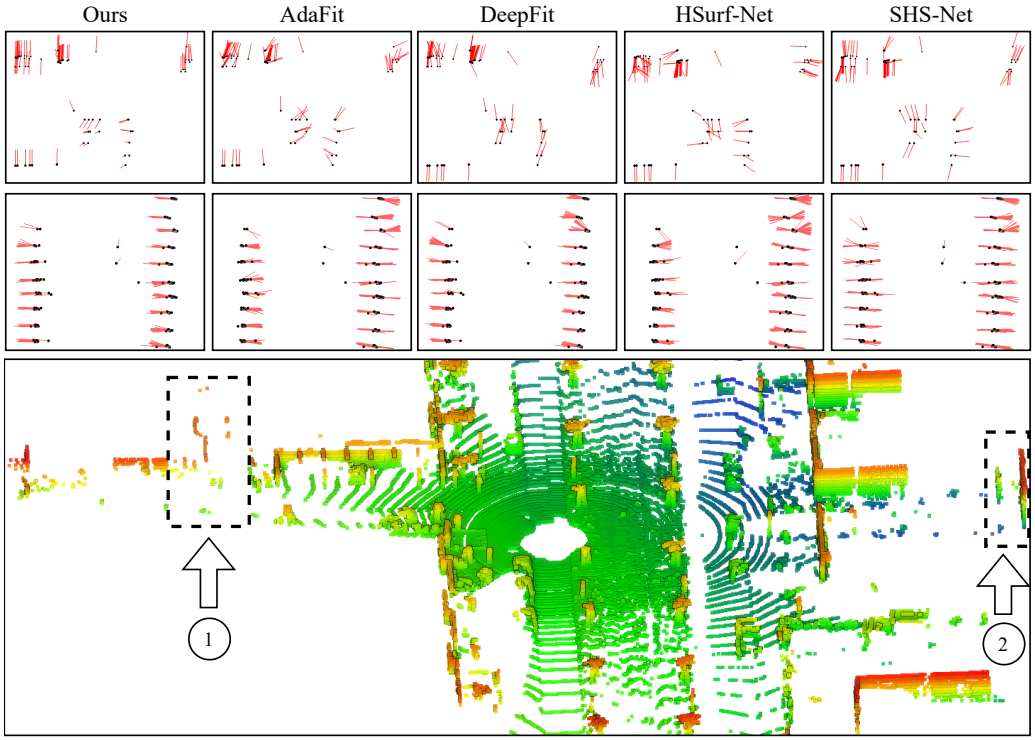


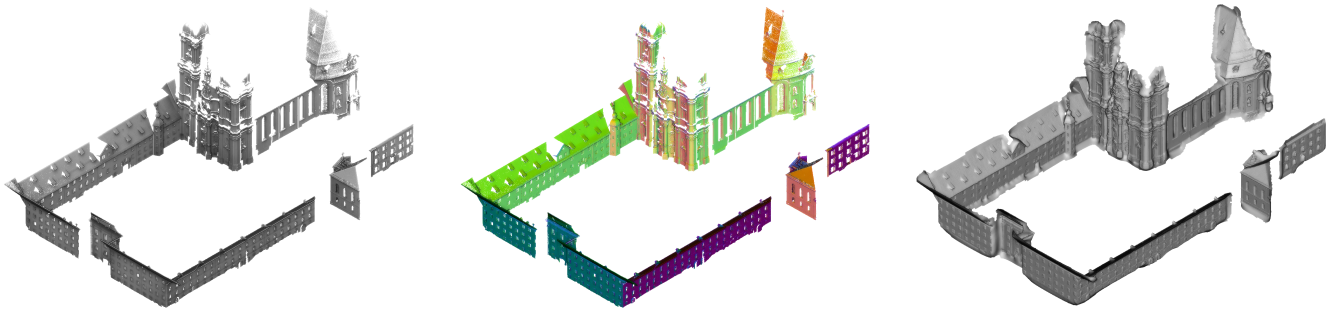
Fig. 19. Visual comparison of the estimated normal vectors (red lines in the first two rows) on the KITTI dataset. In extremely sparse and noisy planar regions, the parallelism between our estimated normals is better. Points in the third row are colored by their height values.

see that our method reveals the fine details of buildings, while the baseline methods perform over-smooth in these complex structure areas. In Fig. 20, we show our estimated point cloud normals on a large-scale outdoor scene of the Semantic3D dataset. Since the ground-truth normals are not available, we map the estimated normal vectors to RGB colors for visualization and reconstruct the corresponding surface.

The above evaluation results demonstrate that our model trained with local patches on shape data can generalize well to real-world LiDAR data.

F. Complexity and Efficiency

We compare the normal estimation methods that use deep neural networks. As shown in Table V, we report the number of the learnable network parameters of each method, and the inference time on the PCPNet dataset using NVIDIA 2080 Ti GPU. Our method has the minimum network parameters and inference time. Compared with the SOTA method, MSEC-Net [19], our method has about 20% of its network parameters and runs about 5.5 times faster.



(a) (b) (c)

Fig. 20. Visualization of our estimated normals and reconstructed surface on a point cloud of the Semantic3D dataset. (a) The input LiDAR point cloud of the real-world outdoor scene. (b) The normal estimation results of our method. The estimated normal vectors are mapped to RGB colors for visualization, where different colors represent different normal directions [13]. (c) The reconstructed surface using estimated normals.

TABLE VIII
ABLATION STUDIES ON THE PCPNET DATASET. THE DISCUSSION IS PROVIDED IN THE TEXT.

	Ablation	Per-point Feature	Block $\mathcal{F}_1$	Block $\mathcal{F}_2$	Attention	Noise None	Noise Low	Noise Medium	Noise High	Density Stripe	Density Gradient	Average
(a)	PointNet-like		✓	✓	✓	4.63	8.91	16.11	21.09	5.31	5.14	10.20
	DGCNN-like		✓	✓	✓	3.83	8.43	15.78	20.96	4.62	4.36	9.66
(b)	w/o weight w	✓		✓	✓	3.71	8.50	16.05	21.46	4.61	4.36	9.78
	w/ other weight	✓		✓	✓	3.61	8.46	16.01	21.47	4.44	4.28	9.71
	w/o $\mathcal{F}_1$	✓		✓	✓	4.18	8.69	16.53	21.93	5.13	4.91	10.23
(c)	w/o $\mathcal{F}_2$	✓	✓		✓	3.69	8.50	16.05	21.41	4.67	4.55	9.81
	$\mathcal{F}_2 \rightarrow \mathcal{F}_1$	✓	✓		✓	3.59	8.48	15.78	21.03	4.64	4.23	9.62
(d)	w/ *softmax*-1	✓	✓	✓		3.39	8.41	15.62	20.97	**4.01**	3.96	9.39
	w/ *softmax*-2	✓	✓	✓		3.41	8.34	15.68	20.97	4.22	4.06	9.45
	w/ simp. concat	✓	✓	✓		3.73	8.33	15.78	21.01	4.57	4.36	9.63
	w/ simp. add	✓	✓	✓		3.57	8.35	15.69	20.99	4.20	4.03	9.47
	w/ add	✓	✓	✓		3.33	8.36	15.70	20.95	4.11	4.02	9.41
(e)	w/o d_{sin}	✓	✓	✓	✓	3.73	8.45	**15.60**	**20.82**	4.49	4.25	9.56
	w/o d_{euc}	✓	✓	✓	✓	3.66	8.37	16.05	21.33	4.38	4.31	9.68
	w/o $\mathcal{L}_{\mathbf{n}}^{p}$	✓	✓	✓	✓	3.67	8.36	15.76	21.05	4.51	4.31	9.61
	w/o $\mathcal{L}_{\tau}$	✓	✓	✓	✓	4.41	8.68	15.77	21.00	5.41	5.16	10.07
(f)	$N=500$	✓	✓	✓	✓	3.33	**8.26**	15.74	21.39	4.17	4.09	9.49
	$N=600$	✓	✓	✓	✓	3.39	8.31	15.66	21.11	4.17	4.02	9.44
	$N=700$	✓	✓	✓	✓	3.43	8.29	15.68	21.02	4.13	4.03	9.43
	$N=900$	✓	✓	✓	✓	3.45	8.29	15.69	20.94	4.39	**3.92**	9.45
	$N=1000$	✓	✓	✓	✓	3.38	8.37	15.64	20.87	4.18	4.03	9.41
	Ours	✓	✓	✓	✓	**3.32**	8.34	15.63	20.94	4.10	**3.92**	**9.38**

TABLE IX
NORMAL RMSE ON THE PCPNET DATASET USING VARIOUS PATCH SIZES.

AdaFit	HSurf-Net	CMG-Net	MSECNet	Ours (N=500)	Ours (N=700)	Ours (N=1000)
10.76	10.11	9.93	9.76	9.49	9.43	**9.41**

TABLE X
NORMAL RMSE ON THE PCPNET DATASET USING VARIOUS PATCH SIZES.

Size	AdaFit [14]	HSurf-Net [16]	CMG-Net [49]	MSECNet [19]	Ours
800	10.85	10.13	9.93	9.78	**9.38**
900	10.89	10.23	9.99	9.79	**9.36**
1000	10.96	10.39	10.04	9.80	**9.36**

G. Ablation Studies

We provide the ablation results in Table VIII (a)-(f), and each ablation experiment is discussed as follows.

(a) Per-point feature. The module in Eq. (5) extracts per-point features from point-wise and local points through a two-branched structure. If φ is removed, it degenerates into a PointNet-like structure [45]. If ϕ is removed, it degenerates into a DGCNN-like structure [65]. The results show that the combination of these two structures is essential.

(b) Block $\mathcal{F}_1$. (1) We do not use the learnable distance-based weight w in both $\mathcal{F}_1$ and $\mathcal{F}_2$. (2) We adopt another weighting technique used in [66]. (3) We replace each layer $\mathcal{P}$ in $\mathcal{F}_1$ with MLP, *i.e.*, without using $\mathcal{F}_1$. The worse results of these ablations validate the effectiveness of our newly designed layer and show its key role in the normal estimation pipeline.

(c) Block $\mathcal{F}_2$. We replace each layer $\mathcal{P}$ in $\mathcal{F}_2$ with MLP, *i.e.*, without using $\mathcal{F}_2$, or replace $\mathcal{F}_2$ with $\mathcal{F}_1$. The results demonstrate that this block is important for improving the algorithm's performance.

(d) Cross-scale compensation. (1) We use *softmax* in this module to generate attention weights for modulating individual feature channels (*softmax*-1) or point channels (*softmax*-2). (2) We provide the results by replacing the entire cross-scale compensation module with the simple operation of concatenation or addition of features. Different from the previous one, we also examine that we only replace $\eta(\cdot,\cdot)$ in Eq. (11) from concat to add. As we can see from the results, the performance of the algorithm decreases if *softmax* is used.

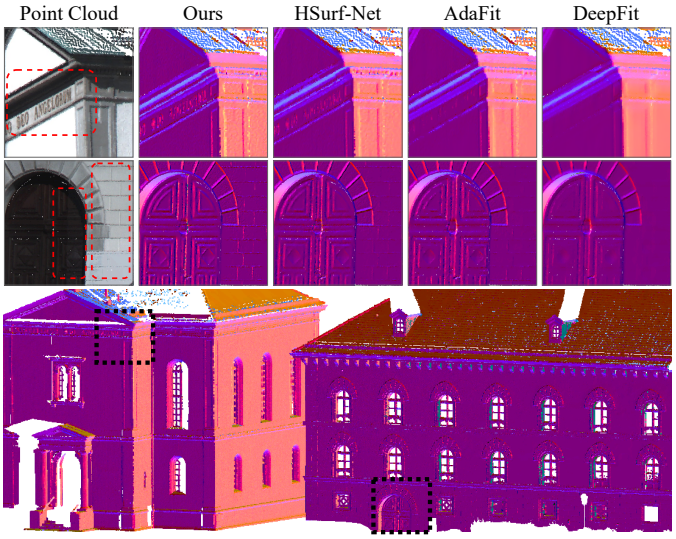


Fig. 21. Visual comparison of normal results on the Semantic3D dataset. We compare the local details of the building in the first two rows and show our estimated normals of the building in the third row. The point normal vectors are mapped to RGB colors.

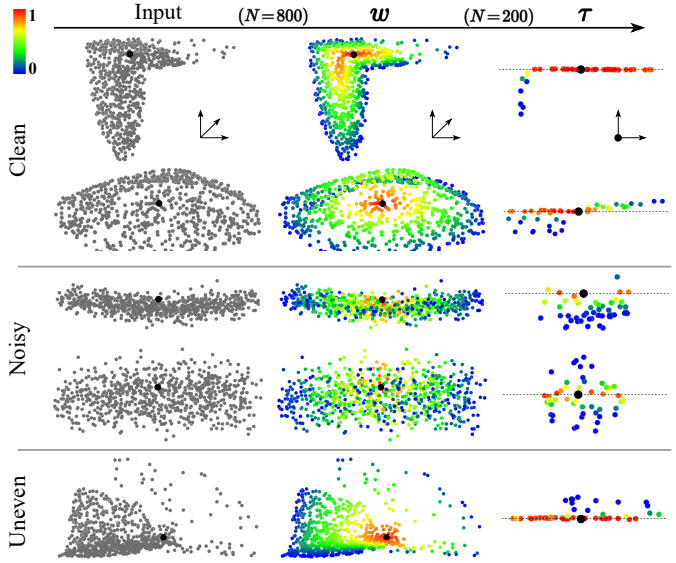


Fig. 22. Visualization of weights w and τ. It shows that the model's focus changes from central to coplanar with respect to the query point (black) at different stages. The perspective in the third column is changed for better visualization.

The simple feature concatenation and addition operations or replacing $\eta(\cdot,\cdot)$ are inadequate for realizing the cross-scale feature compensation.

(e) Loss. We do not calculate the distances d_{sin} or d_{euc} of $\mathcal{L}_{\mathbf{n}}$ in Eq. (13). We also alternately leave the losses $\mathcal{L}_{\mathbf{n}}^{p}$ and $\mathcal{L}_{\tau}$ in Eq.(15) unused. These ablations all lead to worse results.

(f) Input patch size. We train full models with a series of patch sizes $N = 500, 600, 700, 900, 1000$. A smaller size slightly degrades performance. A larger size brings no performance improvement but consumes more time and memory. To analyze the impact of input neighborhood size on algorithm performance, we provide more quantitative comparison results. In Table IX, the baseline methods are trained and tested with their default values, while our method is trained and tested with different patch sizes. In Table X, all methods are trained with their default values but tested with different patch sizes. Our method achieves excellent results even when training or testing with different patch sizes, while the baseline methods perform worse than our method under various input neighborhood sizes.

What does the model focus on? As shown in Fig. 22, we visualize the learned weights w in Eq. (7) and τ in Eq. (12). We can see that the focus of our model changes along the normal estimation pipeline. The model first focuses on points closer to the center during the feature aggregation, where features from large scales are transferred to small scales. Then, the model focuses on some neighboring points coplanar with the query during the final normal prediction.

VI. Conclusion

In this work, we analyze the effect of patch size on point cloud normal estimation, and propose a strategy that exploits the idea of patch feature fitting to approximate optimal features for normal estimation. We use multi-scale features from different patch sizes to build the feature-based polynomial, and apply cross-scale attention to compensate for the approximation error. The approximation strategy is implemented using an effective neural network, which aggregates features from multiple scales and achieves scale adaptation for varying local patches, to facilitate the geometric description around a point. We conduct thorough experiments to compare with baselines and validate the proposed modules. The main limitation of our method is that each point in the patch still needs a fixed neighborhood size to find its neighboring points to extract local features, which is also relatively time-consuming. Future work includes exploring more efficient feature extraction techniques and more application scenarios of our method.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (62402401), the Sichuan Provincial Natural Science Foundation of China (2025ZNSFSC1462) and the Fundamental Research Funds for the Central Universities.

References

[1] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics Symposium on Geometry Processing*, vol. 7, 2006. 1, 9

[2] J. F. Blinn, "Simulation of wrinkled surfaces," *ACM SIGGRAPH Computer Graphics*, vol. 12, no. 3, pp. 286–292, 1978. 1

[3] H. Gouraud, "Continuous shading of curved surfaces," *IEEE Transactions on Computers*, vol. 100, no. 6, pp. 623–629, 1971. 1

[4] B. T. Phong, "Illumination for computer generated pictures," *Communications of the ACM*, vol. 18, no. 6, pp. 311–317, 1975. 1

[5] H. Avron, A. Sharf, C. Greif, and D. Cohen-Or, "L1-sparse reconstruction of sharp point set surfaces," *ACM Transactions on Graphics*, vol. 29, no. 5, pp. 1–12, 2010. 1

[6] X. Lu, S. Wu, H. Chen, S.-K. Yeung, W. Chen, and M. Zwicker, "GPF: Gmm-inspired feature-preserving point set filtering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 8, pp. 2315–2326, 2017. 1

[7] X. Lu, S. Schaefer, J. Luo, L. Ma, and Y. He, "Low rank matrix approximation for 3D geometry filtering," *IEEE Transactions on Visualization and Computer Graphics*, 2020. 1, 9

[8] Z. Liu, Y. Zhao, S. Zhan, Y. Liu, R. Chen, and Y. He, "PCDNF: Revisiting learning-based point cloud denoising via joint normal filtering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 8, pp. 5419–5436, 2024. 1

[9] D. de Silva Edirimuni, X. Lu, G. Li, and A. Robles-Kelly, "Contrastive learning for joint normal estimation and point cloud filtering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 8, pp. 4527–4541, 2024. 1

[10] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, 1992, pp. 71–78. 1, 2, 3, 6, 7, 10

[11] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. J. Mitra, "PCPNet: learning local shape properties from raw point clouds," in *Computer Graphics Forum*, vol. 37, no. 2. Wiley Online Library, 2018, pp. 75–85. 1, 2, 4, 6, 7, 8, 9

[12] Y. Ben-Shabat and S. Gould, "DeepFit: 3D surface fitting via neural network weighted least squares," in *European Conference on Computer Vision*. Springer, 2020, pp. 20–34. 1, 2, 3, 4, 6, 8

[13] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, "Nesti-Net: Normal estimation for unstructured 3D point clouds using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 112–10 120. 1, 2, 6, 8, 12

[14] R. Zhu, Y. Liu, Z. Dong, Y. Wang, T. Jiang, W. Wang, and B. Yang, "AdaFit: Rethinking learning-based normal estimation on point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6118–6127. 1, 2, 3, 4, 6, 8, 12

[15] H. Zhou, H. Chen, Y. Zhang, M. Wei, H. Xie, J. Wang, T. Lu, J. Qin, and X.-P. Zhang, "Refine-Net: Normal refinement neural network for noisy point clouds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 946–963, 2022. 1, 2, 6, 8

[16] Q. Li, Y.-S. Liu, J.-S. Cheng, C. Wang, Y. Fang, and Z. Han, "HSurf-Net: Normal estimation for 3D point clouds by learning hyper surfaces," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35. Curran Associates, Inc., 2022, pp. 4218–4230. 1, 2, 6, 7, 8, 9, 12

[17] S. Li, J. Zhou, B. Ma, Y.-S. Liu, and Z. Han, "NeAF: Learning neural angle fields for point normal estimation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023. 1, 2, 4, 6, 8

[18] Q. Li, H. Feng, K. Shi, Y. Gao, Y. Fang, Y.-S. Liu, and Z. Han, "SHS-Net: Learning signed hyper surfaces for oriented normal estimation of point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2023, pp. 13 591–13 600. 1, 2, 5, 6, 7, 8, 9

[19] H. Xiu, X. Liu, W. Wang, K.-S. Kim, and M. Matsuoka, "MSECNet: Accurate and robust normal estimation for 3D point clouds by multi-scale edge conditioning," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 2535–2543. 1, 2, 6, 8, 11, 12

[20] K. Li, M. Zhao, H. Wu, D.-M. Yan, Z. Shen, F.-Y. Wang, and G. Xiong, "GraphFit: Learning multi-scale graph-convolutional representation for point cloud normal estimation," in *European Conference on Computer Vision*. Springer, 2022, pp. 651–667. 2, 3, 6, 8

[21] J. Zhang, J.-J. Cao, H.-R. Zhu, D.-M. Yan, and X.-P. Liu, "Geometry guided deep surface normal estimation," *Computer-Aided Design*, vol. 142, p. 103119, 2022. 2, 3, 6, 8

[22] Q. Li, H. Feng, K. Shi, Y. Gao, Y. Fang, Y.-S. Liu, and Z. Han, "Learning signed hyper surfaces for oriented point cloud normal estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 46, no. 12, pp. 9957–9974, 2024. 2, 5, 8

[23] F. Cazals and M. Pouget, "Estimating differential quantities using polynomial fitting of osculating jets," *Computer Aided Geometric Design*, vol. 22, no. 2, pp. 121–146, 2005. 2, 4, 6, 7

[24] H. Du, X. Yan, J. Wang, D. Xie, and S. Pu, "Rethinking the approximation error in 3D surface fitting for point cloud normal estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 3, 6, 8

[25] G. W. Stewart, "On the early history of the singular value decomposition," *SIAM Review*, vol. 35, no. 4, pp. 551–566, 1993. 2

[26] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Point set surfaces," in *Proceedings Visualization, 2001. VIS'01.* IEEE, 2001, pp. 21–29. 2

[27] C. Lange and K. Polthier, "Anisotropic smoothing of point sets," *Computer Aided Geometric Design*, vol. 22, no. 7, pp. 680–692, 2005. 2

[28] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or, "Consolidation of unorganized point clouds for surface reconstruction," *ACM Transactions on Graphics*, vol. 28, no. 5, pp. 1–7, 2009. 2

[29] N. J. Mitra and A. Nguyen, "Estimating surface normals in noisy point cloud data," in *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, 2003, pp. 322–328. 2

[30] N. Amenta and M. Bern, "Surface reconstruction by voronoi filtering," *Discrete & Computational Geometry*, vol. 22, no. 4, pp. 481–504, 1999. 2

[31] Q. Mérigot, M. Ovsjanikov, and L. J. Guibas, "Voronoi-based curvature and feature estimation from point clouds," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 6, pp. 743–756, 2010. 2

[32] T. K. Dey and S. Goswami, "Provable surface reconstruction from noisy samples," *Computational Geometry*, vol. 35, no. 1-2, pp. 124–141, 2006. 2

[33] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun, "Voronoi-based variational reconstruction of unoriented point sets," in *Symposium on Geometry Processing*, vol. 7, 2007, pp. 39–48. 2

[34] A. Boulch and R. Marlet, "Fast and robust normal estimation for point clouds with sharp features," in *Computer Graphics Forum*, vol. 31, no. 5. Wiley Online Library, 2012, pp. 1765–1774. 2

[35] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. Zhang, "Edge-aware point set resampling," *ACM Transactions on Graphics*, vol. 32, no. 1, pp. 1–12, 2013. 2

[36] D. Levin, "The approximation power of moving least-squares," *Mathematics of Computation*, vol. 67, no. 224, pp. 1517–1531, 1998. 2

[37] G. Guennebaud and M. Gross, "Algebraic point set surfaces," in *ACM SIGGRAPH 2007 papers*, 2007. 2

[38] S. Aroudj, P. Seemann, F. Langguth, S. Guthe, and M. Goesele, "Visibility-consistent thin surface reconstruction using multi-scale kernels," *ACM Transactions on Graphics*, vol. 36, no. 6, pp. 1–13, 2017. 2

[39] A. C. Öztireli, G. Guennebaud, and M. Gross, "Feature preserving point set surfaces based on non-linear kernel regression," in *Computer Graphics Forum*, vol. 28, no. 2. Wiley Online Library, 2009, pp. 493–501. 2

[40] R. Xu, Z. Dou, N. Wang, S. Xin, S. Chen, M. Jiang, X. Guo, W. Wang, and C. Tu, "Globally consistent normal orientation for point clouds by regularizing the winding-number field," *ACM Transactions on Graphics (TOG)*, 2023. 2, 7, 8

[41] A. Boulch and R. Marlet, "Deep learning for robust normal estimation in unstructured point clouds," in *Computer Graphics Forum*, vol. 35, no. 5. Wiley Online Library, 2016, pp. 281–290. 2

[42] R. Roveri, A. C. Öztireli, I. Pandele, and M. Gross, "PointProNets: Consolidation of point clouds with convolutional neural networks," in *Computer Graphics Forum*, vol. 37, no. 2. Wiley Online Library, 2018, pp. 87–99. 2

[43] D. Lu, X. Lu, Y. Sun, and J. Wang, "Deep feature-preserving normal estimation for point cloud filtering," *Computer-Aided Design*, vol. 125, p. 102860, 2020. 2

[44] J. Zhou, H. Huang, B. Liu, and X. Liu, "Normal estimation for 3D point clouds via local plane constraint and multi-scale selection," *Computer-Aided Design*, vol. 129, p. 102916, 2020. 2, 6, 8

[45] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660. 2, 4, 12

[46] T. Hashimoto and M. Saito, "Normal estimation for accurate 3D mesh reconstruction with point cloud model incorporating spatial structure." in *CVPR Workshops*, 2019, pp. 54–63. 2

[47] Q. Li, H. Feng, K. Shi, Y. Fang, Y.-S. Liu, and Z. Han, "Neural gradient learning and optimization for oriented point normal estimation," in *SIGGRAPH Asia 2023 Conference Papers*, December 2023. 2, 6, 8, 9

[48] Q. Li, H. Feng, K. Shi, Y. Gao, Y. Fang, Y.-S. Liu, and Z. Han, "NeuralGF: Unsupervised point normal estimation by learning neural gradient function," in *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023. 2, 6, 8

[49] Y. Wu, M. Zhao, K. Li, W. Quan, T. Yu, J. Yang, X. Jia, and D.-M. Yan, "CMG-Net: Robust normal estimation for point clouds via chamfer normal distance and multi-scale geometry," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 6, 2024, pp. 6171–6179. 2, 6, 8, 12

[50] J. E. Lenssen, C. Osendorfer, and J. Masci, "Deep iterative surface normal estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 247–11 256. 2, 6

[51] J. Cao, H. Zhu, Y. Bai, J. Zhou, J. Pan, and Z. Su, "Latent tangent space representation for normal estimation," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 1, pp. 921–929, 2021. 3, 6, 8

[52] J. Zhou, W. Jin, M. Wang, X. Liu, Z. Li, and Z. Liu, "Improvement of normal estimation for point clouds via simplifying surface fitting," *Computer-Aided Design*, p. 103533, 2023. 3, 6, 8

[53] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708. 4

[54] Y. Liu, B. Fan, G. Meng, J. Lu, S. Xiang, and C. Pan, "DensePoint: Learning densely contextual representation for efficient point cloud processing," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5239–5248. 4

[55] J. Zhang, J. Cao, X. Liu, H. Chen, B. Li, and L. Liu, "Multi-normal estimation via pair consistency voting," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 4, pp. 1693–1706, 2018. 6

[56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. 5

[57] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 259–16 268. 6

[58] B. Van Hoorick, P. Tendulkar, D. Suris, D. Park, S. Stent, and C. Vondrick, "Revealing occlusions with 4D neural fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3011–3021. 6

[59] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung, "SceneNN: A scene meshes dataset with annotations," in *2016 Fourth International Conference on 3D Vision*. IEEE, 2016, pp. 92–101. 9

[60] K. Lai, L. Bo, and D. Fox, "Unsupervised feature learning for 3D scene labeling," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3050–3057. 10

[61] G. Metzer, R. Hanocka, D. Zorin, R. Giryes, D. Panozzo, and D. Cohen-Or, "Orienting point clouds with dipole propagation," *ACM Transactions on Graphics*, vol. 40, no. 4, pp. 1–14, 2021. 10

[62] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics*, vol. 32, no. 3, pp. 1–13, 2013. 10

[63] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361. 10

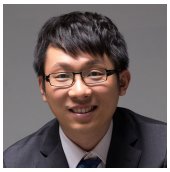
[64] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys, "SEMANTIC3D.NET: A new large-scale point cloud classification benchmark," in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. IV-1-W1, 2017, pp. 91–98. 10

[65] 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 Transactions On Graphics*, vol. 38, no. 5, pp. 1–12, 2019. 12

[66] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *European Conference on Computer Vision*. Springer, 2010, pp. 356–369. 12

Qing Li (Member, IEEE) received his Ph.D. degree from Xiamen University, Xiamen, China in 2021. From 2021 to 2023, he was a Postdoctoral Researcher at the School of Software, Tsinghua University, Beijing, China. He is currently an Associate Researcher at the School of Computing and Artificial Intelligence, Southwest Jiaotong University, China. His research interests include 3D computer vision and point cloud processing.

Huifang Feng received her Ph.D. degree from Xiamen University, Xiamen, China in 2023. She is currently an Associate Professor at the School of Computer and Software Engineering, Xihua University, Chengdu, China. Her research interests include 3D computer vision and point cloud processing.



Kanle Shi received his B.E. and Ph.D. degrees from Tsinghua University, China. He is now working for Kuaishou Technology, Beijing, China, mainly involved in research and development related to computer graphics. His research interests include computer graphics and computer vision.

Yue Gao (Senior Member, IEEE) received his B.E. degree from the Harbin Institute of Technology, Harbin, China, and his M.E. and Ph.D. degrees from Tsinghua University, China. He is currently an Associate Professor at the School of Software, Tsinghua University. His research interests include hypergraph computing, visual perception and augmentation, and data analysis and mining.

Yi Fang received his Ph.D. degree from Purdue University, West Lafayette in 2011. He is currently an Assistant Professor of Electrical and Computer Engineering, New York University Abu Dhabi. He also works at the Embodied AI and Robotics (AIR) Lab, New York University, USA. His research interests include 3D computer vision and machine learning.

Yu-Shen Liu (Member, IEEE) received his B.S. degree in mathematics from Jilin University in 2000, and his Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University in 2006. From 2006 to 2009, he was a Postdoctoral Researcher at Purdue University. He is currently an Associate Professor at the School of Software, Tsinghua University. His research interests include shape analysis, pattern recognition and machine learning.

Zhizhong Han received his Ph.D. degree from Northwestern Polytechnical University, China in 2017. He was a Postdoctoral Researcher with the Department of Computer Science, at the University of Maryland, College Park, USA. Currently, he is an Assistant Professor of Computer Science at Wayne State University, USA. His research interests include 3D computer vision, digital geometry processing, and artificial intelligence.