Learning Implicit Fields for Point Cloud Filtering

Jinxi Wang, Xuequan Lu, *Senior Member, IEEE*, Meili Wang, *Member, IEEE*, Fei Hou, and Ying He, *Member, IEEE*

***Abstract*—Since point clouds acquired by scanners inevitably contain noise, recovering a clean version from a noisy point cloud is essential for further 3D geometry processing applications. Several data-driven approaches have been recently introduced to overcome the drawbacks of traditional filtering algorithms, such as less robust preservation of sharp features and tedious tuning for multiple parameters. Most of these methods achieve filtering by directly regressing the position/displacement of each point, which may blur detailed features and is prone to uneven distribution. In this article, we propose a novel data-driven method that explores the implicit fields. Our assumption is that the given noisy points implicitly define a surface, and we attempt to obtain a point's movement direction and distance separately based on the predicted signed distance fields (SDFs). Taking a noisy point cloud as input, we first obtain a consistent alignment by incorporating the global points into local patches. We then feed them into an encoder-decoder structure and predict a 7D vector consisting of SDFs. Subsequently, the distance can be obtained directly from the first element in the vector, and the movement direction can be obtained by computing the gradient descent from the last six elements (i.e., six surrounding SDFs). We finally obtain the filtered results by moving each point with its predicted distance along its movement direction. Our method can produce feature-preserving results without requiring explicit normals. Experiments demonstrate that our method visually outperforms state-of-the-art methods and generally produces better quantitative results than position-based methods (both learning and non-learning).**

***Index Terms*—Point cloud filtering, implicit surface, deep learning, feature-preserving.**

I. Introduction

RESEARCHERS have made significant progress in point cloud filtering recently. Point cloud filtering methods can be naturally classified into optimization-based and learning-based types. CLOP [1] is one of the LOP-based methods without requiring normals. Although it achieves good smoothing effects, many detailed features become blurred. RIMLS [2] falls in the category of MLS-based methods that can better keep the detailed features of the noisy input. Subsequently, GPF [3] was proposed, which achieves good outcomes in preserving sharp features, especially for CAD models. However, this method requires a pre-processing step of downsampling, making it difficult to keep detailed features. The optimization methods often rely heavily on suitable parameter tuning and high-quality normals, and erroneous normal estimation can easily distort the filtered results.

Data-driven approaches have shown prominent effectiveness in dealing with point cloud tasks [4], [5], [6]. Regarding point cloud filtering, most learning-based methods [7], [8], [9] filter points based on direct regression of point positions or estimation of correction vectors. Specifically, most of them extract point features by local patches and decode them into filtered points. These methods can smooth point clouds with a slight feature-preserving effect. To keep sharp features, Zhang et al. [10] designed a projection loss function that incorporates ground truth normals. They obtain promising sharp feature-preserving outcomes. However, ground truth normals may not provide a suitable guide for the direction of point movement and can drive points toward sharp edges, resulting in gaps that affect the filtering and further reconstruction quality. In addition, when the input points are sparse, it can easily lead to shrinkage and outliers in the filtered results.

We observe that existing methods often employ repulsion forces in the loss function to achieve uniform point distribution, bringing about extra complexity. While SDFs can provide a continuous representation of surfaces, allowing for more accurate recovery of point cloud surfaces. In addition, we find that the gradient of the SDF at a point on the surface can provide the direction of movement towards the underlying surface. By utilizing this property, we can compute more accurate movement directions for noisy points during point cloud filtering, ensuring they are guided properly to move to the underlying surface. Fig. 1 shows that the ground truth normals may not accurately represent the true direction of movement of the noisy point towards the underlying surface. To learn the underlying clean surface without requiring any ground truth normal information, we propose a novel point cloud filtering method via learning the Signed Distance Fields (SDF) of the underlying surface. Different from the implicit learning in surface reconstruction [11], [12], [13], we attempt to explicitly derive the movement direction using merely the learned SDF values for point cloud filtering. Specifically, we first align the input local patches by computing a more stable

Received 8 October 2023; revised 7 June 2024; accepted 16 August 2024. Date of publication 27 August 2024; date of current version 1 August 2025. The work of Meili Wang was supported in part by the National Key Research and Development Program of China under Grant 2022YFD1300200, and in part by Shaanxi Province Key R&D Program under Grant 2022QFY11-03. The work of Fei Hou was supported in part by the National Key R&D Program of China under Grant 2023YFB3002901, in part by the Basic Research Project of ISCAS under Grant ISCAS-JCMS202303, and in part by the Major Research Project of ISCAS under Grant ISCAS-ZD-202401. Recommended for acceptance by P. Guerrero. *(Corresponding authors: Xuequan Lu and Meili Wang.)*

Jinxi Wang is with the College of Information Engineering, Northwest A&F University, Yangling 712100, China, and also with the Department of Computer Science and IT, La Trobe University, Melbourne, VIC 3086, Australia.

Meili Wang is with the College of Information Engineering, Northwest A&F University, Yangling 712100, China (e-mail: wml@nwsuaf.edu.cn).

Xuequan Lu is with the Department of Computer Science and IT, La Trobe University, Melbourne, VIC 3086, Australia (e-mail: B.Lu@latrobe.edu.au).

Fei Hou is with the Key Laboratory of System Software (Chinese Academy of Sciences), and the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China.

Ying He is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798.

Digital Object Identifier 10.1109/TVCG.2024.3450699

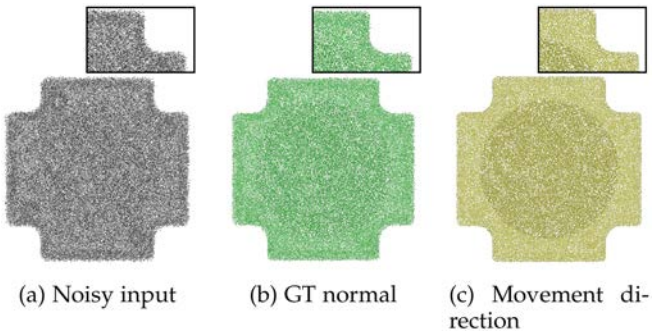

Fig. 1. Filtered point cloud by using ground truth normals and our derived SDF movement direction. The MSE errors are 6.878 and 4.056 ($\times 10^{-3}$), and the CD errors are 5.519 and 0.565 ($\times 10^{-5}$), respectively.

rotation matrix to facilitate the predictions of SDFs. Then, we employ a patch-based encoder to extract the local features of the input point patch and decode it to a 1×7 vector of SDFs of the central point. The first element is directly served as the moving distance. We then calculate gradient descent from the last six elements of the vector (i.e., six surrounding SDFs of this point) and convert it to the final moving direction by giving a negative sign. We finally get the filtered results by shifting each noisy point along its derived movement direction with its predicted distance. To the best of our knowledge, this is a pioneering study that models SDF distance for point cloud filtering and also seminal to explicitly derive an accurate direction with the learned SDF values to guide noisy points moving toward the underlying surface. Our method can obtain uniformly distributed outcomes and sharp feature-preserving effects without requiring either normals or repulsion force in designing our loss function. Comprehensive experiments demonstrate the effectiveness of this method and show it generates state-of-the-art filtering results.

The main technical contributions of this paper are:

- a novel point cloud filtering method via learning the Signed Distance Fields of the implicit surface, leading to the decomposition of a point's movement distance and direction, and
- a multi-task loss function that takes the distance loss, the direction loss, and the projection loss into account.

II. Related Work

A. Traditional Methods in Point Cloud Filtering

MLS-Based: Many filtering works [2], [14], [15], [16], [17] are based on the Moving Least Squares (MLS) since it has strong expressive power in designing effective underlying surface representations. Levin et al. [14] proved that MLS works better in interpolation and smoothing which can be used for surface approximation. Subsequently, Levin et al. [15] proposed an MLS-based projection strategy by surface interpolation where they first define a local reference domain and then construct an MLS approximation thus obtaining a smoothing surface. Alexa et al. [16] designed a smooth surface manifold by incorporating local maps into MLS approximation. Guennebaud et al. [17] presented a new Point Set Surface (PSS) definition based on MLS fitting with algebraic spheres. Öztireli et al. [2] combined the implicit MLS surfaces with robust statistics as a local kernel regression problem. These methods can yield good results in removing noise and maintaining geometric features. However, such local-based methods often do not take into account the distribution of filtered points, which makes the filtered results often unevenly distributed.

LOP-Based: The Locally Optimal Projection (LOP) operator allows promising surface approximation without requiring any explicit or implicit approximation space. Lipman et al. [18] introduced it for surface approximation from the point cloud and provides a second-order approximation to smooth surfaces. There are several variants based on this concept such as WLOP [19], CLOP [1], FLOP [20]. WLOP [19] added a weighted LOP operator so as to improve the quantity of the initial estimate of normals. Preiner et al. [1] proposed a novel continuous formulation of the WLOP operator based on a Gaussian mixture, namely CLOP, which greatly improves the efficiency of the original WLOP operation. Liao et al. [20] designed a bilateral weighting operator that considers spatial and geometric information for feature-preserving approximation. However, such methods assume that the underlying surface is smooth, which makes it difficult to maintain the sharp features of the model.

Nonlocal-Based: Researchers also investigate the nonlocal aspect of point cloud filtering. The nonlocal-based filtering methods include [21], [22], [23], [24], [25] etc., where they tend to update point positions by designing different similarity descriptions. Digne et al. [21] designed a height vector field for each point and updated the point via a low/high-frequency decomposition. Then Digne et al. [22] designed a local shape descriptor named Local Probing Field (LPF) for nonlocal shape analysis, which can handle shapes with mixed intrinsic dimensions. Sarkar et al. [23] divided the point cloud into patches and designed a local descriptor based on the dictionary learning framework. Besides, Zeng et al. [24] introduced a low-dimensional manifold model to surface patches in the point cloud and designed a patch-based manifold prior to seeking self-similar patches for noise removal. Wang et al. [25] designed the similarity of local patches through the RPCA decomposition and aggregated nonlocal similar information in the canonical space for the position update stage. These methods fully consider the nonlocal nature of the geometric model with feature-preserving effects. However, raw point clouds are often vast with the development of scanning devices, and nonlocal methods usually require multiple traversals of each point, which is time-consuming.

Others: In addition to the aforementioned methods, Lu et al. [3] proposed a robust feature-preserving filtering scheme called GPF that extended Gaussian Mixture Models (GMMs) to robustly reconstruct a high-quality point set with a sharp feature-preserving effect. Lu et al. [26] designed a low-rank matrix approximation algorithm for normal estimation which guided the movement of points in the position update stage. These methods mostly require reliable normals to ensure the feature-preserving effect of the outcomes. However, it is still

challenging to compute high-quality normals for severely corrupted point clouds, and poor normals may distort the filtered results.

B. Learning-Based Methods in Point Cloud Filtering

Early neural network architectures [4], [5], [6], [27] for directly processing point clouds have shown various applications in point cloud tasks like filtering, classification, segmentation, etc. PointNet [4] was first designed to tackle the problem of permutation in-variance of input point cloud and provides a unified architecture for classification and segmentation. Then Qi et al. [27] proposed an enhanced version of PointNet, named PointNet++, which incorporates multi-scale information of the point cloud. PointNet++ can better learn deep point features compared to the original version. PointCNN [5] is a generalization of typical CNNs to point clouds by learning $\mathcal{X}$-transformed features. DGCNN [6] is another designed neural network that can consume point clouds using a Dynamic Graph CNN. The developed module dynamically computes the graph at each layer of the network.

In terms of point cloud filtering, Yu et al. [7] first designed an edge-aware neural network called EC-Net for point cloud consolidation. The network is trained patch-wise but needs sharp-edge labeling manually. Roveri [8] proposed a generative neural network called PointProNet for point cloud consolidation via heightmap generation. Hermosilla et al. [28] proposed a point cloud filtering neural network without requiring any ground truth label in the training stage. Followed by PCPNet [29], PointCleanNet [9] was proposed for outlier removal and noise reduction from the corrupted point clouds. They first classified and discarded outlier samples, and then projected noisy points onto the underlying surfaces. Lu et al. [30] proposed to first estimate normals using height maps and then filter the point clouds using an edge-aware position update method. Zhang et al. [10] proposed a novel neural network for point cloud filtering by learning the displacements between noisy and clean points. Their approach can maintain sharp geometric features while removing noise. Francesca et al. [31] proposed a graph-convolutional layer-based method that can handle high noise levels and real LiDAR scans. Luo et al. [32] proposed an autoencoder-like network that learns the underlying manifold of the noisy point clouds. Chen et al. [33] designed a feature-aware recurrent point cloud denoising network to regress the displacement of noisy points. Luo et al. [34] developed a denoising neural network by performing a gradient ascent on the log-probability function. de Silva Edirimuni et al. [35] designed a novel loss function that utilized a dynamically adjusted ground truth at each training iteration.

C. Implicit Surface

The implicit surface representation has an attractive property of quickly identifying the relationship between a point and a surface by its coordinates. This nature has been well investigated in surface reconstruction by data-driven methods [11], [12], [13]. Ladicky et al. [11] designed a suitable feature vector and efficient oct-tree-based neural network, which could predict the SDF distance. However, this method needs ground truth normals to guide the predictions of the sign of the SDF. Chen et al. [13] designed an implicit decoder for representation learning and showed applications in generative shape modeling. Erler et al. [12] proposed a novel point-based learning framework that produces accurate surfaces directly from raw scans without normals by separating predicted features into a coarse global sign with a local absolute distance. In this work, instead of predicting a coarse direction (i.e., whether a given point is inside or outside the underlying surface), our approach introduces the implicit idea of predicting a precise direction that could guide the points to move onto the underlying surface.

III. Overview

Given a noisy point cloud, we first explain how we obtain a more stable rotation matrix for aligning the input patch by defining the local and global sampling strategy in Section IV-A. We then briefly introduce implicit surfaces and decompose the regression of noisy points into directions and distances in Section IV-B. Subsequently, we train an encoder-decoder network introduced in Section IV-C to learn the implicit surface of the input noisy point cloud by predicting the SDFs. Finally, we show how we utilize these predictions to calculate the moving distance as well as the movement direction, followed by introducing the loss functions in Section IV-D. Fig. 2 overviews the proposed approach.

IV. Method

A. Point Sampling

Local Sampling: We first denote the input noisy point cloud with M points as: $\widetilde{P} = \{\tilde{p}_i\}_{i=1}^{M}$, $\tilde{p}_i \in R^3$. The corresponding clean point cloud can be defined as: $P = \{p_i\}_{i=1}^{M}$, $p_i \in R^3$. We assume that each noisy point represents the location of a clean point after moving a certain distance in a specific direction. Our network aims to predict this direction and distance so the noisy point can be directly moved back to the underlying surface. For each noisy point, we define a local structure $\tilde{\mathrm{S}}_i^l$ as follows:

$$\tilde{\mathrm{S}}_i^l = \{\tilde{p}_j \mid \|\tilde{p}_j - \tilde{p}_i\| < \mu\} \tag{1}$$

where $\tilde{p}_i, \tilde{p}_j \in \widetilde{P}$. $\tilde{\mathrm{S}}_i^l$ is composed of neighboring points of $\tilde{p}_i$ within a fixed radius μ.

To project the noisy point onto the nearest surface, we generate the ground truth patch S_i^l by searching the same number of clean points within the fixed radius μ of the noisy point $\tilde{p}_i$. Our ground truth local patch can be defined as:

$$\mathrm{S}_i^l = \{p_j \mid \|p_j - \tilde{p}_i\| < \mu\} \tag{2}$$

Global Sampling: To overcome the distraction of arbitrary rotations of point clouds for training, researchers have explored several techniques for aligning point clouds [10], [29]. Here, we employ the QSTN [29] transformer to predict the rotation matrix for aligning the input patches. Since the local patch has very limited information about the global structure of the point cloud, it is difficult to obtain a robust and valid rotation matrix by feeding only the local points into QSTN. Inspired by Points2Surf [12],

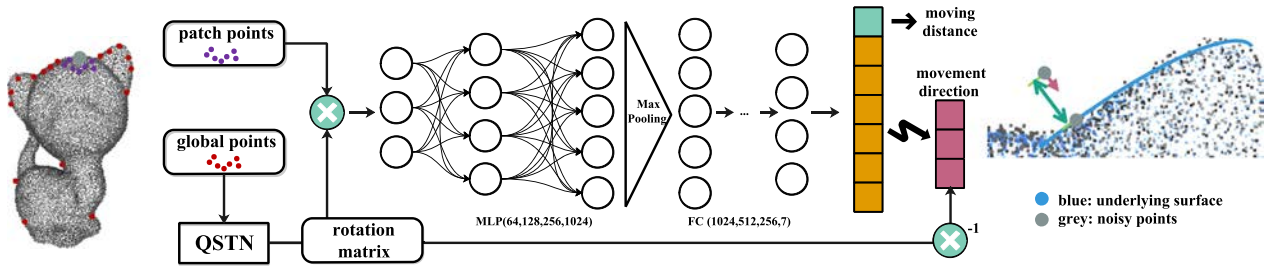

Fig. 2. Our network architecture. Given a noisy point (grey point), we generate local (purple points) and global (red points) patches from the noisy point cloud. We first feed the global part into the QSTN transform, thus obtaining a stable rotation matrix. Then, the aligned local patch is fed into the designed encoder-decoder architecture. The predicted vector consists of a 7-dimensional vector of SDFs, where the absolute value of the first element serves as the moving distance (green square), and the movement direction (pink squares) is obtained by computing the negative direction of the gradient descent from the last six predicted elements. Finally, the noisy point is moved back onto the underlying surface (blue points) by shifting it along the movement direction with the predicted distance.

we obtain a second set of sampled points for each noisy point, which considers global information. Specifically, for each point $\tilde{p}_i$, we define:

$$\sigma\left(\tilde{p}_j\right) = \frac{g\left(\tilde{p}_j\right)}{\sum_{\tilde{p}_k \in \tilde{P}} g\left(\tilde{p}_k\right)},$$
$$g\left(\tilde{p}_j\right) = \left[1 - 1.5\frac{\left\|\tilde{p}_j - \tilde{p}_i\right\|_2}{\max_{\tilde{p}_k \in \tilde{P}} \left\|\tilde{p}_k - \tilde{p}_i\right\|_2}\right]_{0.05}^{1} \quad (3)$$

where σ represents the sampling probability of $\tilde{p}_j$ in $\tilde{P}$ with the fixed $\tilde{p}_i$. g denotes a function which decreases with distance from $\tilde{p}_i$. Equation (3) can sample more points near $\tilde{p}_i$ and fewer points away from it, thus ensuring a closed object. Finally, the global sampling points according to σ can be formed as $\tilde{\mathrm{S}}_i^s$.

Patch Normalization: In our experiments, the point numbers of $\tilde{\mathrm{S}}_i^l$ and $\tilde{\mathrm{S}}_i^s$ are the same, and both are 500. Specifically, when $\tilde{\mathrm{S}}_i^l$ contains points less than 500, we generate 500 points by randomly repeating samples within the fixed radius μ. When the local sampling points are greater than 500, we generate the local patch by randomly choosing 500 points within that radius. As for S_i^l and $\tilde{\mathrm{S}}_i^l$, we first translate the patch points to the origin according to $\tilde{p}_i$, then we scale them to the uniform standard by the fixed radius μ. For the global sampling points $\tilde{\mathrm{S}}_i^s$, we centre them according to $\tilde{p}_i$.

B. Signed Distance Function

Given a noisy point $\tilde{p}_i$, we formulate the absolute distance between it and the implicit surface by Signed Distance Function (SDF) as:

$$d(\tilde{p}_i) = \min\left(|\tilde{p}_i - p_U|\right) \quad (4)$$

where p_U denotes the continuous point set on the underlying surface. It is easy to see that if $\tilde{p}_i$ is on the boundary set, then $d(\tilde{p}_i) = 0$. Thus we can formulate our filtering problem as:

$$P_{pre} = \left\{f(\tilde{p}_i) \in \mathbb{R}^3 \mid d(f) = 0\right\},$$
$$f\left(\tilde{p}_i\right) = \tilde{p}_i + m \cdot d\left(\tilde{p}_i\right) \quad (5)$$

The predicted point cloud P_{pre} can be seen as a set of sampling points from the underlying surface. Our goal is to predict the movement direction m and the distance $d(\tilde{p}_i)$ (i.e., SDF value).

Geometrically, the generated SDFs are given by a specific iso-surface (i.e., $d(\tilde{p}_i) = 0$). Thus we can obtain the movement direction by calculating the gradient of the SDF of $\tilde{p}_i$ with six surrounding SDFs and reverse it. The movement direction can be generated by:

$$\begin{aligned}\text{normalize}\ (\{&d(\tilde{p}_i + \{h,0,0\}) - d(\tilde{p}_i - \{h,0,0\}),\\ &d(\tilde{p}_i + \{0,h,0\}) - d(\tilde{p}_i - \{0,h,0\}),\\ &d(\tilde{p}_i + \{0,0,h\}) - d(\tilde{p}_i - \{0,0,h\})\}) \quad (6)\end{aligned}$$

where h denotes forward differentiation, and we set it to 0.0001. The function normalize is to normalize the vector into a unit-length vector.

C. Network Architecture

We first feed the global sampling points $\tilde{\mathrm{S}}_i^s$ into a Quaternion Spatial Transformer Network (QSTN) [29] to transform the input patch points into a canonical pose. The QSTN is simple and effective, and it is trained end-to-end in an unsupervised manner. This process is fine-tuned during training using a spatial transformer network [36]. The transformation is constrained to rotations by outputting quaternions. The predicted quaternions can then be converted to a 3×3 rotation matrix, aligning the input points to a standard pose. The network first uses three 1d convolutional layers to extract features from the global sampling points $\tilde{\mathrm{S}}_i^s$, each followed by batch normalization and ReLU activation which can be formulated as:

$$\mathbf{X} = \mathrm{ReLU}(\mathrm{BN}(\mathrm{Conv1d}(\mathbf{X}))) \quad (7)$$

After feature extraction, we use max pooling to aggregate the features. The features are then fed through 3 fully connected layers to predict the quaternion representing the rotation.

$$\mathbf{F} = \mathrm{ReLU}(\mathrm{BN}(\mathrm{Linear}(\mathbf{X}_{\mathrm{pooled}}))) \quad (8)$$

Then an identity quaternion is added to the predicted quaternion to handle identity transformations. Finally, the quaternion is converted to a 3×3 rotation matrix to transform the point cloud into a canonical pose [29].

$$\mathbf{q} = \mathbf{F} + [1,0,0,0]$$
$$\mathbf{R} = \mathrm{quat\_to\_rotmat}(\mathbf{q}) \quad (9)$$

We then use the rotation matrix $\mathbf{R}$ to normalize the orientation of the input patch.

$$\tilde{p}_i = \mathbf{R}\tilde{p}_i \tag{10}$$

Our encoder and decoder follow Pointfilter [10]. Specifically, our feature representation for each point is computed by feeding the input patch into MLPs. Each encoder layer uses batch normalization and ReLU. Then, the extracted feature representations can be aggregated by a max pooling layer. The decoder is implemented by 3 MLP layers which use batch normalization and ReLU, except for the last layer. We finally output a 1×7 vector, where each element of the vector represents a SDF value. The first element is directly used as the moving distance, and the last six elements of the vector are used to calculate the direction of movement.

D. Loss Function

Since our goal is to move the noisy point to the underlying surface, we design a powerful loss function that consists of 3 loss terms, to preserve sharp features without requiring any ground truth normal information and distribute more uniformly without using any repulsion force. The predicted vector and the corresponding ground truth vector can be defined as:

$$\begin{aligned} \hat{d}_i &= [\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4, \hat{x}_5, \hat{x}_6, \hat{x}_7] \\ d_i &= [x_1, x_2, x_3, x_4, x_5, x_6, x_7] \end{aligned} \tag{11}$$

where $\hat{x}_1$ represents the SDF value of $\tilde{p}_i$, and the last six elements represent the gradient fields used to calculate the movement direction of $\tilde{p}_i$.

Our first loss term L_{dis} directly calculates the L_2 distance between the predicted SDFs and ground truth, which is defined as:

$$L_{dis} = \left|\tanh(|\hat{d}_i|) - \tanh(|d_i|)\right|^2 \tag{12}$$

where L_{dis} can be seen as a constraint on distance.

Then we introduce a loss term to constrain the movement direction. The movement direction can be obtained by calculating the gradient of $\tilde{p}_i$. Specifically, the gradient can be calculated using the predicted SDFs around $\tilde{p}_i$. The movement direction can be generated by:

$$m_i = -\text{ normalize } (x_2 - x_3, x_4 - x_5, x_6 - x_7) \tag{13}$$

We experimentally found that using the L_2 regression perform slightly better than the cos similarity. Thus the loss can be formulated as:

$$L_{dir} = \|\hat{m}_i - m_i\|^2 \tag{14}$$

We then use our predictions to calculate the gradient of the SDF at $\tilde{p}_i$ located on the surface. Finally, we move the noisy point along the negative of gradient direction to get the denoised point. The predicted point can be formulated as follows:

$$\hat{p}_i = \tilde{p}_i + \hat{x}_1 * \hat{m}_i \tag{15}$$

Our last loss term is inspired by Pointfilter [10] which introduces normal similarity to better keep sharp features. Since we do not use any ground truth normal information, we propose to use our derived movement direction as an alternative. Thus, the third loss term is defined as:

$$L_{proj} = \frac{\sum_{p_j \in \mathrm{S}_i^l} \left|(\hat{p}_i - p_j) \cdot m_j^T\right| \cdot \phi\left(\|\hat{p}_i - p_j\|\right) \theta\left(\hat{m}_i, m_j\right)}{\sum_{p_j \in \mathrm{S}_i^l} \phi\left(\|\hat{p}_i - p_j\|\right) \theta\left(\hat{m}_i, m_j\right)}. \tag{16}$$

According to [37], $\theta(\hat{m}_i, m_j) = \exp(-\frac{1-\hat{m}_i^T m_j}{1-\cos(\sigma_n)})$, where $\hat{m}_i$ is our derived movement direction of the filtered point $\hat{p}_i$ and σ_n is the support angle. $\phi(\|\hat{p}_i - p_j\|)$ denote $\exp(-\frac{\|\hat{p}_i - \mathbf{p}_j\|^2}{\sigma_p^2})$, where $\sigma_p = 4\sqrt{\text{diag}/\text{t}}$ and diag denotes the length of the diagonal of the bounding box of the patch $\tilde{\mathrm{S}}_i^l$ and $t = |\tilde{\mathrm{S}}_i^l|$.

The final loss function can be therefore formulated as a sum of these three loss terms:

$$L_{all} = L_{dis} + L_{dir} + L_{proj} \tag{17}$$

V. Results

A. Experiment Setup

The experiments were conducted on a PC configured with Intel Xeon 8160T @ 2.1GHz and NVIDIA TITAN RTX. Our network is implemented by using PyTorch (CUDA 10.2). We sampled 8000 points for each point cloud and generated 500 points as a local patch for each point during the training stage. We train the network for 50 epochs, with a batch size of 64.

B. Dataset

Train Set: We collect 22 high-quality meshes containing both CAD and non-CAD models. We first sample the vertices of these meshes to 100K points using Furthest Point Sampling (FPS) and scale them uniformly. We then add Gaussian noise to each point cloud, and finally, we obtain 132 (22*6) training point clouds with noise levels of 0%, 0.25%, 0.5%, 1.0%, 1.5%, and 2.5%. We compute the ground truth SDF of each point by using the trimesh library. Since the generation process is time-consuming, we divide the computation into batches and obtain the ground truth movement direction at this stage by computing the surrounding SDFs for each point.

Test Set: We perform our method on three datasets. The first one comprises raw scan data and synthetic data of different sampling levels, with a total of 31 point clouds. Among them, there are 8 raw scan point clouds, 11 synthetic models corrupted with 0.5% noise, and 6 synthetic models with 1.0% and 1.5% noise. In addition, we perform experiments on synthetic data of PCN [9], which includes 26 models of five noise levels. we also compare all learning-based methods on the real scanned Kinect_v1 and Kinect_v2 [38].

C. Compared Methods

We select eight state-of-the-art point cloud filtering algorithms for qualitative and quantitative comparisons. There are three optimization-based methods: a position-based method of CLOP [19], two normal-based methods of GPF [3] and RIMLS [2]; five learning-based methods: TotalDenoising [28], PointCleanNet [9], Pointfilter [10], ScoreDenoise [34], and IterativePFN [35]. For fair comparisons, we employ the following rules: (a) For all methods, we first normalize the noisy input.

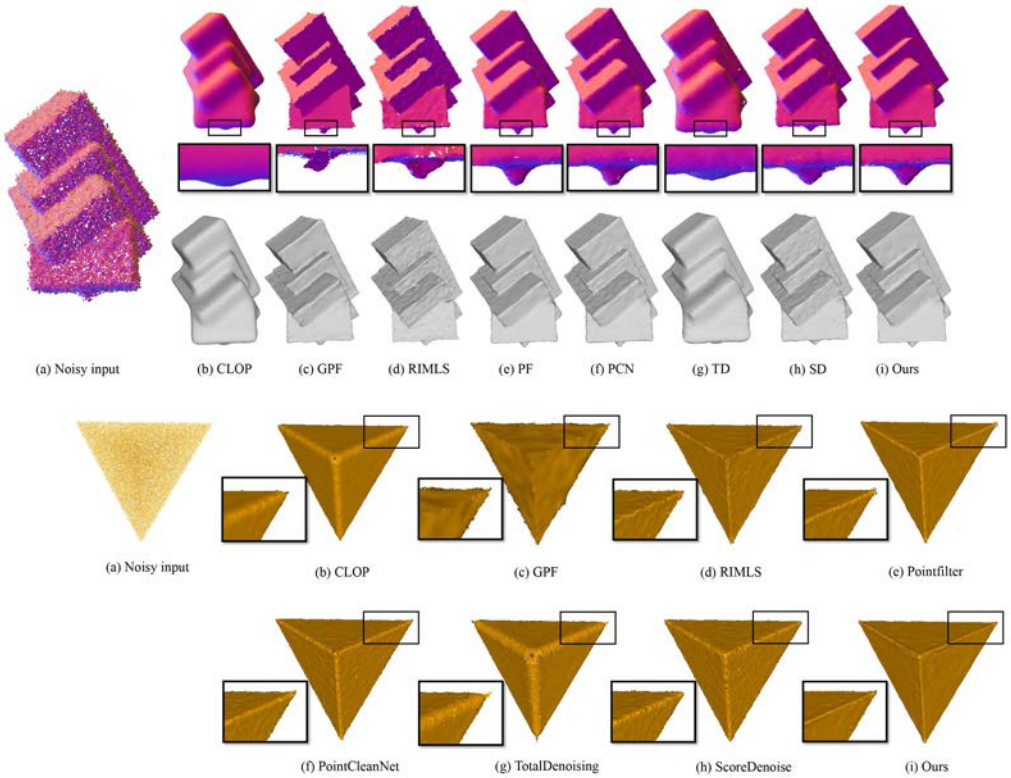

Fig. 3. Filtered results on models corrupted with 0.5% synthetic noise. From top to bottom: Boxunion, and Tetrahedron.

(b) For methods that rely upon normals, we first utilize PCA to calculate the initial normals which are then filtered by Bilateral Normal Filter [37] to ensure a more reliable input of normal information. (c) All the compared deep learning methods are re-trained on our train set with their suggested hyperparameters. (d) For the optimization-based methods, we adjust the main parameters of each method as well as we could to produce the best visual effect. (e) For learning-based methods, we choose to iterate 6 times for TotalDenoising, 3 times for PointCleanNet and Pointfilter, and 1 for ScoreDenoise.

D. Visual Comparison

For visual comparison purpose, we utilize EAR [37] for up-sampling to provide a similar number of points for point set rendering. In addition, we adopt the same parameters for the surface reconstruction of a given model for all methods. *Point clouds corrupted with synthetic noise.* Fig. 3 shows filtered results on point clouds corrupted by synthetic noise. The bottom of the figure shows up-sampling point rendering results and surface reconstruction of the model Boxunion with 0.5% noise. We can see that our approach outperforms other methods in terms of maintaining the geometric feature of the small triangle. The corresponding surface reconstruction also demonstrates the high quality of our filtered outcome. Besides, the figure also shows other up-sampling results on a CAD model corrupted with 0.5% noise. It is easy to see from the figure that our method can best maintain sharp features. The zoom-in box also demonstrates that we can obtain a smoother effect than other approaches. Fig. 4 shows more filtered results over all methods on models corrupted with synthetic noise. The first row shows filtered results on Fandisk corrupted with 0.5% synthetic noise. From the zoom-in box, it is easy to see that our method can best maintain sharp features. The second model shown in the figure presents point set rendering results on Buffle corrupted by 1.0% noise. From the back of the model, we can see that our method obtains a smoother outcome with feature-preserving effects. The model Julius showed at the bottom of Fig. 4 also indicates that our method preserves the detailed features of its chin best, while other methods appear to blur them.

From all the compared approaches, we observe that CLOP can smooth the noisy input well, but keeping geometric features is still challenging since it does not require any normals. GPF can keep the sharp features of the model but still has difficulties in keeping detailed geometric features. We believe this is due to the preprocessing step of downsampling, which may lead to the loss of some detailed features of the original model. RIMLS can obtain promising filtering results, but this method tends to distribute the model unevenly. TotalDenoising can smooth the model but cannot preserve features since it is unsupervised. PointCleanNet and ScoreDenoise can handle models with lower levels of noise. However, it is still challenging to maintain the smoothing effect when dealing with models of higher noise. Pointfilter can keep the detailed features and sharp features of the model, but when the point sampling is sparse, this method cannot output reliable filtered results.

Point Clouds Corrupted With Real Scan Noise: We also evaluate our approach with seven other state-of-the-art methods on point clouds corrupted with real scan noise. Fig. 5 shows filtered results on two raw scanned models. The bottom of the

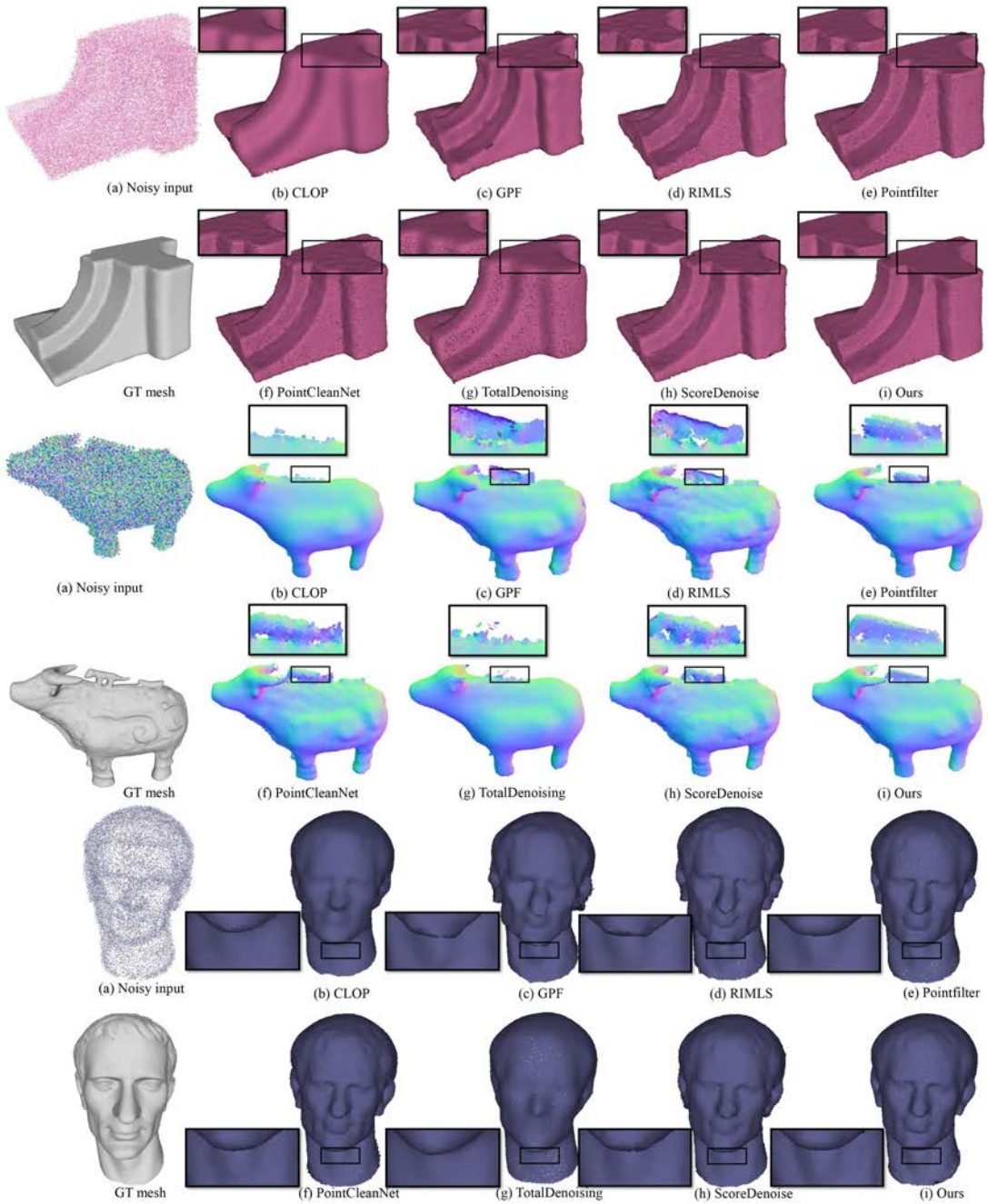

Fig. 4. Filtered results on models corrupted with synthetic noise. From top to bottom: Fandisk (corrupted with 0.5% noise), Buffle (corrupted with 1.0% noise), and Julius (corrupted with 1.0% noise).

figure shows the filtered results on the model David. We can see that our method preserves detailed features better during smoothing. Some optimization-based methods seem to blur the detailed characteristics of the model's hair. Most learning-based methods can keep features better but still contain certain noise, as seen in the zoom-in window. As shown in the zoom-in window of the second model, it is demonstrated again that our method can smooth the model better than other approaches while preserving features. Fig. 6 shows filtered outcomes over two other raw scanned models. The first row shows results on a model named Boy. As seen from the enlarged box, our method smooths the model better with feature-preserving effects. The filtered results produced by other methods still involve some noise. We can see from the enlargement box of the bottom row that our method preserves the circle part of the model Patrick Star better than other approaches.

E. Quantitative Comparison

Error Metrics: We follow Pointfilter [10] to perform quantitative comparisons of all compared methods on two error metrics, namely Chamfer Distance (CD) and Mean Square Error (MSE).

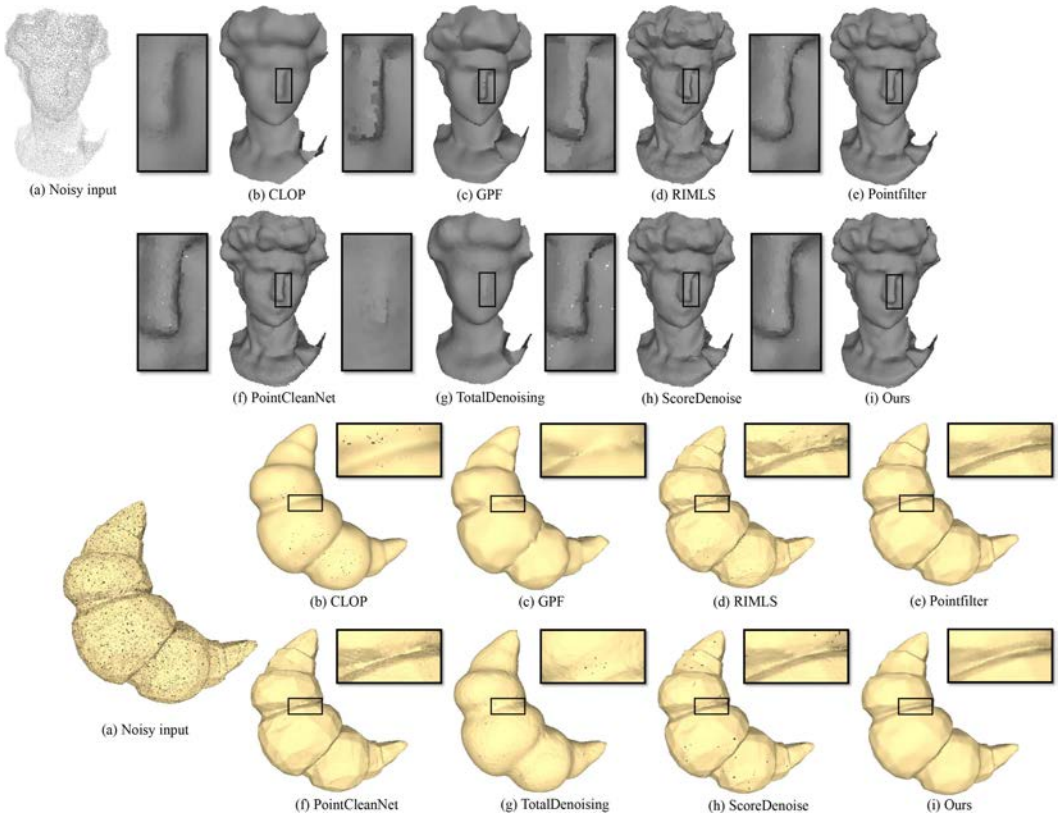

Fig. 5. Filtered results on models corrupted with raw scanned noise. From top to bottom: David, and Croissant.

Given a clean point cloud $S_c = \{x_i\}_{i=1}^{|S_c|}$, the corresponding predicted point cloud could be depicted as $S_p = \{y_i\}_{i=1}^{|S_p|}$. The point numbers of $|S_c|$ and $|S_p|$ may vary due to different filtering algorithms.

a) *Chamfer Distance:*

$$e_{\text{CD}}(S_c, S_p) = \frac{1}{|S_c|}\sum_{x\in S_c}\min_{y\in S_p}\|x-y\|_2^2 + \frac{1}{|S_p|}\sum_{y\in S_p}\min_{x\in S_c}\|y-x\|_2^2, \quad (18)$$

b) *Mean Square Error:*

$$e_{\text{MSE}}(S_c, S_p) = \frac{1}{|S_c|}\frac{1}{|N(y)|}\sum_{x\in S_c}\sum_{y\in N(y)}\|x-y\|_2^2, \quad (19)$$

where $N(y)$ denotes the nearest neighbors in S_c for point y in S_p. Similar to [10], we set $|N(y)| = 10$, which means we search 10 nearest neighbors for each point y in the predicted point set S_p.

Errors: For our test set, real scan dataset Kinect_v1 and Kinect_v2 [38], we evaluate the filtered results using the two metrics quantitatively as described above. Table I shows the average quantitative comparisons of all methods over synthetic point clouds in our test set. Consistent with the visual comparison, our method outperforms all traditional methods as well as deep learning methods. Table II shows the average quantitative comparisons of all deep learning class methods on the real scan dataset Kinect_v1 and Kinect_v2 [38]. We see from the table that

TABLE I
QUANTITATIVE EVALUATION OF ALL METHODS ON OUR TEST SET

Methods	MSE	CD
CLOP [19]	6.123	5.101
GPF [3]	7.227	6.337
RIMLS [2]	5.945	2.993
TotalDenoising [28]	6.300	4.640
PointCleanNet [9]	5.468	1.805
Pointfilter [10]	5.406	1.834
ScoreDenoise [34]	5.559	1.791
Ours	**5.396**	**1.758**

Metrics used are Mean Square Error ($\times 10^{-3}$) and Chamfer Distance ($\times 10^{-5}$).

TABLE II
QUANTITATIVE EVALUATION OF LEARNING-BASED METHODS ON REAL-SCANNED KINECT_V1, AND KINECT_V2 [38]

Methods	Kinect_v1		Kinect_v2	
	MSE	CD	MSE	CD
TotalDenoising [28]	5.915	4.678	8.780	5.502
PointCleanNet [9]	5.939	2.692	9.121	4.237
Pointfilter [10]	5.884	2.631	8.987	**3.934**
ScoreDenoise [34]	5.834	2.611	8.911	4.015
IterativePFN [35]	**5.750**	2.661	**8.588**	4.000
Ours	5.874	**2.595**	9.011	3.978

Metrics used are Mean Square Error ($\times 10^{-3}$) and Chamfer Distance ($\times 10^{-5}$).

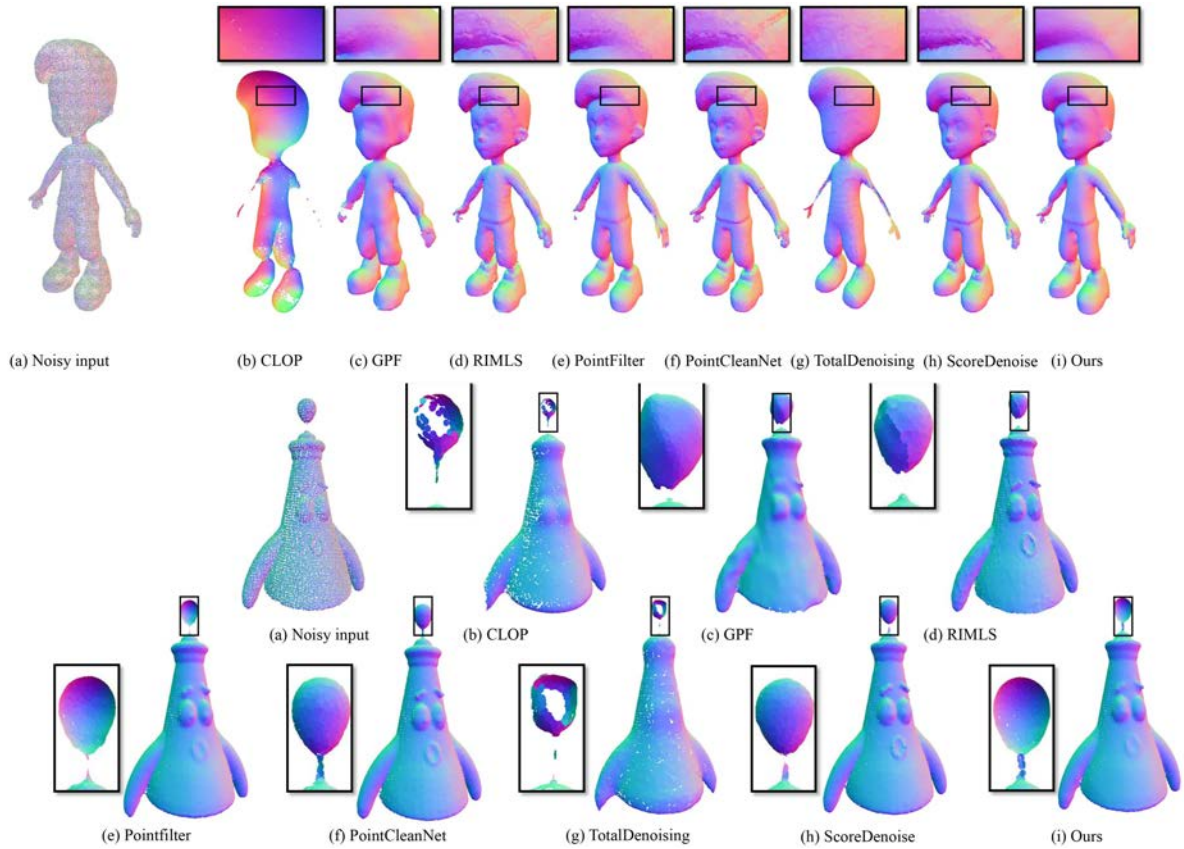

Fig. 6. Filtered results on models corrupted with raw scanned noise. From top to bottom: Boy and Patrick Star.

TABLE III
QUANTITATIVE EVALUATION OF LEARNING-BASED METHODS ON PCN [9]

Methods	PCN_3K						PCN_50K					
	0.5%_noise		1.5%_noise		2.5%_noise		0.5%_noise		1.5%_noise		2.5%_noise	
	CD	P2M	CD	P2M	CD	P2M	CD	P2M	CD	P2M	CD	P2M
Pointfilter [10]	14.312	5.025	43.585	18.209	96.789	44.660	2.210	0.729	3.916	1.518	**7.659**	**3.249**
ScoreDenoise [34]	26.859	11.384	57.464	27.257	121.087	59.650	1.912	0.610	4.100	1.560	11.082	4.885
PointCleanNet [9]	13.414	4.715	50.618	22.585	107.465	51.061	1.864	0.545	3.487	1.246	14.971	6.584
TotalDenoising [28]	14.310	4.854	**26.446**	**10.294**	84.670	38.876	6.000	2.559	4.816	1.870	12.970	5.677
IterativePFN [35]	39.580	18.196	35.846	16.024	**36.381**	**15.719**	**1.606**	**0.273**	**3.403**	**0.941**	14.681	5.651
Ours	**12.922**	**4.516**	26.887	11.516	59.067	27.576	1.768	0.535	3.485	1.436	10.577	4.565

Metrics used are Chamfer Distance ($\times 10^{-5}$), and P2M ($\times 10^{-4}$).

our method induces the lowest Chamfer Distance on Kinect_v1 dataset. Notice there might be differences between the compared methods and their papers since we retrained all learning-based methods on our train set. Table III shows the average quantitative comparisons of all deep learning methods on the PCN [9] dataset with different sampling and noise levels. We highlight the top methods with the lowest error in bold and underline the 2nd best. From the table, we observe that our method outperforms Pointfilter in most cases. Our method achieves the lowest error at a 0.5% noise level with sparse point density. Notably, our method outperforms the recent IterativePFN [35] that involves more complex graph convolution layers, in six cases, i.e., the 0.5% and 1.5% noise levels on the 3K resolution and 2.5% noise level on the 50K resolution. In addition, we also visualize the Chamfer Distance (CD) on some filtered models corrupted by synthetic noise in Fig. 7.

TABLE IV
RUNTIME (SECONDS) FOR COMPARED METHODS IN THE TEST STAGE

Methods (points)	TD [28]	PCN [9]	PF [10]	SD [34]	Ours
David (37k)	22.0	252.8	356.9	11.0	100.8
Patrick(74k)	91.0	594.3	303.0	33.5	147.7
Croissant(287k)	2502.1	3000	1296.4	770.1	1847.8
Boxunion(100k)	61.7	631.8	116.8	60.2	354.8
Fandisk(100k)	69.6	694.3	488.1	60.5	271.7
Tetrahedron(100k)	62.8	511	660.5	60.9	1732.3
Buffle(58k)	33.8	361.9	282.5	23.0	103.4
Julius(36k)	16.1	184.5	233.4	10.2	50.0

All methods were run on the same computer.

Runtime: In addition to the above experiments, we record the running time of each learning-based method. Table IV gives the runtime of TotalDenoising (TD), PointCleanNet (PCN), Pointfilter (PF), ScoreDenoise (SD), and our method. It is evident

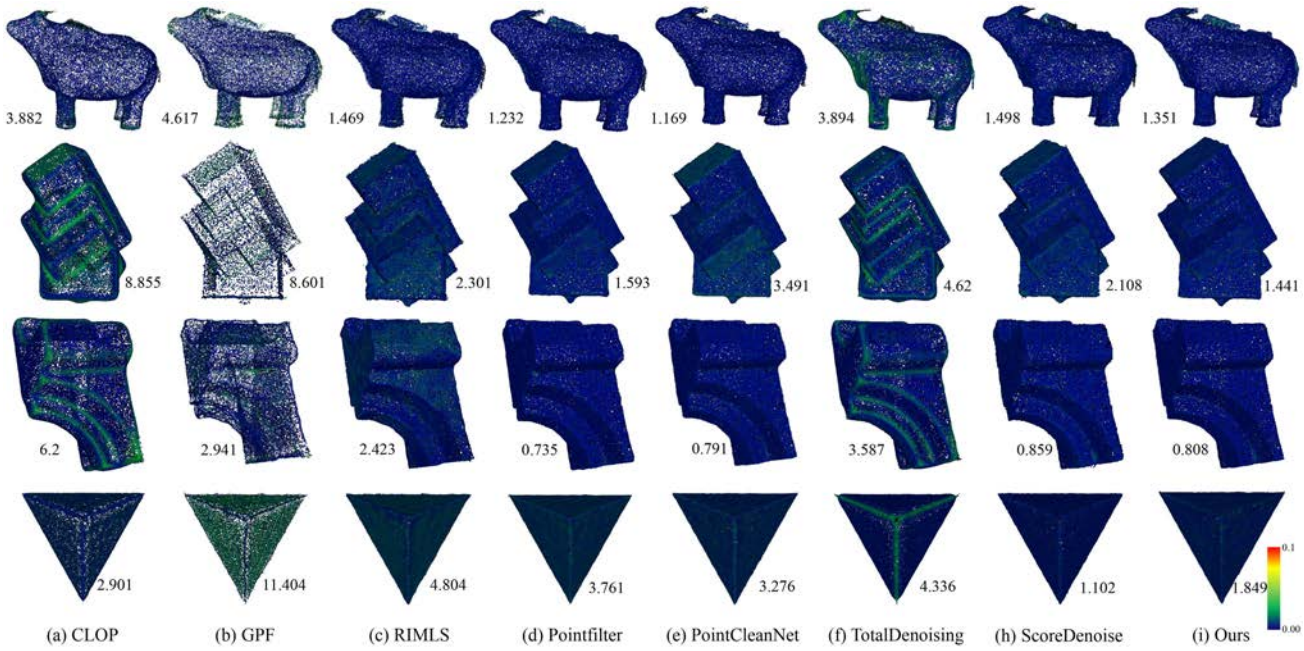

Fig. 7. Visualization in chamfer distance (CD). The overall errors for different methods over Buffle (1.0% noise), Boxunion (0.5% noise), Fandisk (0.5% noise), and Tetrahedron (0.5% noise) are shown.

TABLE V
FLOPS AND PARAMS (PARAMETERS) OF DIFFERENT MODELS

Methods	FLOPs(G)	Params (M)
ScoreDenoise [34]	0.3629491	0.187291
TotalDenoising [28]	0.0000795	0.03584
Pointfilter [10]	0.0872735	1.360899
PointCleanNet [9]	0.376156	31.392863
Ours	0.048054	2.163723

TABLE VI
QUANTITATIVE EVALUATION OF DEEP LEARNING-BASED METHODS ON THE SPARSE SAMPLING CUBE CORRUPTED WITH 0.5% SYNTHETIC NOISE

Methods	MSE	CD
TotalDenoising [28]	**5.968**	9.781
PointCleanNet [9]	6.327	8.413
Pointfilter [10]	6.297	8.871
ScoreDenoise [34]	6.106	<u>8.25</u>
Ours	<u>6.059</u>	**7.947**

Metrics used are Mean Square Error ($\times 10^{-3}$) and Chamfer Distance ($\times 10^{-5}$).

from the table that despite our method involving QSTN, it still exceeds Pointfilter in terms of running time for certain models. We also conducted an experiment to validate the computational complexity as well as model weights. Table V shows FLOPs and parameters over compared methods. From the table, we see that TotalDenoising has the smallest FLOPs and parameters. Since our method calculates 7-dimensional vector, it has a slightly larger parameter count than Pointfilter.

Sparse Sampling: To show the performance of handling sparse sampling point clouds, we conduct experiments on a cube model sampled with 6K points of all learning-based methods. The quantitative evaluation is shown in Table VI. Our method outperforms most learning-based methods. In addition, our method is superior to Pointfilter in handling sparsely sampled point clouds.

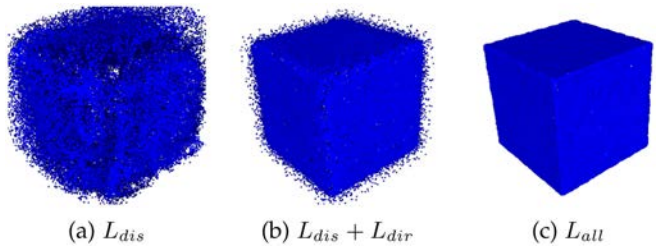

Fig. 8. Filtered results with different loss functions.

F. Ablation Study

Loss: To show the effectiveness of our loss function, we conduct experiments on different variants of losses. Specifically, we train L_{dis}, $L_{dis} + L_{dir}$, and L_{all}, respectively, with all other training configurations being the same. Fig. 8 shows the filtered results under different loss functions. It is easy to see that L_{dis} and L_{dir} which do not consider the geometric features of the model cannot remove noise well. Our complete loss function produces the most decent results in terms of noise removal as well as feature preservation.

Movement Direction: We also conduct experiments to directly predict the direction of movement. Table VII shows the results of the experiments under PCN [9] dataset. We can see that the lowest error is still obtained by predicting the 7-dimensional SDF values without utilizing any normal information.

TABLE VII
QUANTITATIVE EVALUATION OF DIRECTLY PREDICTING MOVEMENT DIRECTION VERSUS OUR PREDICTION OF 7 SDF VECTORS ON THE PCN [9]

Methods	MSE	CD
Direct Prediction	4.616	2.229
Ours	**4.528**	**1.768**

Metrics used are Mean Square Error ($\times 10^{-3}$) and Chamfer Distance ($\times 10^{-5}$).

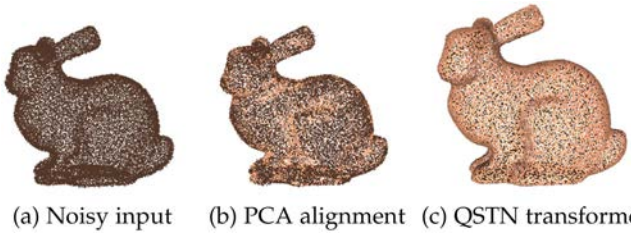

Fig. 9. Filtered results on Bunny (corrupted with 0.5% noise) under PCA alignment and QSTN transformer.

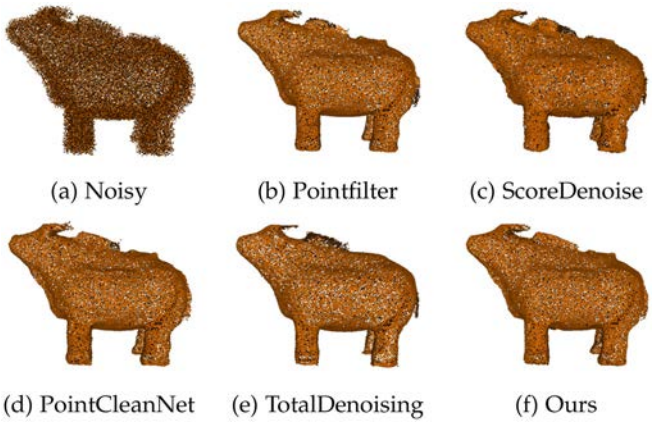

Fig. 10. Filtered results for a point cloud corrupted with 1.5% synthetic noise.

PCA versus QSTN: Alternatively, the input patches can also be aligned by PCA alignment [10]. We train our model using PCA alignment and QSTN transformer separately under the same training configuration. Fig. 9 shows filtered results under these two alignments. It is easy to see that the QSTN transformer provides a more consistent alignment, which facilitates better-filtered results.

High-Level Noise: Fig. 10 shows the filtered results of different methods at 1.5% noise. For this high-level noise, our approach yields the cleanest points with feature-preserving effects. The filtered results of other methods still retain some noise.

Point Distribution: Fig. 11 shows the filtered results of our method and Pointfilter [10]. From the zoom-in box, we can see that our method produces a more uniform point distribution. Pointfilter incorporates ground truth normals in the training stage to obtain a better feature-preserving effect. However, the ground truth normals may cause points to gather around the sharp edges. To solve this problem, they add a repulsive force to the loss function, similar to GPF [3]. By contrast, our method does not require any ground truth normal information or introduce any repulsive force into the loss function.

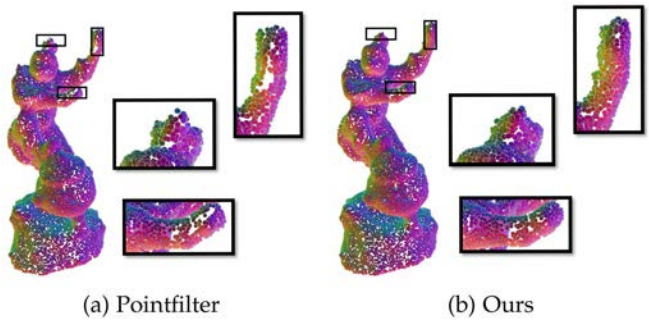

Fig. 11. Filtered results of pointfilter and ours on raw scanned model monkey.

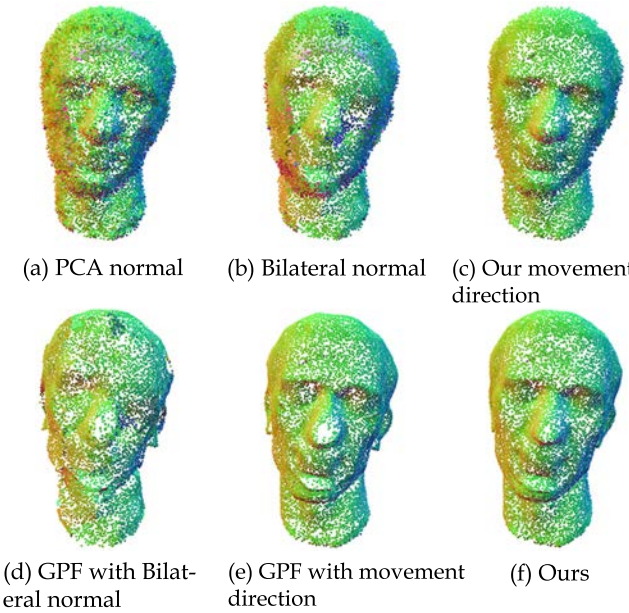

Fig. 12. Filtered results of GPF under Bilateral normal filter and movement direction.

Bilateral Normal versus Our Movement Direction: Filtering methods that require high-quality normals as input often fail under high noise conditions. We also confirmed this in our experiments. Nevertheless, we observed that our predicted movement direction under high noise could provide guidance for the normal-based approaches. Specifically, we perform experiments on a 1.0% synthetic noise-corrupted point cloud and compare with the GPF method [3]. The input normals are obtained by two techniques. We first obtain the initial normals by PCA and filter them using the Bilateral Normal Filter method [37], which are then used as the input normals. Alternatively, we use our movement direction directly as a solution. Fig. 12 shows the filtered results of both techniques. It is easy to see that our prediction of movement direction is more robust under high noise levels and thus can facilitate the GPF to obtain a more promising filtered result. In addition, we can see that our method yields a more uniform point distribution, although we do not use any repulsive force which is a part of GPF.

VI. Conclusion

We have proposed a novel data-driven method for feature-preserving point cloud filtering, without requiring either normal information or repulsion force. By learning the implicit surface of a given noisy point, we predict a 7D vector, where each element of the vector represents a distance scalar. The first element is considered as the moving distance. The direction of movement is then obtained by computing the gradient of the predicted point using the last six elements (i.e., surrounding SDFs of this point). Finally, the filtered point can be obtained directly by moving each noisy point along the corresponding predicted direction with the corresponding predicted distance. Extensive experiments demonstrate our method can produce feature-preserving filtering results with uniform point distribution, and show that it outperforms traditional methods and generally surpasses state-of-the-art learning-based methods.

References

[1] R. Preiner, O. Mattausch, M. Arikan, R. Pajarola, and M. Wimmer, "Continuous projection for fast L1 reconstruction," *ACM Trans. Graph.*, vol. 33, pp. 1–13, 2014.

[2] A. C. Öztireli, G. Guennebaud, and M. H. Gross, "Feature preserving point set surfaces based on non-linear kernel regression," *Comput. Graph. Forum*, vol. 28, pp. 493–501, 2009.

[3] X. Lu, S. Wu, H. Chen, S.-K. Yeung, W. Chen, and M. Zwicker, "GPF: GMM-inspired feature-preserving point set filtering," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 8, pp. 2315–2326, Aug. 2018.

[4] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.

[5] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-transformed points," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 828–838.

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

[7] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "EC-Net: An edge-aware point set consolidation network," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 386–402.

[8] R. Roveri, A. C. Öztireli, I. Pandele, and M. Gross, "PointProNets: Consolidation of point clouds with convolutional neural networks," *Comput. Graph. Forum*, vol. 37, no. 2, pp. 87–99, 2018.

[9] M.-J. Rakotosaona, V. La Barbera, P. Guerrero, N. J. Mitra, and M. Ovsjanikov, "PointCleanNet: Learning to denoise and remove outliers from dense point clouds," *Comput. Graph. Forum*, vol. 39, no. 1, pp. 185–203, 2020.

[10] D. Zhang, X. Lu, H. Qin, and Y. He, "PointFilter: Point cloud filtering via encoder-decoder modeling," *IEEE Trans. Vis. Comput. Graphics*, vol. 27, no. 3, pp. 2015–2027, Mar. 2021.

[11] L. Ladicky, O. Saurer, S. Jeong, F. Maninchedda, and M. Pollefeys, "From point clouds to mesh using regression," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3893–3902.

[12] P. Erler, P. Guerrero, S. Ohrhallinger, N. J. Mitra, and M. Wimmer, "Points2Surf learning implicit surfaces from point clouds," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 108–124.

[13] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5939–5948.

[14] D. Levin, "The approximation power of moving least-squares," *Math. Comput.*, vol. 67, pp. 1517–1531, 1998.

[15] D. Levin, "Mesh-independent surface interpolation," in *Geometric Modeling for Scientific Visualization*. Berlin, Germany: Springer, 2004.

[16] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Trans. Vis. Comput. Graphics*, vol. 9, no. 1, pp. 3–15, First Quarter, 2003.

[17] G. Guennebaud and M. H. Gross, "Algebraic point set surfaces," in *Proc. ACM SIGGRAPH Papers*, 2007, pp. 23–es.

[18] Y. Lipman, D. Cohen-Or, D. Levin, and H. Tal-Ezer, "Parameterization-free projection for geometry reconstruction," in *Proc. ACM SIGGRAPH Papers*, 2007, pp. 22–es.

[19] H. Huang, D. Li, H. Zhang, U. M. Ascher, and D. Cohen-Or, "Consolidation of unorganized point clouds for surface reconstruction," in *Proc. ACM SIGGRAPH Asia Papers*, 2009, Art. no. 176.

[20] B. Liao, C. Xiao, L. Jin, and H. Fu, "Efficient feature-preserving local projection operator for geometry reconstruction," *Comput. Aided Des.*, vol. 45, pp. 861–874, 2013.

[21] J. Digne, "Similarity based filtering of point clouds," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, 2012, pp. 73–79.

[22] J. Digne, S. Valette, and R. Chaine, "Sparse geometric representation through local shape probing," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 7, pp. 2238–2250, Jul. 2018.

[23] K. Sarkar, F. Bernard, K. Varanasi, C. Theobalt, and D. Stricker, "Structured low-rank matrix factorization for point-cloud denoising," in *Proc. Int. Conf. 3D Vis.*, 2018, pp. 444–453.

[24] J. Zeng, G. Cheung, M. Ng, J. Pang, and C. Yang, "3D point cloud denoising using graph Laplacian regularization of a low dimensional manifold model," *IEEE Trans. Image Process.*, vol. 29, pp. 3474–3489, Jan. 2020, doi: 10.1109/TIP.2019.2961429.

[25] J. Wang, J. Jiang, X. Lu, and M. Wang, "Rethinking point cloud filtering: A non-local position based approach," *Comput.-Aided Des.*, vol. 144, 2022, Art. no. 103162.

[26] X. Lu, S. Schaefer, J. Luo, L. Ma, and Y. He, "Low rank matrix approximation for 3D geometry filtering," *IEEE Trans. Vis. Comput. Graphics*, vol. 28, no. 4, pp. 1835–1847, Apr. 2022.

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

[28] P. Hermosilla, T. Ritschel, and T. Ropinski, "Total denoising: Unsupervised learning of 3D point cloud cleaning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 52–60.

[29] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. J. Mitra, "PCPNet learning local shape properties from raw point clouds," *Comput. Graph. Forum*, vol. 37, no. 2, pp. 75–85, 2018.

[30] D. Lu, X. Lu, Y. Sun, and J. Wang, "Deep feature-preserving normal estimation for point cloud filtering," *Comput.-Aided Des.*, vol. 125, 2020, Art. no. 102860.

[31] F. Pistilli, G. Fracastoro, D. Valsesia, and E. Magli, "Learning graph-convolutional representations for point cloud denoising," 2020, *arXiv: 2007.02578*. [Online]. Available: https://api.semanticscholar.org/CorpusID:220364174

[32] S. Luo and W. Hu, "Differentiable manifold reconstruction for point cloud denoising," in *Proc. 28th ACM Int. Conf. Multimedia*, 2020, pp. 1330–1338. [Online]. Available: https://api.semanticscholar.org/CorpusID:220793119

[33] H. Chen, Z. Wei, X. Li, Y. Xu, M. Wei, and J. Wang, "RePCD-Net: Feature-aware recurrent point cloud denoising network," *Int. J. Comput. Vis.*, vol. 130, pp. 615–629, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID

[34] S. Luo and W. Hu, "Score-based point cloud denoising," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 4583–4592.

[35] D. de Silva Edirimuni, X. Lu, Z. Shao, G. Li, A. Robles-Kelly, and Y. He, "IterativePFN: True iterative point cloud filtering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 13530–13539.

[36] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," 2015, *arXiv:1506.02025*. [Online]. Available: https://api.semanticscholar.org/CorpusID:6099034

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

[38] P.-S. Wang, Y. Liu, and X. Tong, "Mesh denoising via cascaded normal regression," *ACM Trans. Graph.*, vol. 35, no. 6, 2016, Art. no. 232.

Jinxi Wang received the bachelor's and master of engineering degrees from the College of Information Engineering, Northwest Agriculture and Forestry University in Yangling, China. Her current research focuses on computer graphics and geometric modeling.

Xuequan Lu (Senior Member, IEEE) received the PhD degree from Zhejiang University, China, in 2016. He is a senior lecturer with the Department of Computer Science and IT, La Trobe University, Australia. He spent more than two years as a research fellow in Singapore. His research interests mainly fall into the category of visual computing, for example, geometry modeling, processing and analysis, animation/simulation, 2D data processing, and analysis.

Meili Wang (Member, IEEE) received the PhD degree in computer animation from the National Centre for Computer Animation, Bournemouth University. She is a professor with the College of Information Engineering, Northwest Agriculture and Forestry University. Her research interests include computer graphics, image processing, and virtual reality.

Fei Hou received the PhD degree in computer science from Beihang University, in 2012. He is currently a research associate professor with the Institute of Software, Chinese Academy of Sciences. He was a postdoctoral researcher with Beihang University from 2012 to 2014 and a research fellow with the School of Computer Science and Engineering, Nanyang Technological University from 2014 to 2017. His research interests include geometry processing, image-based modeling, data vectorization, medical image processing, etc.

Ying He (Member, IEEE) received the BS and MS degrees in electrical engineering from Tsinghua University, China, and the PhD degree in computer science from Stony Brook University, USA. He is currently an associate professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests fall into the general areas of visual computing and he is particularly interested in the problems which require geometric analysis and computation.