This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3292464

PCDNF: Revisiting Learning-based Point Cloud Denoising via Joint Normal Filtering

Zheng Liu, Yaowu Zhao, Sijing Zhan, Yuanyuan Liu, Renjie Chen†, Ying He

Abstract—Point cloud denoising is a fundamental and challenging problem in geometry processing. Existing methods typically involve direct denoising of noisy input or filtering raw normals followed by point position updates. Recognizing the crucial relationship between point cloud denoising and normal filtering, we re-examine this problem from a multitask perspective and propose an end-to-end network called PCDNF for joint normal filtering-based point cloud denoising. We introduce an auxiliary normal filtering task to enhance the network's ability to remove noise while preserving geometric features more accurately. Our network incorporates two novel modules. First, we design a shape-aware selector to improve noise removal performance by constructing latent tangent space representations for specific points, taking into account learned point and normal features as well as geometric priors. Second, we develop a feature refinement module to fuse point and normal features, capitalizing on the strengths of point features in describing geometric details and normal features in representing geometric structures, such as sharp edges and corners. This combination overcomes the limitations of each feature type and better recovers geometric information. Extensive evaluations, comparisons, and ablation studies demonstrate that the proposed method outperforms state-of-the-art approaches in both point cloud denoising and normal filtering.

Index Terms—Point cloud denoising, normal filtering, 3D deep learning, point cloud processing

1 Introduction

Point clouds are widely used in various fields, including computer graphics, 3D computer vision, photogrammetry, autonomous driving, simultaneous localization, and mapping (SLAM), among others. With the rapid development of modern 3D digital acquisition devices, such as LiDAR and depth cameras, more and more 3D models are routinely obtained and stored as point clouds in shape repositories. However, due to physical measurement and reconstruction errors, the acquired point clouds are inevitably corrupted by noise [1]. Noise not only degrades the visual quality of 3D models but also causes unexpected problems in downstream applications [2], [3]. Therefore, point cloud denoising is highly desired and often considered the first step in geometry processing. Due to the high-frequency nature of both noise and geometric features, it is challenging to distinguish and recover features while removing noise from point clouds. Point cloud denoising has been a topic of extensive research in the past two decades. While significant progress has been made, traditional denoising methods [4], [5], [6], [7], [8], [9], [10] typically require numerous parameters and tedious parameter tuning. The tuning process is not only time-consuming but also crucial for achieving promising results.

The success of deep neural networks in image processing has recently led to the adoption of data-driven approaches for various tasks in point cloud processing, including denoising. Deep learning methods are automatic and eliminate the need for parameter tuning, making them suitable for a wider range of 3D models than traditional methods. Existing deep denoising methods can generally be classified into one- and two-stage methods.

One-stage methods, such as PointCleanNet [11], Pointfilter [3], RePCD-Net [2], typically use point feature representations to regress a displacement per noisy point and adjust its position to the ground-truth directly. Due to a lack of consideration of normal information, PointCleanNet [11] and RePCD-Net [2] blur sharp features. Pointfilter [3], which incorporates normal information in its loss function, can better preserve sharp features but may oversmooth geometric details.

Two-stage methods, which filter normals followed by updating their locations, have gained widespread attention due to their ability to incorporate local geometry information [1], [12]. The main difference among the two-stage methods is in their normal filtering networks. Similar to Pointfilter [3], small-scale geometric features including fine details may get blurred when using only the learned normal-based features to update point positions. To recover detailed features accurately, the method in [12] relies on the additional feature detection network, while GeoDualCNN [1] needs the guidance of geometry expertise and a feature-preserving position updating algorithm.

The aforementioned learning-based methods either directly smooth noisy points or filter raw normals followed by updating point positions, making them unsuitable for joint denoising and normal filtering. However, denoising and normal filtering tasks are intrinsically intertwined, influencing and benefiting each other. If better normals can be estimated from the noisy point cloud, the performance of the denoising task can be significantly improved, and vice

- *Z. Liu and Y. Zhao are with School of Computer Science, S. Zhan is with National Engineering Research Center of Geographic Information System, Y. Liu is with School of Computer Science, also with Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences (Wuhan).*
- *R. Chen is with School of Mathematical Sciences, University of Science and Technology of China.*
- *Y. He is with School of Computer Science and Engineering, Nanyang Technological University.*

†*Corresponding author. E-mail:renjiec@ustc.edu.cn.*

versa. To date, no existing work can perform denoising and normal filtering tasks jointly.

This observation has led us to develop a point-normal feature interaction network within a joint task paradigm, which can be regarded as a special category of multitask learning. For the rest of this paper, we refer to our multitask learning specifically to joint task learning. Unlike previous work, we present a unified architecture for jointly learning point cloud denoising and normal filtering. The proposed network comprises two branches — one for point cloud denoising and the other for normal filtering — that can mutually benefit each other. The combined technique leverages the best properties of each task while attempting to overcome their respective weaknesses.

Specifically, the proposed network consists of four modules: the multiscale feature extractor, shape-aware selector, feature refinement, and the decoder. The feature extractor utilizes DGCNN [13] as the backbone for learning point and normal feature representations in a multiscale manner. Next, the learned point and normal features, combined with geometric priors, are fed into the shape-aware selector to construct the latent tangent space for the specific point, which helps reduce the negative impact of points not within the tangent space.

Following this, we design a feature refinement module consisting of two units — feature augmentation and fusion — to enhance the network's ability to accurately recover geometric features. The feature augmentation unit aggregates neighboring features to obtain richer point and normal feature representations, while the feature fusion unit integrates the augmented point and normal features to better preserve both structural and detailed geometric features. Finally, the integrated features are fed into the decoder to predict the denoised point coordinates and filtered normals.

To summarize, the main contributions of this work include the following:

- We introduce a novel architecture for jointly learning point cloud denoising and normal filtering. To the best of our knowledge, this is the first end-to-end framework for point cloud denoising that adopts a multitask perspective.
- We develop a shape-aware selecting module to enhance denoising performance by reducing the adverse effects of neighboring points outside the latent tangent space of the specific point. This module represents the tangent space using learned point and normal features combined with geometric priors.
- We design a feature refinement module to boost the network's ability for preserving geometric features. This module first expands the receptive fields of learned features and subsequently integrates the learned point and normal features to better recover various geometric features, such as structural and detailed features.
- Qualitative and quantitative experiments on both synthetic and scanned data demonstrate that our network outperforms state-of-the-art methods in both denoising and normal filtering tasks.

2 Related Work

2.1 Point Cloud Denoising

As a fundamental geometry processing problem, point cloud denoising has garnered significant attention over the past few decades. Given the vast amount of literature on denoising techniques, providing an exhaustive review is beyond the scope of this paper. We refer interested readers to [14] for a comprehensive review. In this section, we briefly discuss traditional methods before focusing on recent learning-based techniques.

Traditional methods. Moving Least Squares (MLS)-related methods [4], [15], [16], [17] project the input point set onto the approximated underlying surface iteratively. Originally designed for reconstructing noise-free surfaces, these classical methods assume piecewise smooth priors for the underlying surface, which can result in the smoothing of geometric features. Later on, Lipman et al. [18] proposed the pioneering Locally Optimal Projection (LOP) method, which has been proven successful for point cloud consolidation. LOP and its variants [5], [19], [20], [21] aim to generate a point set that describes the underlying surface while maintaining a uniform distribution. Despite their ability to robustly remove noise and produce uniformly sampled results, LOP-related methods struggle to preserve geometric features in the presence of large noise. Optimization-based methods, on the other hand, formulate the denoising process as optimization problems with suitable priors. Among them, sparse optimization methods, such as [9], [22], [23], [24], effectively preserve geometric features (particularly sharp ones) by leveraging the sparsity of geometric features over underlying surfaces. Recently, low-rank and dictionary learning techniques [6], [8], [25], [26], [27] have gained attention due to their ability to preserve structural repetition on underlying surfaces by exploiting self-similarity characteristics. Hu et al. [28] proposed a feature graph learning approach and employed it for point cloud denoising with points' positions and normals as features, leading to impressive denoising results. Although optimization-based methods excel in preserving certain geometric features, their reliance on geometric priors may hinder their performance in preserving other types of features. In general, traditional methods involve complex computations or optimization problems and necessitate a tedious trial-and-error process to achieve satisfactory results.

Learning-based methods. Recently, with the development of neural networks [13], [29], [30], [31], [32], deep learning techniques have been introduced into point cloud denoising extensively and achieved impressive results. Roveri et al. [33] proposed PointProNet, a fully differentiable denoising architecture based on 2D CNN, which converts unordered points to regularly sampled height maps. EC-Net [34] and DMRDenoise [35] mainly focus on upsampling and consolidating techniques over point clouds. EC-Net designs an edge-aware consolidation network to denoise point clouds. This method preserves sharp geometric features but retains noise to some extent. Although DMRDenoise can remove noise successfully during the downsampling stage, it may blur geometric features. TotalDenoising, developed by Hermosilla et al. [36], introduces a spatial prior that steers converge to underlying

surfaces without supervision. However, as an unsupervised method, TotalDenoising is sensitive to large noise and may suffer shrinkage artifacts. GPDNet [37] uses the graph-convolutional neural network for denoising. Luo and Hu [38] proposed a paradigm of denoising by exploiting the distribution model of noisy point clouds. Most of the above learning-based methods cannot effectively preserve geometric features, especially sharp features. Moreover, the noise removal performance of these methods decreases evidently as the noise level increases.

More recently, some techniques belonging to the two-stage paradigm [1], [12] - normal filtering followed by updating point coordinates - have been developed for feature-preserving denoising. Lu et al. [12] classified the noisy input as feature points and non-feature points and then predicted multi-normal on the feature points to preserve sharp features. Wei et al. [1] proposed a geometry-supporting dual convolutional neural network (GeoDualCNN) to filter normals and then updated point coordinates to match the filtered normals. Although GeoDualCNN can produce promising results, the requirement of computing the extra homogeneous neighborhood for each point limits its end-to-end applicability. Luo and Hu [38] proposed to first estimate the gradient for the noisy point cloud and then perform denoising via gradient ascent. Chen et al. [39] proposed a global and continuous gradient field model, which can resample degraded point clouds via gradient ascent with the introduced graph Laplacian regularizer.

Compared to these two-stage methods, an alternative paradigm involves developing networks that directly predict displacements of the noisy point cloud and then apply the predicted displacements to reposition point coordinates. Several methods have been developed following this paradigm [2], [3], [11]. PointCleanNet [11] first excludes outliers and then predicts displacement vectors for the remaining noisy points. This method may retain extra noise on the denoised results and tends to smooth sharp features to varying degrees. To address this issue, Pointfilter [3] introduces a loss function preserving features with an encoder-decoder framework. Although Pointfilter can preserve sharp features, it cannot recover small-scale geometric features well. Later, Chen et al. [2] proposed a feature-aware recurrent architecture to learn more representative features for recovering multiscale geometric features effectively. However, in the case of large noise, their method seems to be difficult to keep a balance between noisy removal and geometric feature recovery. More recently, Edirimuni et al. [40] introduced a contrastive learning framework to generate effective patch-wise representations with noise corruption as augmentation, which can infer both displacements and normals simultaneously.

2.2 Point Cloud Normal Filtering

Point normals, which indicate the orientation of the scanned surface, are essential signals that have been widely applied in various practical problems, such as surface reconstruction [41], [42], 3D descriptors [43], and registration [44]. However, accurately estimating point normals is challenging, as captured point clouds are inevitably corrupted by noise and outliers. Consequently, normal filtering has been extensively studied over the past decades. Due to the vast amount of literature on normal filtering, we only review the learning-based methods related to our work.

Since the pioneering work of Qi et al. [29], numerous studies have extended and applied the PointNet architecture to point cloud processing problems. PCPNet [47] was the first to apply the PointNet architecture for estimating normals from noisy point clouds, addressing the normal filtering task. Zhou et al. [48] proposed a plane constraint mechanism to divide neighborhood points into main plane points and error points, using only the learned features of the main plane points to regress normals. Their method is robust to noise and neighborhood scales. To overcome the oversmoothing artifacts, Nesti-Net [49] introduces mixtures of experts to predict the optimal neighborhood scale instead of simply concatenating multiple scales together. Their multiscale strategy can improve performance effectively but leads to evident time consumption. Zhou et al. [50] proposed a normal filter based on multipatch stitching. Thanks to their patch-level architecture, their method can reduce computational costs and improve the robustness of noise removal. Instead of directly predicting normals from the learned features, some methods [51], [52], [53] estimate the normal for a specific point by fitting a local underlying surface through its neighboring points and then compute the normal from the fitting surface. These methods are based on the weighted least squares surface fitting of the local geometric neighborhood, which can improve the generalization ability of their networks on real scanning data [52]. Cao et al. [54] learned a latent tangent space representation with a lightweight network and then utilized a differentiable RANSAC to estimate normals of the underlying surface. Zhou et al. [46] deployed a multi-feature scheme to capture geometric information from multiple feature representations and then updated normals in a refinement system. Sharma et al. [55] proposed a normal estimation network with point upsampling for robust surface reconstruction. Unlike these existing methods, we present a unified architecture for jointly learning point cloud denoising and normal filtering. The combined technique is able to apply the best properties of each of the two tasks, and try to overcome the weakness of both. Thus, it performs well in preserving geometric features and removing noise, and at the same time avoids the artifacts in the results.

3 Method

This section starts with an overview of our framework of point cloud denoising with joint normal filtering. Then, we present our network architecture, followed by elaborating on each module of the network. Finally, an end-to-end joint loss function is introduced.

3.1 Problem Statement

Point cloud denoising is nontrivial due to the ill-posed nature of the problem. Many learning-based methods [2], [3], [11] cast the noise as pointwise residuals (i.e., a displacement vector per input point), and try to predict the residual vectors in order to smooth the input point cloud. However, given only the point positions, it is still challenging

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3292464

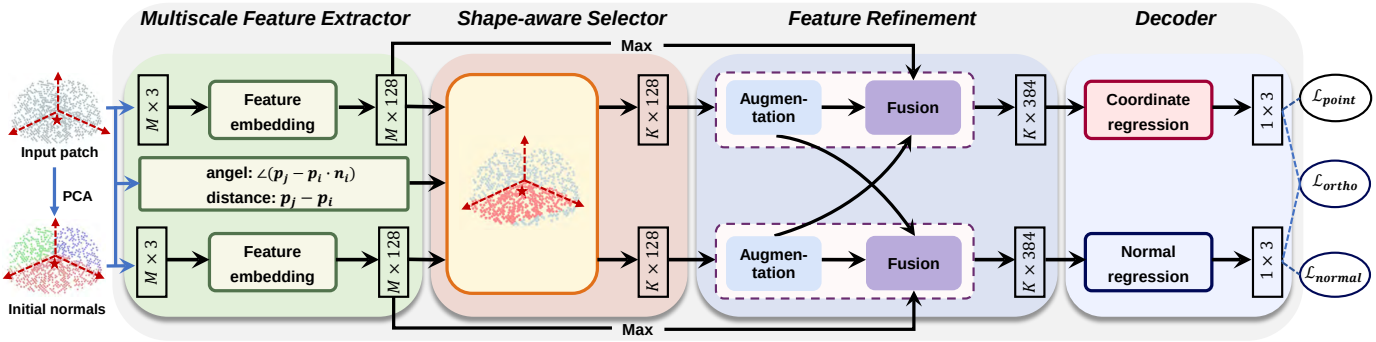


Fig. 1: An overview of our joint point cloud denoising and normal filtering network, coined as PCDNF. PCDNF consists of four main modules: the multiscale feature extractor, shape-aware selector, feature refinement (including feature augmentation and fusion units), and the decoder. Given a noisy patch of a point (along with the corresponding raw normals of the patch), PCDNF is capable of concurrently predicting the coordinate and filtered normal of the specific point.

to achieve satisfactory denoising results while preserving geometric features. Note that, high-quality normals can improve denoising performance; conversely, accurate normals can be computed from high-quality point clouds. Thus, as an alternative to directly predict the displacement vectors from the noisy input, we propose to perform point cloud denoising by combining position correction and normal filtering, which is stated as follows

$$(\widehat{\mathrm{D}}, \widehat{\mathrm{N}}) = \mathcal{F}(\mathrm{P}, \mathrm{N}), \quad \mathrm{P} = \widehat{\mathrm{P}} + \widehat{\mathrm{D}}, \tag{1}$$

where P and $\widehat{\mathrm{P}}$ are the noisy point cloud and its corresponding denoised point cloud, $\widehat{\mathrm{D}}$ denotes the predicted displacement vectors, N denotes the raw normals of the noisy input and $\widehat{\mathrm{N}}$ denotes the corresponding predicted normals. Our method aims to learn a mapping $\mathcal{F} : (\mathrm{P}, \mathrm{N}) \rightarrow (\widehat{\mathrm{D}}, \widehat{\mathrm{N}})$ for predicting displacement vectors and filtered normals simultaneously. Then, the denoised point cloud can be derived from (1) straightforwardly.

3.2 Network Architecture

Based on the problem statement (1), we design a multi-task network, dubbed PCDNF, for joint point cloud denoising and normal filtering. Fig. 1 shows the architecture of PCDNF. Our network consists of four modules: the multiscale feature extractor, shape-aware selector, feature refinement, and decoder.

Specifically, given a noisy patch and its corresponding raw normals, the multiscale feature extractor embeds the inputs into the coarse representations of point and normal features. Then, the shape-aware selector selects the points highly related to the specific point in terms of geometric information and coarse representations. These selected points form a latent tangent space of the specific point, making noise removal more effective. The feature refinement module first encodes local spatial information to augment the similarity representations and then fuses the augmented representations for better geometric feature preservation. Finally, the coordinate and filtered normal of the specific point can be predicted by the coordinate and normal regressors of the decoder.

3.2.1 Multiscale feature extractor

Given a point p_i of the noisy point cloud P, a patch centered at p_i is defined as

$$\mathrm{P}_i = \{p_j \,|\, \|p_j - p_i\| < r\} \in \mathbb{R}^{M \times 3}.$$

M is the number of points within the patch and r is the patch radius. The corresponding raw normals of P_i can be denoted as $\mathrm{N}_i = \{n_j\} \in \mathbb{R}^{M \times 3}$. It is known that the features learned from a single-scale receptive field cannot faithfully describe the local shape of the underlying surface. To address this issue, we propose a multiscale feature extractor using a series of EdgeConv operations [13], which can learn multiscale discriminative representations in both the Euclidean and the feature spaces.

Given an input patch with coordinates P_i and normals N_i, our multiscale feature extractor learns a coarse point feature $\mathrm{F}_{\mathrm{P}_i} = \{f_{p_j} \,|\, \forall p_j \in \mathrm{P}_i\} \in \mathbb{R}^{M \times 128}$ from P_i, and a coarse normal feature $\mathrm{F}_{\mathrm{N}_i} = \{f_{n_j} \,|\, \forall n_j \in \mathrm{N}_i\} \in \mathbb{R}^{M \times 128}$ from N_i. Fig. 2a provides details on our feature embeddings. We elaborate on the process of extracting the coarse point feature $\mathrm{F}_{\mathrm{P}_i}$. The normal feature embedding is done similarly. For any point $p_j \in \mathrm{P}_i$, to extract its pointwise feature f_{p_j}, we first construct two k-nearest neighbors (kNN) graphs of different sizes in order to capture multiscale geometric information around p_j. Then, we compute the features of p_j using the EdgeConv operation [13] with graph sizes k_1, k_2 in the 1-st layer as

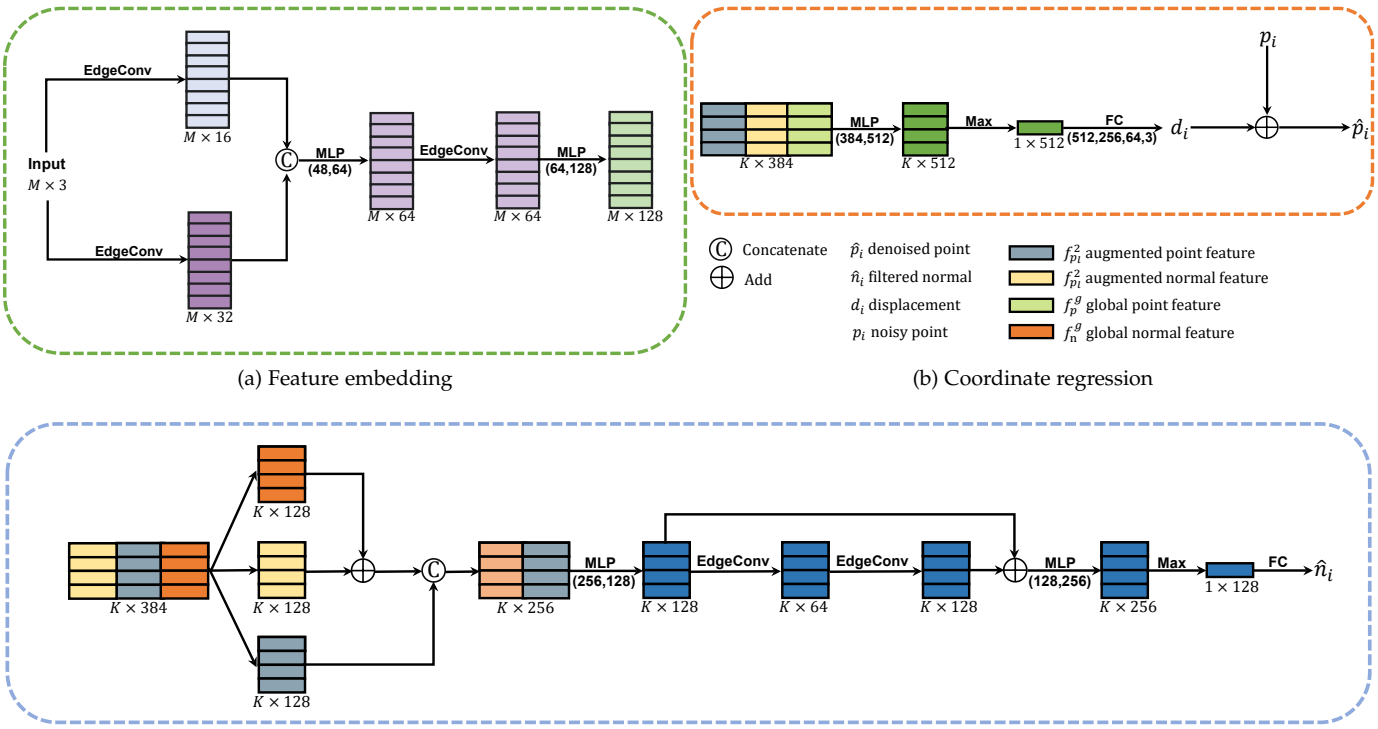
$$f_{p_j}^{1,k_1} = \max_{q \in \mathcal{N}_{k_1}(p_j)} \mathrm{h}_\theta(p_j, q - p_j),$$
$$f_{p_j}^{1,k_2} = \max_{q \in \mathcal{N}_{k_2}(p_j)} \mathrm{h}_\theta(p_j, q - p_j),$$

where $\mathcal{N}_{k_1}(p_j)$ and $\mathcal{N}_{k_2}(p_j)$ denote the neighbors of point p_j in the graphs with sizes k_1 and k_2, h_θ denotes the multi-layer perception (MLP) parameterized by θ, $\max$ denotes the max pooling operation. To make the pointwise feature more discriminative, we concatenate features $f_{p_j}^{1,k_1}$ and $f_{p_j}^{1,k_2}$, followed by MLP as

$$f_{p_j}^{1} = \mathrm{MLP}(\mathrm{concat}(f_{p_j}^{1,k_1}, f_{p_j}^{1,k_2})),$$

where concat is the concatenation operation. To further enlarge receptive fields and object relations in the feature

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3292464



(a) Feature embedding

(b) Coordinate regression

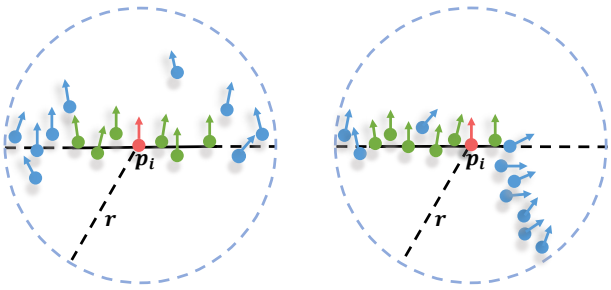
(c) Normal regression

Fig. 2: Illustration of (a) feature embedding and decoder consisting of (b) coordinate regression and (c) normal regression.

space, we perform EdgeConv in the feature space followed by MLP as

$$f_{p_j} = \mathrm{MLP}\left(\max_{q \in \mathcal{N}_{k_3}(p_j)} \mathrm{h}_\theta(f^1_{p_j}, q - f^1_{p_j})\right),$$

where f_{p_j} is the pointwise feature of p_j with graph size k_3. Thus, the coarse point feature $\mathrm{F}_{\mathrm{P}_i}$ of patch P_i can be derived easily. Similarly, we can learn the pointwise normal feature f_{n_j}, and obtain the coarse normal feature $\mathrm{F}_{\mathrm{N}_i}$ of the patch.



(a) (b)

Fig. 3: Illustration for selecting similar points to the specific point p_i plotted in red. The selected similar points in the same latent space of p_i are plotted in green, and the negative points are plotted in blue. The solid line is the latent tangent space of p_i. The dashed line is the patch radius.

3.2.2 Shape-aware selector

The extracted coarse representations $\mathrm{F}_{\mathrm{P}_i}$ and $\mathrm{F}_{\mathrm{N}_i}$ for point p_i can roughly describe the overall shape of patch P_i. How-

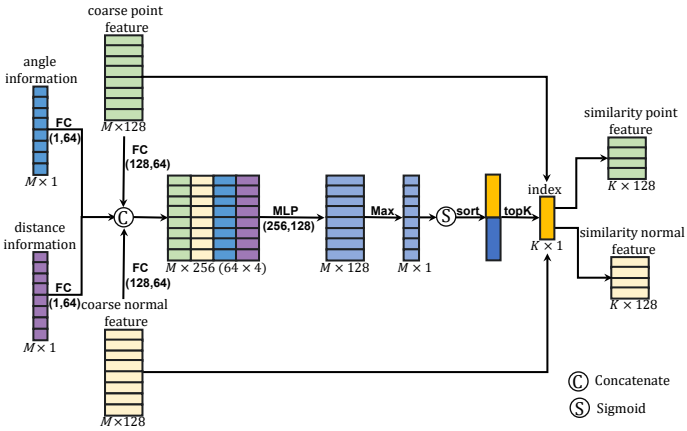


Fig. 4: Illustration of the shape-aware selector.

ever, outliers, different scales and types of noise, and sampling anisotropy may be contained in the noisy patch. These defects of the patch inevitably hurt network performance and lead to unsatisfactory denoising results. For example, suppose the patch of p_i contains outliers and heavy-noise points; see Fig. 3a. In this case, these points negatively influence the representations of the patch, which may degrade denoising results, causing problems such as residual noise or shape collapsing. Furthermore, if the patch contains sharp features, the neighbors outside the same tangent space of p_i have negative influences on feature representations of the patch, which may blur geometric features in the denoising results; see Fig. 3b.

In order to address the above problems, we design a

shape-aware selector to choose points similar to the specific point within the patch, which can greatly reduce the influences of negative points. Fig. 4 shows the detailed structure of the shape-aware selector. As we can see, the selector leverages the guidance of four types of information, including the coarse point and normal features learned by our feature extractor and two geometry information (distance and angle information). Specifically, we first apply fully connected layers to extract the feature from the four types of information, respectively. Then, we calculate the similarity score between point p_j and p_i using the following score function:

$$w_{p_j} = \text{score}\big(\text{concat}(\text{FC}(ang_j), \text{FC}(dist_j), \text{FC}(f_{p_j}), \text{FC}(f_{n_j}))\big), \quad \forall p_j \in \text{P}_i, \forall n_j \in \text{N}_i, \tag{2}$$

where FC denotes fully connected layers. ang_j and $dist_j$ represent angle and distance information of p_j defined as

$$ang_j = \angle(p_j - p_i, n_j), \; dist_j = \exp(-\|p_j - p_i\|^2).$$

The implementation of score function (2) is shown in Fig. 4. It produces a similarity vector $W_{\text{P}_i} = \{w_{p_j}\} \in \mathbb{R}^{M\times 1}$, recording the degree of similarity between point p_j and p_i. Then, the points within the patch that have top-K scores are retained, while the others are discarded as follows:

$$\text{P}_i^K = \{p_l | l \in \text{topK}(W_{\text{P}_i})\} \in \mathbb{R}^{K\times 3},$$

where topK is the function that extracts the indices of the K largest elements of the given input, and P_i^K is the similarity point set to the specific point. As a result, we can obtain the similarity representation for point and normal vectors from the similarity point set within the patch as

$$\text{F}_{\text{P}_i}^1 = \{f_{p_l}\} \in \mathbb{R}^{K\times 128}, \; \text{F}_{\text{N}_i}^1 = \{f_{n_l}\} \in \mathbb{R}^{K\times 128},$$

where f_{p_l} and f_{n_l} are pointwise coordinate and normal features of point $p_l \in \text{P}_i^K$. Fig. 5 shows examples of selecting

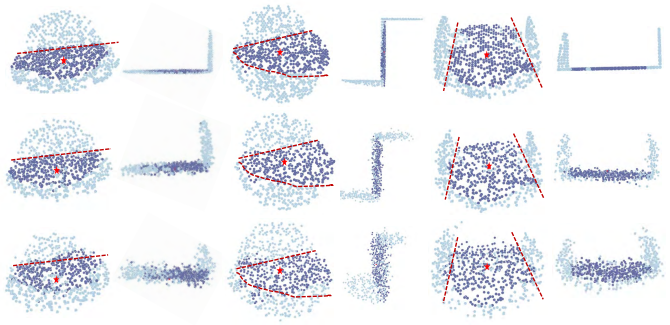


Fig. 5: Illustration of similar points selected from three different shaped patches. From top to bottom: patches corrupted with 0%, 0.25%, 0.5% noise, respectively. From left to right: for each patch, we show the front and side views of the patch to demonstrate the selected points (plotted in dark blue) similar to the specific point (plotted in red). The negative points to the specific point are plotted in light blue.

similar points from different shapes of patches using our shape-aware selector. Consistent with our intuition, our selector is robust against noise and can choose similar points distributed on the tangent space of the specific point.

Remark 1. We provide an intuitive interpretation of the shape-aware selector. To effectively preserve geometric features for denoising and normal filtering tasks, our selector chooses neighboring points that do not span geometric features (residing on the same tangent space). Our method relies solely on the prior information and feature-preserving loss function for the two related tasks to ensure the co-planarity of the selector, eliminating the need to label these co-planar points. In future work, we plan to investigate the use of the selector for weakly supervised plane segmentation.

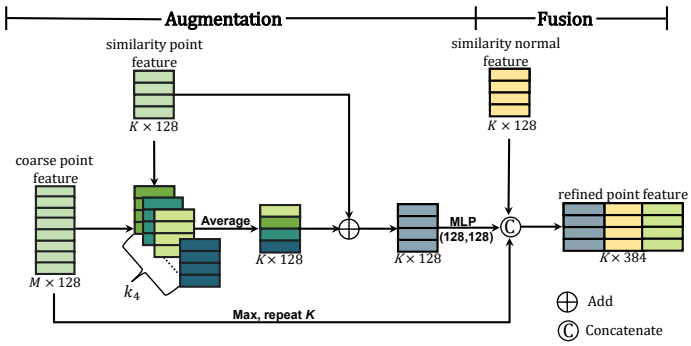


Fig. 6: Illustration of the feature refinement module consisting of feature augmentation and fusion units, using point cloud denoising network branch as an example.

3.2.3 *Feature refinement*

Due to the unavoidable trade-off between noise removal and preservation of geometric features, we can remove the noise effectively but inevitably blur some small-scale geometric features through the previous two modules (the multiscale feature extractor and shape-aware selector). To address this issue, we propose a feature refinement module to facilitate the feature-preserving ability of the overall network. Fig. 6 shows the proposed module consisting of two units: feature augmentation and feature fusion, which will be detailed in the following.

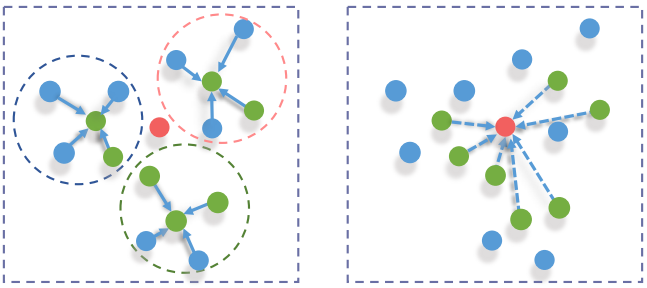
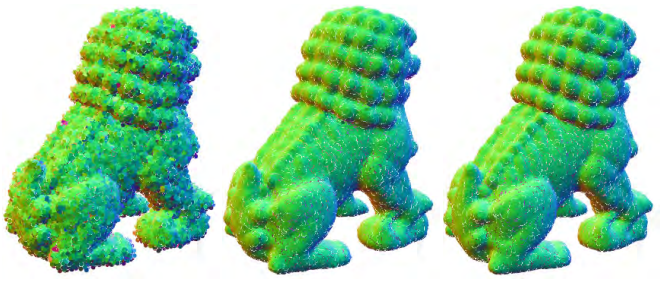


Fig. 7: (a) Illustration of the feature augmentation unit, which enlarges receptive fields (dotted circles) of similar points plotted in green. Thus, the specific point (plotted in red) can obtain a more representative feature from those augmented features of similar points, illustrated in (b).

Feature augmentation. After the stage of similar point selection, we sequentially augment the learned features of similar points by enlarging the receptive fields for discovering more



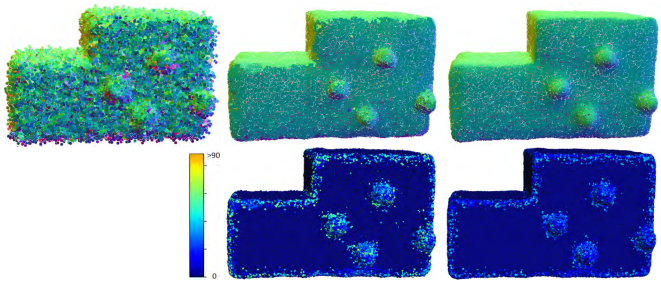
(a) (b) (c)

Fig. 8: (a) Noisy input. (b) Denoising result without the feature augmentation unit. (c) Denoising result with the feature augmentation unit.

locally geometric details; see Fig. 7 for illustration. To learn more representative features, for each similar point (to the specific point within the patch), we first search KNN neighbors of it and gather the features of the neighbors according to the similarity scores. Then, the augmented point and normal features of each similar point are computed as

$$f_{p_l}^2 = \mathrm{MLP}\left(f_{p_l} + \left(\frac{1}{k_4}\sum_{q\in\mathcal{N}_{k_4}(p_l)} w_q f_q\right)\right),$$
$$f_{n_l}^2 = \mathrm{MLP}\left(f_{n_l} + \left(\frac{1}{k_4}\sum_{q\in\mathcal{N}_{k_4}(p_l)} w_q f_{n_q}\right)\right),$$

where k_4 is the neighboring number of p_l, and w_q is the similarity score obtained with (2). Fig. 8 shows the capability of the feature augmentation unit. Fig. 8c shows that the proposed unit plays a key role in recovering local geometric features. Without this unit, some geometric details and shallow structures are blurred in the result; see Fig. 8b.



(a) (b) (c)

Fig. 9: Illustration of the feature fusion unit. (a) Noisy input. (b) Denoising and normal filtering results without the feature fusion unit. (c) Denoising and normal filtering results with the feature fusion unit. The second row visualizes the normal error maps, measured as the angular differences between the filtered normals and the ground truth.

Feature fusion. The feature fusion unit is motivated by the following observation. The point features are more instrumental in recovering local geometric features, while the normal features are more suitable for recovering sharp edges, corners, and smooth transition regions. Thus, the fusion of point and normal features can help better recover various types of geometric features. In addition, the global feature can better describe the whole shape of the patch. Specifically, for each similar point within the patch, we concatenate the three types of features together (including point and normal features and the global feature) to obtain the fused features as

$$f_{p_l}^3 = \mathrm{concat}(f_{p_l}^2, f_{n_l}^2, f_{\mathrm{P}_i}^g),$$
$$f_{n_l}^3 = \mathrm{concat}(f_{n_l}^2, f_{p_l}^2, f_{\mathrm{N}_i}^g),$$

where $f_{p_l}^3$, $f_{n_l}^3$ are the refined point and normal features of the similar point. $f_{\mathrm{P}_i}^g$ and $f_{\mathrm{N}_i}^g$ are global features of the patch which are obtained by feeding the coarse features $\mathrm{F}_{\mathrm{p}_i}$ and $\mathrm{F}_{\mathrm{N}_i}$ into the max-pooling operator, respectively. We validate the effectiveness of our feature fusion unit in Fig. 9. As Figs. 9b and 9c show, incorporating normal features into point features allows the denoising task to better preserve sharp geometric features and smooth regions while incorporating point features into normal features can yield a more satisfactory normal filtering result, see the normal error maps in the second row of Figs. 9b and 9c.

3.2.4 Decoder

Given a point p_i, the feature refinement module generates refined point and normal features of the patch P_i, $\mathrm{F}_{\mathrm{P}_i}^3 = \{f_{p_l}^3\} \in \mathbb{R}^{K\times 384}$ and $\mathrm{F}_{\mathrm{N}_i}^3 = \{f_{n_l}^3\} \in \mathbb{R}^{K\times 384}$, which are inputs to our decoder. Our decoder includes two regressors: coordinate and normal regression. For coordinate regression, we apply MLP and FC layers to predict the displacement vector from the refined point feature $\mathrm{F}_{\mathrm{P}_i}^3$, and then obtain the denoised point by adding the predicted displacement vector to the original coordinate; see Fig. 2b. For normal regression, we use ResNet-like operations [56] to obtain the filtered normal from the refined normal feature $\mathrm{F}_{\mathrm{N}_i}^3$; see the details in Fig. 2c.

3.3 Training Losses

To train our network in an end-to-end manner, we design three types of losses as our optimization objectives: a point-denoise loss, a normal-filter loss, and an orthogonality loss. The joint loss function is formulated as

$$\mathcal{L} = \lambda_1\mathcal{L}_{\mathrm{point}} + \lambda_2\mathcal{L}_{\mathrm{normal}} + \lambda_3\mathcal{L}_{\mathrm{ortho}}, \tag{3}$$

where λ_1, λ_2, and λ_3 are three hyperparameters that balance the importance of each term. We empirically set $\lambda_1 = 100$, $\lambda_2 = 10$ and $\lambda_3 = 10$ for training.

Point-denoise loss. To ensure that the denoising result can approximate the underlying surface while preserving sharp features well, we apply a bilateral mechanism proposed in [3] to compute the project distance between the denoised point and its neighboring points within the ground-truth patch. To further improve the distribution of the denoising result, we follow [2], [3], [18] and adopt a repulsive term to penalize those points that are too close to each other. Thus, our two-term point denoising loss is defined as follows:

$$\mathcal{L}_{\mathrm{point}} = \alpha\frac{\sum_{\bar{p}_j\in\overline{\mathrm{P}}_i}\left|(\hat{p}_i-\bar{p}_j)\cdot\bar{n}_j\right|\cdot\phi\left(\|\hat{p}_i-\bar{p}_j\|\right)\theta\left(\bar{n}_i,\bar{n}_j\right)}{\sum_{\bar{p}_j\in\overline{\mathrm{P}}_i}\phi\left(\|\hat{p}_i-\bar{p}_j\|\right)\theta\left(\bar{n}_i,\bar{n}_j\right)} + (1-\alpha)\max_{\bar{p}_j\in\overline{\mathrm{P}}_i}\|\hat{p}_i-\bar{p}_j\|,$$

where α is a parameter that balances the denoising and uniform distribution terms. α is set to 0.97 referring to [3]. $\overline{\mathrm{P}}_i$ is the ground-truth patch centered at the denoised point $\hat{p}_i$, $\bar{n}_i$ is the ground-truth normal of the clean point which is nearest to the denoised point $\hat{p}_i$, $\bar{n}_j$ is the ground-truth normal of $\bar{p}_j$. ϕ and θ are two monotonically decreasing functions in terms of distance and normal deviation. ϕ is the Gaussian function, and θ is given as $\theta(\bar{n}_i, \bar{n}_j) = \exp(-\frac{1-\bar{n}_i \cdot \bar{n}_j}{1-\cos(15^\circ)})$.

Normal-filter loss. For the denoised point $\hat{p}_i$, we simply use the Euclidean distance between its filtered normal $\hat{n}_i$ and the ground truth $\bar{n}_i$ as our normal-filter loss:

$$\mathcal{L}_{\text{normal}} = \|\hat{n}_i - \bar{n}_i\|^2.$$

Orthogonality loss. To encourage the denoised point to move in the direction of its filtered normal while ensuring the orthogonality between the filtered normal and the edges connecting the denoised point and its neighboring points, we propose the following orthogonality loss that can constrain the denoising and normal filtering branches at the same time:

$$\mathcal{L}_{\text{ortho}} = \sum_{\bar{p}_j \in \overline{\mathrm{P}}_i} \left(w_{p_j} \left|(\hat{p}_i - \bar{p}_j) \cdot \hat{n}_i\right|\right)^2.$$

Remark 2. Our method employs several mechanisms and modules, including multitasking and a shape-aware selecting module, to achieve high-quality results. Even in the presence of high levels of noise, our method can progressively improve the filtered normal (used in the orthogonality loss), and make it to be consistent with the ground-truth normal (used in the point-denoise loss). Our multitask network can be iteratively performed to enhance the results (denoising and normal filtering) gradually. The selecting module reduces the negative impact of large noise, resulting in high-quality results. Additionally, our normal-filter loss optimizes the filtered normal to closely match the ground-truth normal.

4 Experiments and Discussions

We evaluate our method for denoising and normal filtering tasks visually and numerically and show its superiority compared to the SOTA denoising and normal filtering methods. We further modify our method and conduct ablation studies to validate the effectiveness of each design choice made in our method.

4.1 Experimental Settings

Dataset. To train our end-to-end network for denoising and normal filtering tasks, we adopt the dataset provided by [3]. The training set contains 11 CAD and 11 non-CAD models (clean data without noise). The ground truth point clouds with normal information can be obtained by randomly sampling the clean models. The number of sample points for each model is set uniformly as 100K. For these ground truth point clouds, we corrupt each of them by Gaussian noise with standard deviations of 0.25%, 0.5%, 1%, 1.5%, and 2.5% to the diameter of the bounding box. Thus, the final training set contains 110 noisy point clouds (normals are estimated with PCA) and the corresponding 22 ground truth point clouds (with normals).

To conduct comparisons more effectively for both tasks, we construct a test set consisting of synthetic point clouds and raw scanned data. For synthetic data, we rely on the synthetic dataset released in [3] including three categories: simple, medium, and complicated, in which there are 7, 6, and 7 clean models, respectively. The raw scanned data will be introduced in the following experimental section. Again, the number of sampled points for each model is unified to 100K. Each of the clean point clouds is perturbed by Gaussian noise with standard deviations of 0.25%, 0.5%, 1%, 1.5%, and 2.5% to the diagonal of the bounding box.

Network inference. Our multitask network excels at producing feature-preserving denoising results and satisfactory normal filtering results simultaneously. Additionally, our method can be iteratively applied to further refine the results when the noise level is high. In this process, the denoised points and filtered normals from previous iterations can serve as inputs to our method for further refinement. It is important to note that this iterative process is only performed during the inference phase.

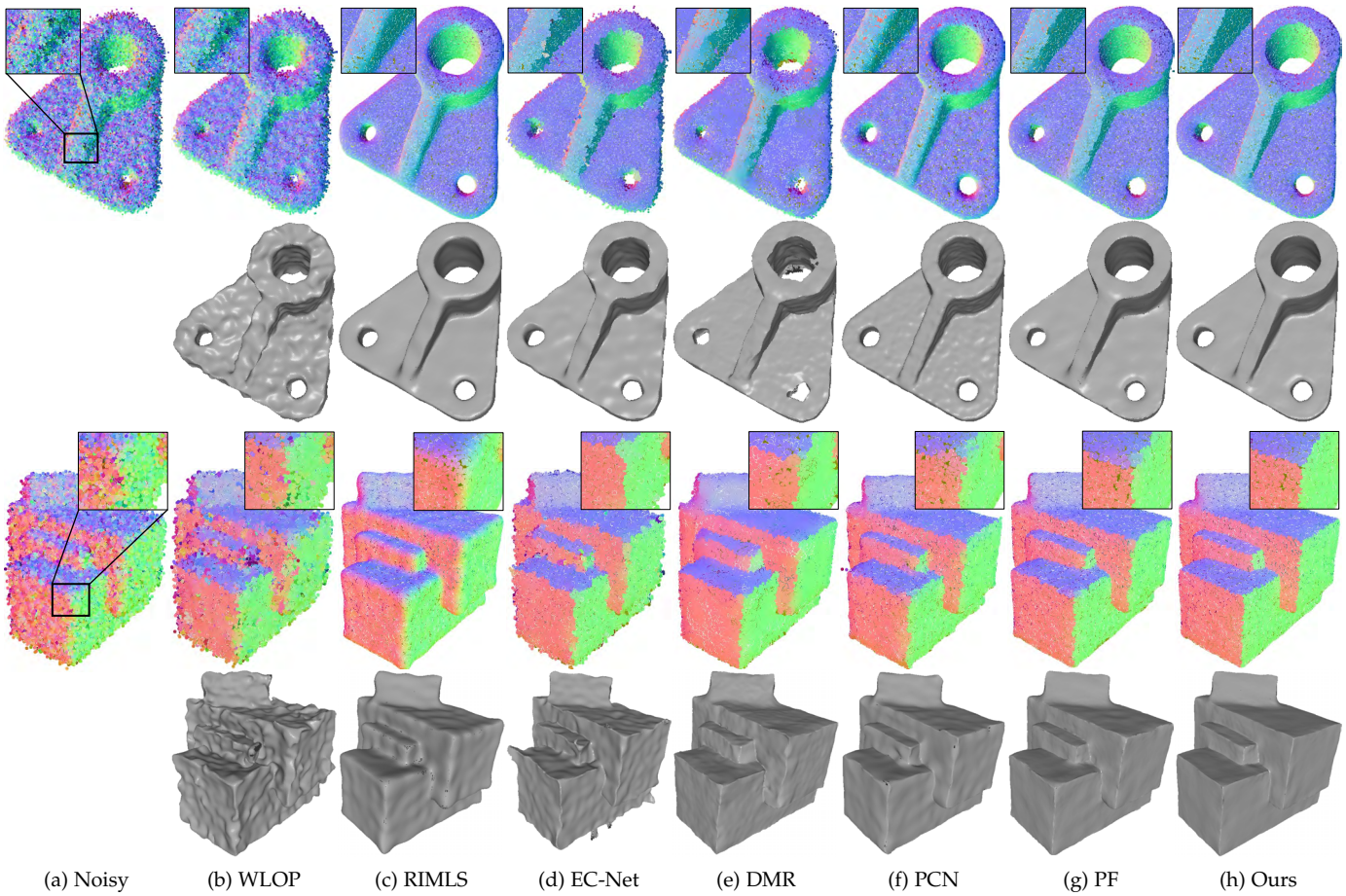
Implementation details. In our implementation, the patch radius r is set to 5% of the diagonal length of the bounding box by default. We also set the number of points within each patch as $M = 512$. If the patch has insufficient points (< 512), we pad the origin, and if it has sufficient points (> 512), we randomly select a subset. Our feature extractors employ three scales, two based on Euclidean distance ($k_1 = 8, k_2 = 16$) and one in feature space ($k_3 = 16$). For the topK strategy of the shape-aware selector, we set $K = \frac{1}{2}M = 256$. The neighborhood size in the feature augmentation unit is set to $k_4 = 10$. Our network is implemented in PyTorch and trained on a single NVIDIA GTX 2080Ti GPU for 45 epochs using the SGD optimizer. The learning rate decreased from 1e-4 to 1e-8. Upon publication, we will release our source code and pretrained models on GitHub.

4.2 Experiments for Point Cloud Denoising

We present visual and numerical comparisons between our method and SOTA denoising methods, including WLOP [19], RIMLS [4], EC-Net [34], DMRDenoise (DMR) [35], PointCleanNet (PCN) [11], and Pointfilter (PF) [3]. For WLOP and RIMLS, we perform the code provided by their authors and carefully tune the parameters to produce denoised results. For DMR and PCN, we use the code released by their authors to retrain new models over our training set. For EC-Net and PF, we use the pretrained models provided by the authors. Sometimes, we reconstruct the denoised results produced by the tested methods for enhancing visual effects via RIMLS (provided by Meshlab for feature-preserving reconstruction).

Synthetic data. Fig. 10 demonstrates comparisons of CAD surfaces including sharp features and smooth regions. The tested CAD surfaces are corrupted by a significant amount of noise. As depicted in Fig. 10, WLOP and EC-Net exhibit excessive noise in their results when the noise level is high. DMR is effective at removing the noise, but it distorts the overall shapes of the surfaces, as illustrated in Fig. 10e. While RIMLS and PCN recover smooth regions well, they blur sharp features to varying degrees, as shown

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3292464



(a) Noisy (b) WLOP (c) RIMLS (d) EC-Net (e) DMR (f) PCN (g) PF (h) Ours

Fig. 10: Denoising results of synthetic data with CAD models. From left to right: noisy input, results produced by WLOP, RIMLS, EC-Net, DMR, PCN, PF, and our method, respectively. The first and third rows show the denoised results with 1% noise and their zoomed views. The second and fourth rows show the corresponding surface reconstruction results. Our method better removes noise and preserves sharp features.

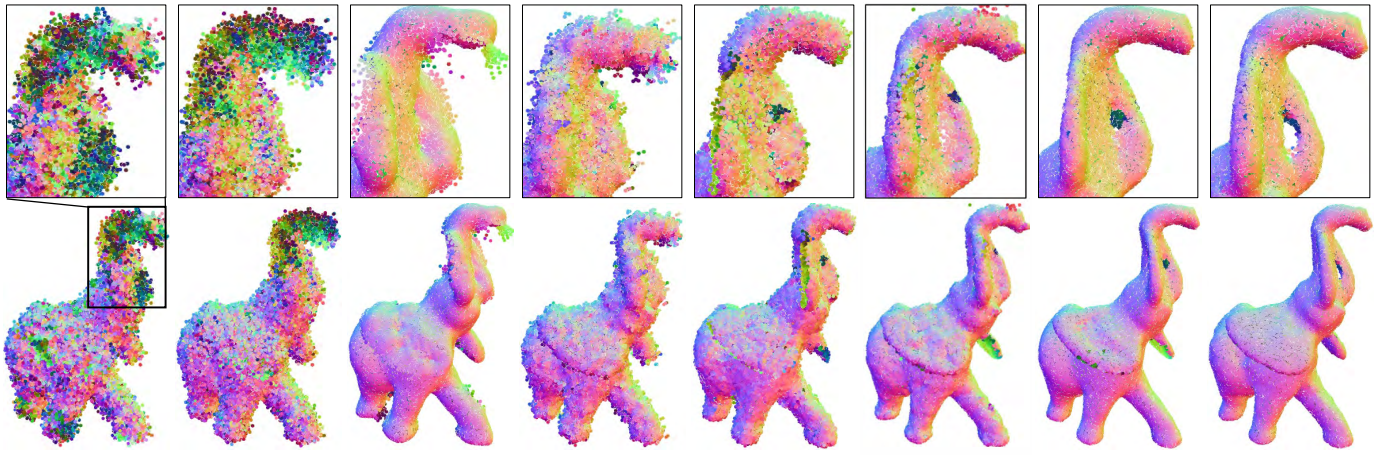
in Figs. 10c and 10f. Additionally, PCN may induce some artifacts in the smooth regions of the results at the tested noise level. In contrast, our method and PF effectively preserve sharp features. Compared to PF, our method more accurately recovers sharp features, as shown in Figs. 10g and 10h. As a result, our method is superior to the compared methods in preserving sharp features and smooth regions simultaneously.

Fig. 11 gives comparisons of a non-CAD surface corrupted by considerable noise. WLOP and EC-Net fail to remove noise entirely in this example. Although DMR does a good job of noise removal, it causes shape distortion and induces bumps in the result, as shown in Fig. 11e. As Figs. 11c and 11f show, RIMLS and PCN seem to have difficulty balancing the performance of noise removal and feature recovery. PF and our method can generate visually better results than the other tested methods. However, from the zoomed views of 11g and 11h, we observe that the trunk of the elephant in the result of PF is swelling; in contrast, our result does not induce this artifact. Therefore, our method visually yields the best result for faithfully preserving geometric features.

Fig. 12 shows comparisons of a non-CAD surface with multiscale geometric features. The tested non-CAD surface is corrupted by moderate noise. Except for WLOP, all the other methods can remove noise effectively. DMR flattens medium- and small-scale features and induces distortion and swelling in the ear regions of the bunny surface, as depicted in Fig. 12e. RIMLS and PF oversmooth small-scale features to varying degrees, and PF makes this situation even worse; as illustrated in Figs. 12c and 12g. Furthermore, although EC-Net and PCN can preserve different levels of geometric features in a better manner, they induce some artifacts in the ear regions of the results. These artifacts degrade the visual quality of the denoised results and further affect the reconstruction results; see the zoomed views and reconstruction results of Figs. 12d and 12f. In contrast, our method outperforms the other methods in terms of recovering most geometric features and preventing inducing annoying artifacts.

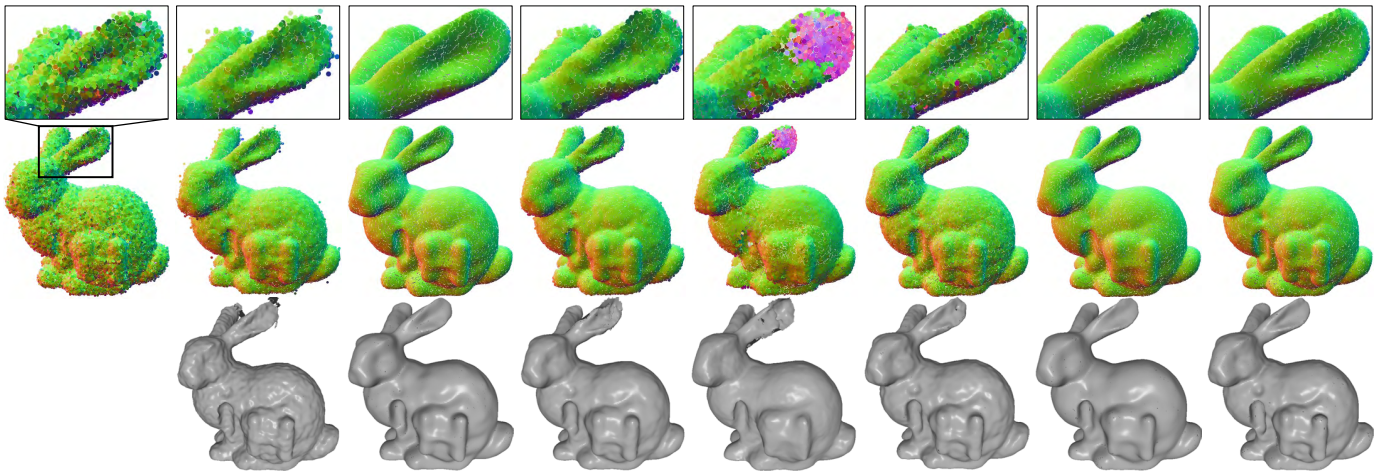
Scanned data. To further validate the efficacy of our method, we conducted experiments on real scanning data without retraining the method. Fig. 13 shows the results for Kinect scanning data. The first and second rows reveal that for CAD surfaces, the other methods retained additional bumps on the denoised results, while our method

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3292464



(a) Noisy (b) WLOP (c) RIMLS (d) EC-Net (e) DMR (f) PCN (g) PF (h) Ours

Fig. 11: Denoising results of synthetic data with 1% non-CAD models. From left to right: noisy input, results produced by WLOP, RIMLS, EC-Net, DMR, PCN, PF, and our method, respectively. The zoomed views, shown in the first row, highlight that our method better preserves detailed features.



(a) Noisy (b) WLOP (c) RIMLS (d) EC-Net (e) DMR (f) PCN (g) PF (h) Ours

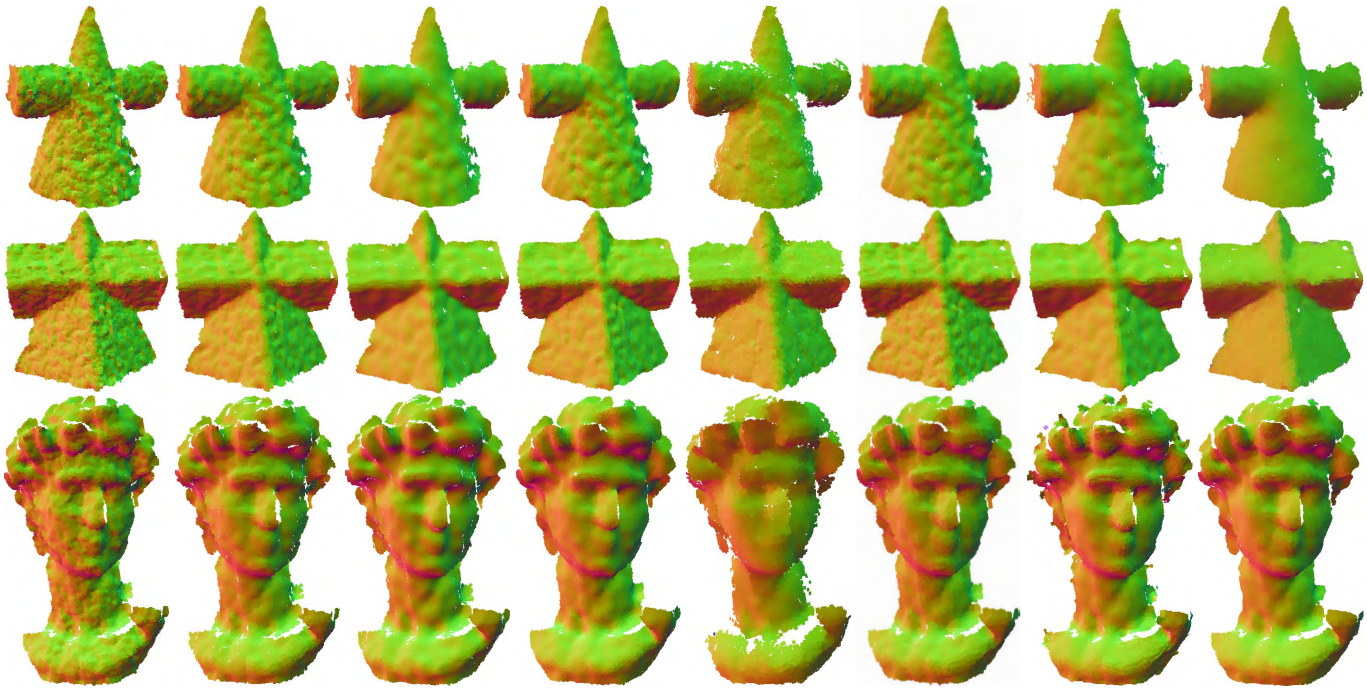
Fig. 12: Denoising results of synthetic data with 0.5% non-CAD models. From left to right: noisy input, results produced by WLOP, RIMLS, EC-Net, DMR, PCN, PF, and our method, respectively. The zoomed views, shown in the first row, highlight that our method better preserves multiscale features.

effectively prevents visible artifacts, demonstrating superior performance in preserving geometric features and recovering smooth regions. For the non-CAD surface (David) in the third row of Fig. 13, RILMS, PF, and our method generated more natural results than other methods. Yet, numerical metrics in Table 1 indicate that our method outperforms RIMLS and PF in dealing with these scanned surfaces which contain complex surface characteristics.

To evaluate our method's effectiveness further, we tested it on laser-scanned data. As the zoomed views in Figure 14 indicate, while all the tested methods can remove noise and preserve geometric features to some extent, our method produces more appealing results, preserving neat structural features and recovering clean smooth regions. Finally, we tested our method on raw outdoor scenes from the Paris-rue-Madame Database, as illustrated in Fig. 15. Compared to the other methods, our approach produces better denoised results, with fewer outliers while removing large noise in the scene.

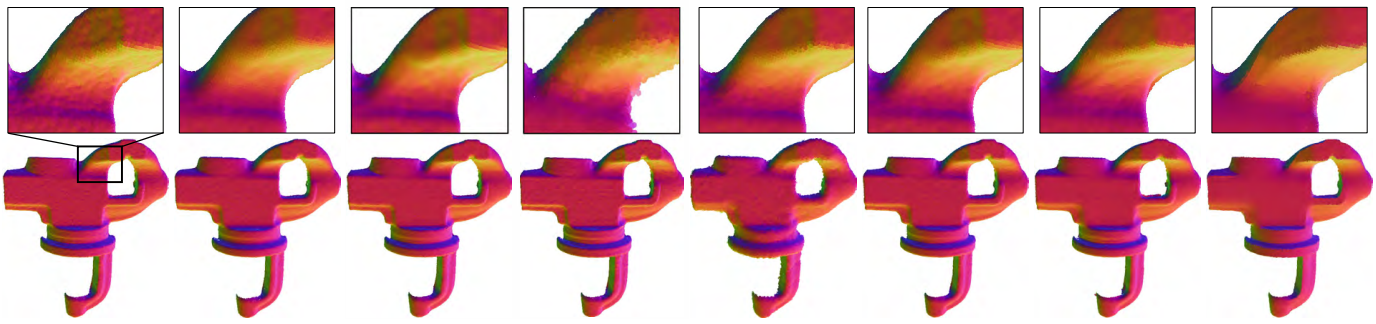
Robustness tests. To further verify the robustness of our method, we demonstrate the performance of our method against irregular sampling, different noise levels, and outliers in the following. We show the effectiveness of our method against non-uniform sampling in Fig. 16. Although the noisy surface suffers from irregular sampling, our method is noticeably better than all other compared methods, which can produce a compelling result that preserves sharp edges and corners. Fig. 17 shows the robustness of our method against different noise levels. As we can see in Figs. 17a, 17b, and 17c, our method can remove noise while preserving sharp features well when the noise level is increased. However, when the noise level is larger than

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3292464



(a) Noisy (b) WLOP (c) RIMLS (d) EC-Net (e) DMR (f) PCN (g) PF (h) Ours

Fig. 13: Denoising results of data captured by Kinect. From left to right: noisy input, results produced by WLOP, RIMLS, EC-Net, DMR, PCN, PF, and our method.



(a) Noisy (b) WLOP (c) RIMLS (d) EC-Net (e) DMR (f) PCN (g) PF (h) Ours

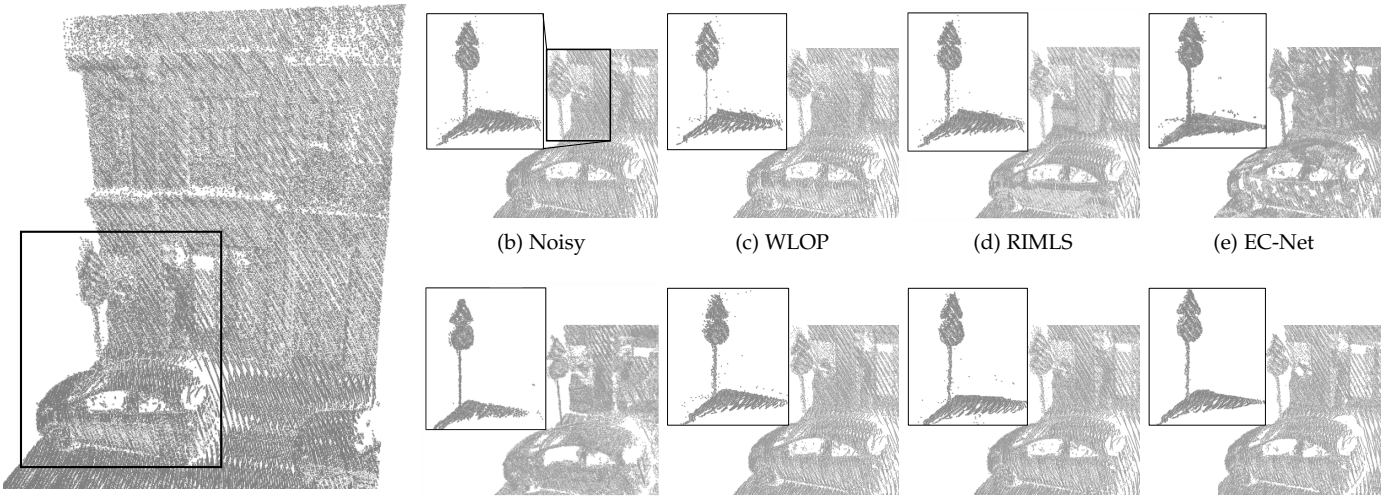
Fig. 14: Denoising results of the laser scanned point cloud. From left to right: noisy input, results produced by WLOP, RIMLS, EC-Net, DMR, PCN, PF, and our method. The zoomed views, highlight that our method better keeps geometrical features.

the geometric feature sizes, our method may blur some geometric features and cannot produce satisfactory results; see Fig. 17d. Since our method takes into account the handling of outliers, it can deal with the surface corrupted by a larger number of outliers effectively; see Fig. 18.

Quantitative evaluation. We observe from the aforementioned qualitative comparisons that our method can generate visually better denoised results than competing methods. Here, we compare them numerically. We utilize Chamfer distance (CD) and Point-to-surface distance (P2S) to measure the fidelity of the denoised result to the ground truth point cloud. The evaluation metrics are widely used in work [2], [3], [11]. We first compare the examples in Figs. 10, 11, 12 and 13 and list the evaluation results in Table 1. As we can see, our method outperforms the competing ones because our CD and P2S errors are significantly smaller than all the other compared methods. Then we further compare our method to those learning-based methods (EC-Net, DMR, PCN, and PF) in the test set and show the evaluation results in Table 2. Our method (PCA) takes low-quality initial normals estimated by PCA as input, and our method (AdaFit) takes the better normals estimated by AdaFit as input. We observe that, except for under the noise level 2.5%, our method (PCA) produces the second-best results on CD and P2S metrics. Moreover, our method (AdaFit) achieves the best results on both metrics in most cases, which show the superiority of our method.

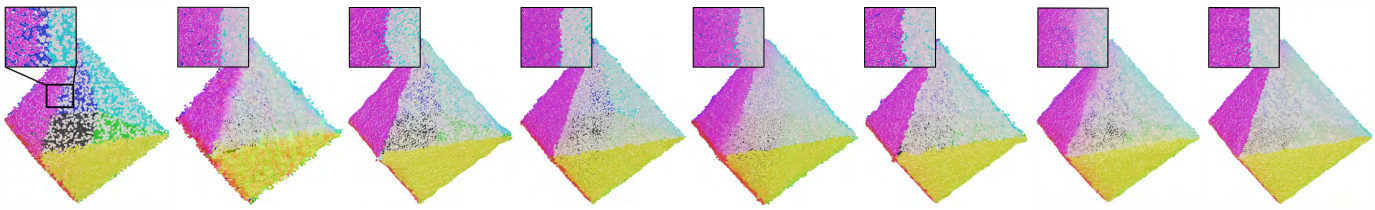
Computational time. We also record the running time

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3292464



(a) Real scanned outdoor scene (b) Noisy (c) WLOP (d) RIMLS (e) EC-Net (f) DMR (g) PCN (h) PF (i) Ours

Fig. 15: Denoising results of a real scanned point cloud scene. The zoomed views highlight that our method better removes heavy noise and avoids introducing additional artifacts.



(a) Noisy (b) WLOP (c) RIMLS (d) EC-Net (e) DMR (f) PCN (g) PF (h) Ours

Fig. 16: Denoising results of irregular sampling data. From left to right: noisy input, results produced by WLOP, RIMLS, EC-Net, DMR, PCN, PF, and our method.

TABLE 1: Quantitative evaluation of results in Figs. 10, 11, 12, and 13. For each result, we list CD ($\times 10^{-5}$) and P2S ($\times 10^{-3}$) errors and running time (in seconds).

Model	WLOP	RIMLS	EC-NET	DMR	PCN	PF	Ours
PartLp	18.09, 10.10; 89.0	2.78, 2.66; 708.2	4.15, 5.96; 51.1	3.32, 3.18; 14.3	1.84, 1.34; 704.0	1.02, 1.02; 152.1	**0.93**, **0.96**; 311.1
Boxunion	4.80, 3.71; 94.7	2.57, 2.31; 669.3	4.29, 5.93; 49.8	2.98, 3.11; 20.6	1.98, 1.94; 726.0	1.29, 1.11; 170.2	**1.21**, **1.06**; 303.5
Elephant	35.10, 8.20; 107.3	11.42, 5.40; 551.7	9.83, 4.73; 52.2	7.01, 5.41; 14.7	3.65, 3.11; 726.5	4.48, 2.86; 163.9	**2.87**, **2.78**; 310.9
Bunny_Hi	1.29, 1.82; 56.2	1.06, 1.64; 187.7	0.90, 2.16; 52.2	1.87, 2.40; 15.2	1.03, 1.32; 736.1	0.81, 1.00; 155.4	**0.70**, **0.82**; 311.3
Cone	20.97, 7.50; 10.7	18.04, 5.73; 27.2	19.74, 7.17; 6.7	28.19, 6.90; 2.4	19.47, 6.92; 47.9	17.54, **5.71**; 29.5	**17.08**, **5.71**; 34.1
Pyramid	20.18, 6.60; 9.1	17.26, 5.77; 21.8	19.86, 7.04; 8.6	22.21, 7.07; 2.8	20.16, 6.84; 56.3	18.20, 5.90; 20.4	**17.17**, **5.66**; 45.1
David	15.78, 5.56; 13.8	19.03, 5.45; 40.0	15.27, 6.02; 9.7	26.47, 6.96; 2.9	15.30, 5.87; 67.9	16.37, 5.51; 26.2	**15.01**, **5.41**; 49.1

of all the tested methods in Table 1. As seen, DMR is the fastest method, while PCN is the slowest one. Our method ranks fourth in speed among the learning-based methods. In particular, the traditional methods (WLOP, RIMLS) require trial-and-error efforts to tune parameters to produce satisfactory results in practice; thus, we only discuss the inference time of the deep learning methods (EC-Net, DMR, PCN, PF, and ours). DMR and EC-Net are faster than the other deep learning methods (PCN, PF, and ours). The reason is that DMR and EC-Net have an analogous upsampling process and only use a few patches of the noisy point cloud as input. In contrast, the PCN, PF, and our method use a pointwise manner to deal with the noisy input; thus, the inference times of these three methods are higher. Moreover, due to the network complexity, our method takes less time than the PCN and more time than the PF. In summary, although the inference time of our method seems to be slightly computationally intensive, it can generate more appealing results in terms of visual quality and error metrics in most examples.

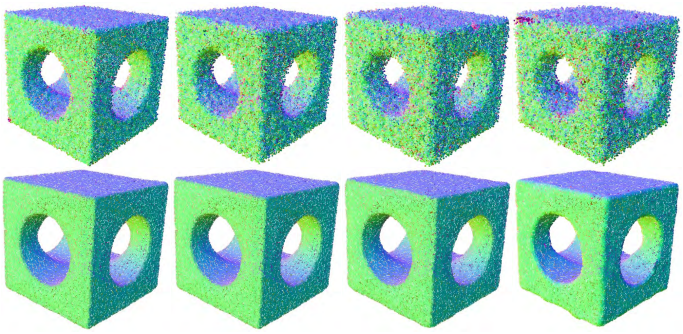
4.3 Experiments for Normal Filtering

The normal filtering task also plays an important role in our method. We compare our method with state-of-the-art normal filtering methods, including the traditional method of Jet [58] and the deep learning methods of PCPNet [47], DeepFit [51] and AdaFit [52]. We retrain PCPNet, DeepFit, and AdaFit over our training set. For PCPNet and AdaFit,

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3292464

TABLE 2: Quantitative comparisons for learning-based denoising methods on the synthetic dataset. Bold represents the best results, and underline denotes the second-best results.

	CD ($\times 10^{-5}$), P2S ($\times 10^{-3}$)					
Noise Level	EC-Net	DMR	PCN	PF	Ours (PCA)	Ours (AdaFit)
0.25%	0.67, 0.88	2.12, 2.16	0.96, 1.15	0.70, 0.55	0.61, 0.52	**0.60**, **0.43**
0.5%	1.08, 1.61	2.33, 2.32	1.33, 1.48	0.92, 0.78	0.80, **0.65**	**0.78**, 0.68
1%	4.47, 4.56	3.49, 3.27	2.11, 2.25	1.62, 1.36	1.31, 1.30	**1.24**, **1.24**
1.5%	12.90, 8.38	5.70, 4.90	3.73, 3.54	2.77,2.32	2.76, 2.05	**1.86**, **1.88**
2.5%	51.05, 16.66	19.96, 9.97	20.16, 8.92	5.11, 3.40	5.62, 3.73	**4.96**, **3.38**



(a) 1% (b) 2% (c) 3% (d) 4%

Fig. 17: Denoising results of Cylinder corrupted by different levels of noise. The first row shows noisy point clouds (1%, 2%, 3%, and 4% noise), while the second row shows the corresponding denoising results produced by our method.

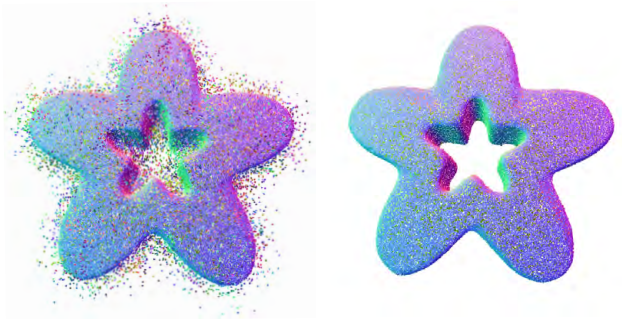


Fig. 18: Denoising result of Trim_star with outliers. Noisy input (left) and the corresponding denoising result (right).

we retrain the multi-scale version of the network provided by the authors.

Qualitative comparisons. Fig. 19 visualizes the error map for normals, where the error is defined as the angular deviations between the filtered normals and the ground truth normals. As we can see, Jet and PCPNet produce higher errors in geometric feature regions. DeepFit can deal with smoothly curved regions well, but it blurs small-scale features. AdaFit effectively recovers small-scale features and fine details but slightly oversmoothes sharp features. In contrast, our method produces the desired results in terms of preserving sharp and detailed features. To test the generalization capability of our method, we also present Fig. 20 for comparisons between our method and four other methods applied to real indoor scenes acquired by Kinect sensors (NYU Depth V2 dataset [57]). Note that we do not train all the tested methods on scanned scenes. For a fair comparison, we only use the filtered normals produced by our method instead of retaining denoised point coordinates. As we can see in Fig. 20, Jet retains considerable noise in the results. PCPNet can smooth noisy surfaces effectively but flatten medium- and small-scale geometric features. DeepFit and AdaFit preserve geometric features well even for small-scale and detailed features, although they induce some bumps in the results. The reason for inducing the bumpy artifacts may be as follows. Since both noise and geometric details are high-frequency information, DeepFit and Adafit may erroneously restore some high level noise as geometric features. Compared to DeepFit and AdaFit, our method can better preserve structure features, although some geometric details are flattened slightly. Moreover, our method tends to produce visually cleaner results without noticeable artifacts.

TABLE 3: Quantitative comparisons for normal estimation for classical geometric and learning-based methods. Bold represents the best results, and underline denotes the second-best results.

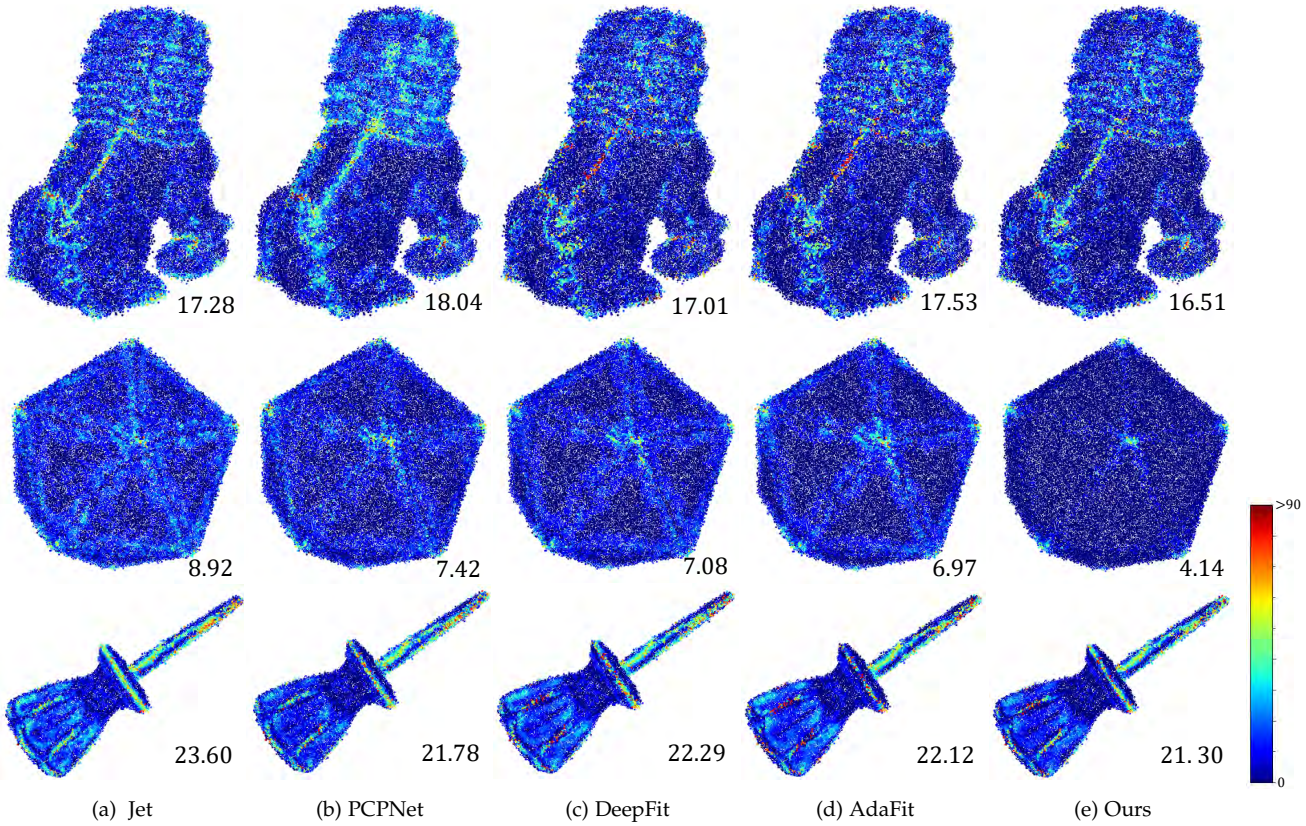
	RMSE ($\times 10^{-3}$)						
Noise level	PCA	Jet	PCPNet	DeepFit	AdaFit	Ours (PCA)	Ours (AdaFit)
0.25%	13.48	13.10	12.59	11.69	11.09	10.97	**10.68**
0.5%	16.54	16.29	15.48	15.01	14.92	14.78	**14.27**
1%	25.82	22.96	21.72	21.54	21.15	21.38	**19.36**
1.5%	32.87	28.58	26.66	27.25	25.76	26.04	**25.41**
2.5%	42.95	37.97	35.86	36.29	32.81	33.32	**31.67**

Quantitative Comparisons. We adopt the root mean squared error of the angle difference (abbreviated as RMSE) to quantify normal filtering results [47], [52]. Lower RMSE values indicate better results. The results of our method (PCA) have the lowest RMSE values for all the tested examples, as shown in Fig. 19. We also compare our method with the other four on the test set introduced in subsection 4.1. Our method (PCA) can produce the second-best results when the noise is low and moderate (0.25%, 0.5% noise). AdaFit achieves the second-best results as the noise level increases (1.0%, 1.5%, and 2.5% noise), but our results are comparable to those of AdaFit. Furthermore, by using better initial normals as input, our method (AdaFit) can significantly outperform the other compared methods and achieves the smallest RMSE values at all noise levels, as shown in Table 3.

4.4 PCDNF versus DR

To further demonstrate the effectiveness of PCDNF, we compare it to DR, a method recently proposed by Chen et

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3292464



(a) Jet (b) PCPNet (c) DeepFit (d) AdaFit (e) Ours

Fig. 19: Visualization of normal error for synthetic data with 0.5% noise. From left to right: Jet, PCPNet, DeepFit, AdaFit, and our method. The point clouds are color-coded based on angular difference, with a color map given by the color bar on the right. The numerical value denotes RMSE ($\times 10^{-3}$), and a lower error is better.

TABLE 4: Quantitative comparisons of DR, CL, and our PCDNF. We list CD and P2S for the denoising task and RMSE for the normal filtering task.

	CD ($\times 10^{-5}$), P2S ($\times 10^{-3}$); RMSE ($\times 10^{-3}$)		
Noise Level	DR	CL	Ours (PCA)
0.25%	1.03, 0.54; –	0.64, **0.48**; 12.65	**0.61**, 0.52; **10.97**
0.5%	1.28, 0.82; –	0.85, 0.73; 15.51	**0.80**, **0.65**; **14.78**
1%	1.92, 1.46; –	1.36, 1.32; 21.40	**1.31**, **1.30**; **21.38**
1.5%	3.22, 2.31; –	2.89, 2.54; **25.45**	**2.76**, **2.05**; 26.04
2.5%	10.66, 6.12; –	6.27, 4.14; **32.14**	**5.62**, **3.73**; 33.32

TABLE 5: Ablation analysis: quantitative comparisons of different network versions. For each version, we list CD for the denoising task and RMSE for the normal filtering task.

	CD ($\times 10^{-5}$), RMSE ($\times 10^{-3}$)				
Noise Level	*V1*	*V2*	*V3*	*V4*	Full
0.25%	0.64, 15.27	0.62, 13.42	0.63, –	–, **10.95**	**0.61**, 10.97
0.5%	0.87, 18.15	0.81, 16.30	0.82, –	–, 15.18	**0.80**, **14.78**
1%	1.52, 22.81	1.33, 21.58	1.39, –	–, 21.60	**1.31**, **21.38**
1.5%	2.99, 27.84	2.88, 26.62	2.80, –	–, 26.68	**2.76**, **26.04**
2.5%	7.23, 35.41	6.46, 34.53	6.40, –	–, 34.58	**5.62**, **33.32**

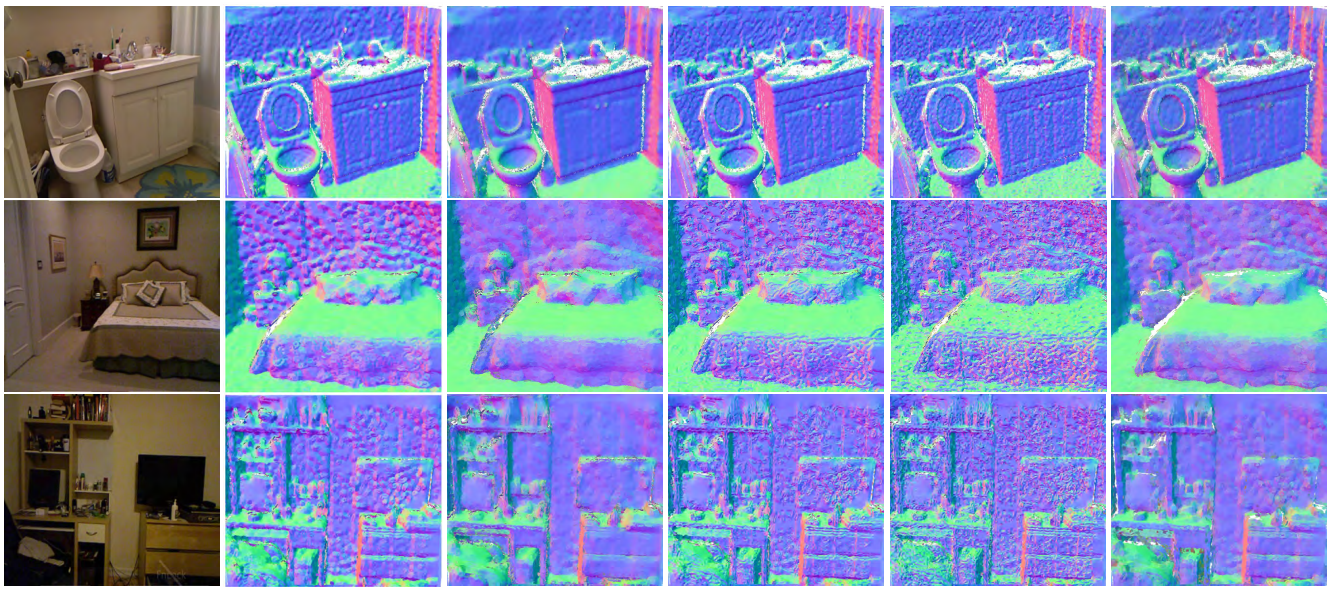
al. [39]. DR uses gradient fields to model the distribution of degraded point clouds, which can predict the gradient field over the point cloud that converges points toward the underlying surface. Quantitative comparisons of results produced by DR and our method are listed in Table 4. As we can see, our method yields lower CD and P2S values than the competing method DR for the denoising task, indicating that our results are more faithful to the ground truth.

4.5 PCDNF versus CL

During the final stages of the paper's completion, we became aware of a concurrent work proposed by Edirimuni et al. [40], which also addresses the challenges of point cloud denoising and normal filtering. Although both methods share similar goals, they differ in approaches: CL uses a contrastive learning mechanism to tackle both tasks, whereas PCDNF takes a multitask perspective. Specifically, PCDNF utilizes two network branches for denoising and normal filtering, respectively. This design allows the two tasks to benefit each other mutually. In contrast, CL uses only a single network branch to predict the combined vector (point position offset and normal vector) for denoising points and smoothing normals. Therefore, CL does not consider the association and interaction of the two tasks (denoising and normal filtering), which is the main difference between our method and CL.

Table 4 presents the evaluation results of our method and CL using the CD and RMSE metrics to assess the performance of denoising and normal filtering. We observe that our method outperforms CL in the denoising task for most cases except for the 0.25% noise level. For the normal

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3292464



(a) RGB image (b) Jet (c) PCPNet (d) DeepFit (e) AdaFit (f) Ours

Fig. 20: Visual comparison of normal estimation results on scanned point clouds from the NYU Depth V2 dataset [57]. From left to right: RGB image (noisy input), Jet, PCPNet, DeepFit, AdaFit, and our method.

filtering task, our method also achieves better results at the noise levels 0.25%, 0.5%, and 1.0%. However, CL achieves lower MSAE values when the noise level is high, specifically at 1.5% and 2.5%. Therefore, we believe that PCDNF and CL are complementary approaches, each offering unique advantages within their respective paradigms.

4.6 Ablation Studies

We verify the individual contributions of the major modules in our network by conducting the following four ablation studies.

- *Removing feature selection and refinement modules (V1).*
- *Removing feature refinement module (V2).*
- *The normal filtering network branch is removed (V3).*
- *The denoising network branch is removed (V4).*

For each ablated variant, we evaluate it in the validation set to choose the best hyperparameters. We perform quantitative comparisons of our method and four ablated variants in the test set and record the evaluation in Table 5. From the table, we have the following observations. All four variants show lower performance than our full pipeline. Each pipeline module is necessary to ensure high-quality denoising and normal filtering results. More specifically, by comparing *V1* with *V2*, it can be seen that the feature selection module is necessary for removing noise by selecting those feature points with similar characteristics to the denoised point. By comparing *V2* with our full pipeline, we can see that the additional feature refinement module positively impacts denoising accuracy. The feature refinement module consists of two units (feature augmentation and fusion). The roles of these two units are demonstrated in Figs. 8 and 9, and are explained in subsection 3.2.3. We use a two-branch network structure for point cloud denoising and normal filtering tasks. To demonstrate the positive interaction between these two tasks, we design two variants (*V3* and *V4*) that perform only the denoising task and normal estimation task. As we can see, our full network can achieve the best quantitative results in terms of CD and RMSE. Specifically, comparing variant *V3* with our full network, our additional normal filtering branch helps our method produce the best denoising results. Comparing variant *V4* with our full network, our denoising branch can help our method yield the best normal filtering results. The above ablation studies confirm the effectiveness of the major modules in our network and the mutual promotion of the two tasks (point cloud denoising and normal filtering).

4.7 Limitations

To some extent, the quality of the initial normals has an impact on the results of our method. Our method (with PCA) outperforms AdaFit at lower noise levels but is less effective at higher noise levels, as Table 3 shows. The reason for this is that our method prioritizes denoising and treats normal estimation as a secondary task. Consequently, our subnet for smoothing raw normals lacks the complexity needed to accurately estimate normals in the presence of large noise. In order to maintain the simplicity and generality of input, we choose to use raw normals estimated by PCA, which can negatively affect the performance of our method in the presence of high noise levels. Nonetheless, due to our multitask and iteration mechanisms, our method can incrementally enhance denoising and normal filtering results, making them comparable to state-of-the-art methods. Furthermore, when better initial normals (estimated by AdaFit) are used as input, our method can concurrently achieve the best denoising and normal filtering results, as demonstrated in Tables 2 and 3.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3292464

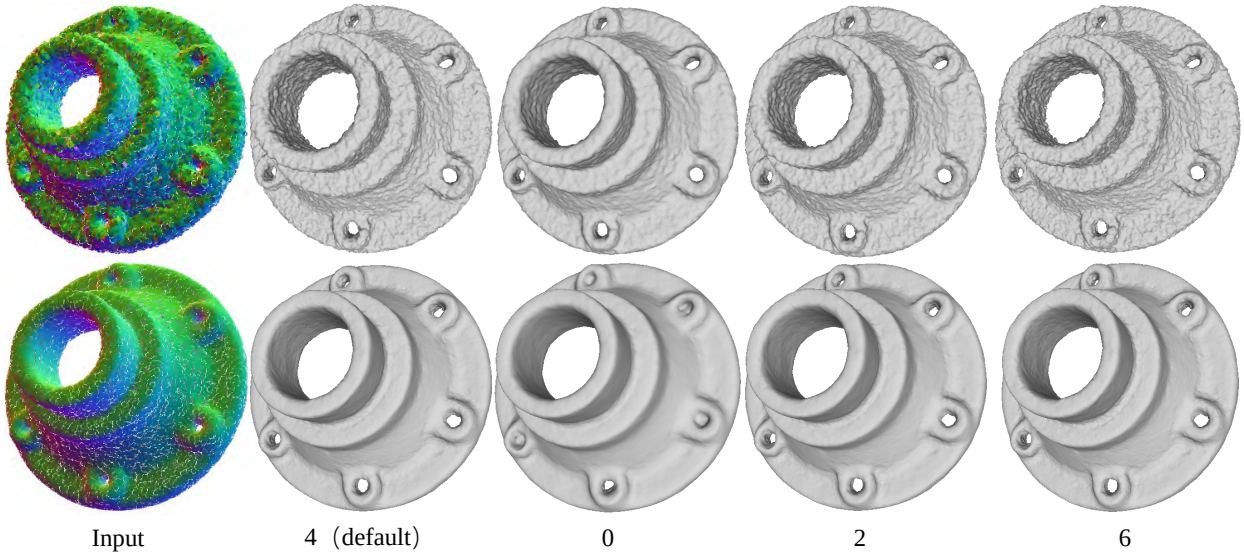


Fig. 21: The screened Poisson surface reconstruction (sPSR) algorithm balances smoothness and accuracy via an interpolation weight α. The top row shows the surfaces produced by applying sPSR directly to the noisy input with weight $\alpha = 4$, 0, 2, 6. The second row shows the reconstructed surfaces, which are computed by applying sPSR (with the same interpolation weight as the top row) to the denoised input. Our reconstructed surfaces are not sensitive to the perturbation of α.

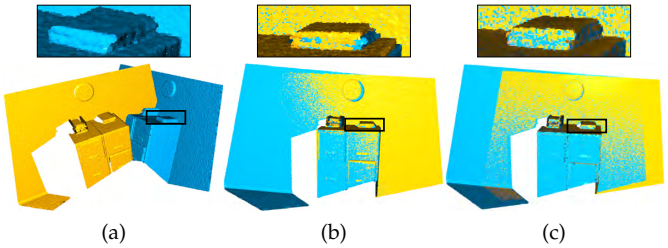


Fig. 22: Application of point cloud registration. (a) Noisy input. (b) Registration result of (a). (c) Registration result after applying our denoising method to (a).

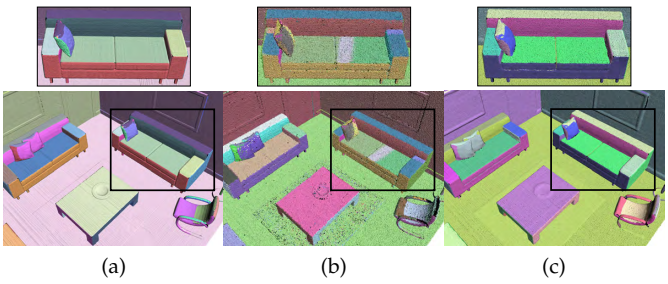


Fig. 23: Application of RANSAC plane fitting. (a) Clean point cloud. (b) The corresponding noisy point cloud. (c) Plane segmentation result produced after applying our method to (b).

4.8 Applications

To demonstrate the effectiveness of our method across various applications, we utilize our denoised outputs as inputs for surface reconstruction [41], point cloud registration [59], and plane fitting [60] tasks. In Fig. 21, we observe that the reconstruction results from noisy point clouds exhibit significant artifacts, whereas those from our denoised input are high-quality and preserve features. Fig. 22 demonstrates that our denoising method can enhance the accuracy of the point cloud registration task, as outputs after applying our method are more precise than those from raw inputs. We perform plane fitting [60] on an indoor scene point cloud. As Fig. 23 shows, using our denoised output as input for the RANSAC algorithm [60] results in a more reliable plane segmentation outcome, which is closer to the segmentation outcome from the corresponding clean point cloud.

5 Conclusion

We have proposed a learning method for denoising point clouds via joint normal filtering. Our key insight is that denoising and normal filtering tasks are inseparably intertwined. Our method takes the noisy point cloud and corresponding initial normals as input and uses an end-to-end approach to predict denoised points and filtered normals simultaneously. Our method comprises two innovative modules: the shape-aware selector and feature refinement. The shape-aware selector reduces the negative effects of noise and outliers on feature learning, thereby enhancing denoising and filtering performance. The feature refinement has advantages in recovering structure and detailed features. Our experiments demonstrate that incorporating normal filtering improves the denoising performance significantly. Our method achieves a new SOTA for the denoising task, with substantial improvements in visual quality and quantitative evaluation. Although our method is not specifically designed for normal filtering, it performs favorably against most SOTA normal filtering methods.

To our knowledge, this is the first work to couple the interdependent tasks of point cloud denoising and normal filtering within a single deep neural network. We believe that there is a wealth of opportunity for exploring future directions. For instance, we can improve the normal filtering

subnet to obtain more accurate normal estimations, especially in situations with large noise. Moreover, we can further improve the subnet to address the normal orientation ambiguity problem. As a pointwise denoising approach, our method incurs high computational costs for training and inference. To address this, we plan to develop a patchwise framework to improve runtime performance.

ACKNOWLEDGMENTS

This work was supported by National Key R&D Program of China (2022YFB3904100), NSF of China (62072422, 62076227), NSF of Anhui Province (2008085MF195), Collaborative Innovation Center for Natural Resources Planning and Marine Technology of Guangzhou (2023B04J0301), Key-Area Research and Development Program of Guangdong Province (2020B0101130009), and the Ministry of Education, Singapore, under its Academic Research Fund Grants (MOE-T2EP20220-0005 & RT19/22).

REFERENCES

[1] M. Wei, H. Chen, Y. Zhang, H. Xie, Y. Guo, and J. Wang, "GeoDualCNN: Geometry-supporting dual convolutional neural network for noisy point clouds," *IEEE Trans. Vis. Comput. Graph.*, 2021.

[2] 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. Vision*, vol. 130, no. 3, pp. 615–629, 2022.

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

[4] A. C. Öztireli, G. Guennebaud, and M. Gross, "Feature preserving point set surfaces based on non-linear kernel regression," *Comput. Graph. Forum*, vol. 28, no. 2, pp. 493–501, 2009.

[5] 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. Graph.*, vol. 24, no. 8, pp. 2315–2326, 2017.

[6] H. Chen, M. Wei, Y. Sun, X. Xie, and J. Wang, "Multi-patch collaborative point cloud denoising via low-rank recovery with graph constraint," *IEEE Trans. Vis. Comput. Graph.*, vol. 26, no. 11, pp. 3255–3270, 2019.

[7] H. Chen, J. Huang, O. Remil, H. Xie, J. Qin, Y. Guo, M. Wei, and J. Wang, "Structure-guided shape-preserving mesh texture smoothing via joint low-rank matrix recovery," *Comput-Aided Des.*, vol. 115, pp. 122–134, 2019.

[8] X. Lu, S. Schaefer, J. Luo, L. Ma, and Y. He, "Low rank matrix approximation for 3d geometry filtering," *IEEE Trans. Vis. Comput. Graph.*, vol. 28, no. 4, pp. 1835–1847, 2020.

[9] Z. Liu, X. Xiao, S. Zhong, W. Wang, Y. Li, L. Zhang, and Z. Xie, "A feature-preserving framework for point cloud denoising," *Comput-Aided Des.*, vol. 127, p. 102857, 2020.

[10] Z. Liu, Y. Li, W. Wang, L. Liu, and R. Chen, "Mesh total generalized variation for denoising," *IEEE Trans. Vis. Comput. Graph.*, vol. 28, no. 12, pp. 4418–4433, 2022.

[11] 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.

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

[13] 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.

[14] L. Zhou, G. Sun, Y. Li, W. Li, and Z. Su, "Point cloud denoising review: from classical to deep learning-based approaches," *Graph. Models*, vol. 121, p. 101140, 2022.

[15] 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. Graph.*, vol. 9, no. 1, pp. 3–15, 2003.

[16] N. Amenta and Y. J. Kil, "Defining point-set surfaces," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 264–270, 2004.

[17] S. Fleishman, D. Cohen-Or, and C. T. Silva, "Robust moving least-squares fitting with sharp features," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 544–552, 2005.

[18] Y. Lipman, D. Cohen-Or, D. Levin, and H. Tal-Ezer, "Parameterization-free projection for geometry reconstruction," *ACM Trans. Graph.*, vol. 26, no. 3, p. 22, 2007.

[19] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or, "Consolidation of unorganized point clouds for surface reconstruction," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 176:1–7, 2009.

[20] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. R. Zhang, "Edge-aware point set resampling," *ACM Trans. Graph.*, vol. 32, no. 1, pp. 9:1–9:12, 2013.

[21] R. Preiner, O. Mattausch, M. Arikan, R. Pajarola, and M. Wimmer, "Continuous projection for fast L_1 reconstruction." *ACM Trans. Graph.*, vol. 33, no. 4, pp. 47–1, 2014.

[22] H. Avron, A. Sharf, C. Greif, and D. Cohen-Or, "ℓ_1-sparse reconstruction of sharp point set surfaces," *ACM Trans. Graph.*, vol. 29, no. 5, pp. 135:1–12, 2010.

[23] Y. Sun, S. Schaefer, and W. Wang, "Denoising point sets via ℓ_0 minimization," *Comput. Aided Geom. Des.*, vol. 35, pp. 2–15, 2015.

[24] E. Mattei and A. Castrodad, "Point cloud denoising via moving RPCA," *Comput. Graph. Forum*, vol. 36, no. 8, pp. 123–137, 2017.

[25] J. Digne, S. Valette, and R. Chaine, "Sparse geometric representation through local shape probing," *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 7, pp. 2238–2250, 2017.

[26] J. Wang, J. Jiang, X. Lu, and M. Wang, "Rethinking point cloud filtering: A non-local position based approach," *Comput-Aided Des.*, vol. 144, p. 103162, 2022.

[27] G. Sun, C. Chu, J. Mei, W. Li, and Z. Su, "Structure-aware denoising for real-world noisy point clouds with complex structures," *Comput-Aided Des.*, vol. 149, p. 103275, 2022.

[28] W. Hu, X. Gao, G. Cheung, and Z. Guo, "Feature graph learning for 3d point cloud denoising," *IEEE Trans. Signal Process.*, vol. 68, pp. 2841–2856, 2020.

[29] 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 (CVPR)*, 2017, pp. 652–660.

[30] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2017, pp. 5100–5109.

[31] S.-L. Liu, H.-X. Guo, H. Pan, P.-S. Wang, X. Tong, and Y. Liu, "Deep implicit moving least-squares functions for 3d reconstruction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 1788–1797.

[32] D. de Silva Edirimuni, X. Lu, Z. Shao, G. Li, A. Robles-Kelly, and Y. He, "IterativePFN: True iterative point cloud filtering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[33] 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.

[34] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "EC-Net: an edge-aware point set consolidation network," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 386–402.

[35] S. Luo and W. Hu, "Differentiable manifold reconstruction for point cloud denoising," in *Proceedings of the ACM International Conference on Multimedia (ACM MM)*, 2020, pp. 1330–1338.

[36] P. Hermosilla, T. Ritschel, and T. Ropinski, "Total Denoising: Unsupervised learning of 3d point cloud cleaning," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 52–60.

[37] F. Pistilli, G. Fracastoro, D. Valsesia, and E. Magli, "Learning graph-convolutional representations for point cloud denoising," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020, pp. 103–118.

[38] S. Luo and W. Hu, "Score-based point cloud denoising," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021, pp. 4583–4592.

[39] H. Chen, B. Du, S. Luo, and W. Hu, "Deep point set resampling via gradient fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 2913–2930, 2023.

[40] D. de Silva Edirimuni, X. Lu, G. Li, and A. Robles-Kelly, "Contrastive learning for joint normal estimation and point cloud filtering," *IEEE Trans. Vis. Comput. Graph.*, 2023.

[41] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 3, pp. 1–13, 2013.

[42] F. Hou, C. Wang, W. Wang, H. Qin, C. Qian, and Y. He, "Iterative Poisson surface reconstruction (iPSR) for unoriented points," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 128:1–13, 2022.

[43] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3d registration," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 3212–3217.

[44] J. Zhang, Y. Yao, and B. Deng, "Fast and robust iterative closest point," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, 2022.

[45] 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.

[46] J. Zhou, H. Huang, B. Liu, and X. Liu, "Normal estimation for 3d point clouds via local plane constraint and multi-scale selection," *Comput-Aided Des.*, vol. 129, p. 102916, 2020.

[47] 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 (CVPR)*, 2019, pp. 10 112–10 120.

[48] J. Zhou, W. Jin, M. Wang, X. Liu, Z. Li, and Z. Liu, "Fast and accurate normal estimation for point clouds via patch stitching," *Comput.-Aided Des.*, vol. 142, p. 103121, 2022.

[49] Y. Ben-Shabat and S. Gould, "DeepFit: 3d surface fitting via neural network weighted least squares," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020, pp. 20–34.

[50] 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 International Conference on Computer Vision (ICCV)*, 2021, pp. 6118–6127.

[51] J. Zhang, J.-J. Cao, H.-R. Zhu, D.-M. Yan, and X.-P. Liu, "Geometry guided deep surface normal estimation," *Comput.-Aided Des.*, vol. 142, p. 103119, 2022.

[52] J. Cao, H. Zhu, Y. Bai, J. Zhou, J. Pan, and Z. Su, "Latent tangent space representation for normal estimation," *IEEE Trans. Ind. Electron.*, vol. 69, no. 1, pp. 921–929, 2021.

[53] 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 Trans. Pattern Anal. Mach. Intell.*, 2022.

[54] R. Sharma, T. Schwandt, C. Kunert, S. Urban, and W. Broll, "Point cloud upsampling and normal estimation using deep learning for robust surface reconstruction," *arXiv:2102.13391*, 2021.

[55] 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 (CVPR)*, 2016, pp. 770–778.

[56] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012, pp. 746–760.

[57] F. Cazals and M. Pouget, "Estimating differential quantities using polynomial fitting of osculating jets," *Comput. Aided Geom. Des.*, vol. 22, no. 2, pp. 121–146, 2005.

[58] Q.-Y. Zhou, J. Park, and V. Koltun, "Fast global registration," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016, pp. 766–782.

[59] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for point-cloud shape detection," *Comput. Graph. Forum*, vol. 26, no. 2, pp. 214–226, 2007.

Zheng Liu is currently an associate professor at China University of Geosciences (Wuhan). He received the Ph.D. degree from Central China Normal University in 2012. From 2013 to 2014, he held a post-doctoral position with School of Mathematical Sciences, University of Science and Technology of China. His research interests include geometry processing, computer graphics, 3D computer vision, and deep learning.

Yaowu Zhao is currently a M.S. candidate in China University of Geosciences (Wuhan). He received the B.S. degree from China University of Geosciences (Wuhan), in 2021. His research interests include geometry processing and deep learning.

Sijing Zhan is currently a M.S. candidate at China University of Geosciences (Wuhan). He received B.S. degree from Jimei University, in 2020. Her research interests include geometry processing, 3D vision and deep learning.

Yuanyuan Liu received the B.E. degree from Nanchang University, Nanchang, China, in 2005, the M.E. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2007, and the Ph.D. degree from Central China Normal University, Wuhan. She is an associate professor at the China University of Geosciences, Wuhan. Her research interests include image processing, computer vision, and pattern recognition.

Renjie Chen is a professor at the University of Science and Technology of China (USTC). He holds a PhD degree from Zhejiang University, China. Before joining USTC, he was a postdoctoral fellow at the Technion–Israel Institute of Technology, a postdoctoral research associate at the University of North Carolina at Chapel Hill, a key researcher in the BeingThere Center in Nanyang Technological University, Singapore, and a senior researcher heading a research group working on 3D geometry and images at the Max Planck Institute for Informatics (MPII) in Saarbrucken, Germany. His research interests includes computer graphics, geometry modeling, computational geometry and glasses-free 3D display.

Ying He is currently an associate professor at School of Computer Science and Engineering, Nanyang Technological University, Singapore. He 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. 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.