

Structure-Aware Denoising for Real-world Noisy Point Clouds with Complex Structures

Guoxing Sun^a, Chao Chu^a, Jialin Mei^a, Weiqing Li^b, Zhiyong Su^{a,*}

^a*School of Automation, Nanjing University of Science and Technology, Nanjing, Jiangsu Province 210094, P.R. China*

^b*School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, Jiangsu Province 210094, P.R. China*

Abstract

Point cloud denoising is a crucial and fundamental step in geometry processing, which has achieved significant progress in the last two decades. Denoising real-world noisy point clouds is a very challenging problem since it is hard to describe the complex real-world noise by simple distributions such as Gaussian distribution. Furthermore, existing methods may suffer from performance degradation when dealing with real-world noisy point clouds with complex structures, which contain not only sharp features (sharp edges, sharp corners, etc.) but also smooth features, fine features, etc. To solve the above-mentioned problems, we propose a novel structure-aware denoising approach by exploiting the prior information in both external clean point clouds and the given noisy point cloud. We first group nonlocal self-similarity (NSS) patches from a set of external clean point clouds. Then, we employ the Gaussian Mixture Model (GMM) learning algorithm to learn external NSS priors over patch groups. Next, the internal priors are learned from the given noisy point cloud in the same way to refine the prior model. We integrate both the learned external and internal priors into a set of orthogonal dictionaries to efficiently estimate point normals. Finally, we propose a feature-aware point updating method through adaptive

*Corresponding author

Email addresses: gx_sun@njjust.edu.cn (Guoxing Sun), 979953220@qq.com (Chao Chu), 1904329513@qq.com (Jialin Mei), li_weiqing@njjust.edu.cn (Weiqing Li), su@njjust.edu.cn (Zhiyong Su)

neighborhood selection to reposition points to match the estimated normals. Extensive experiments show that our approach achieves favorable comprehensive performance compared with many popular or state-of-the-art methods in terms of both objective and visual perception. The source code can be found at <https://zhiyongsu.github.io>.

Keywords: Point cloud denoising, nonlocal self-similarity, real-world noise, gaussian mixture model, point updating.

1. Introduction

Point cloud denoising is a crucial and fundamental research area in computer graphics. It seeks to remove unwanted noise from a given noisy input while preserving structures prominently. Recently, the rapid development of geometric sensing techniques, such as laser scanning, time-of-flight range finding, etc., has witnessed the wide use of geometric sensors. Point clouds acquired with these sensors inevitably suffer from some level of noise and outliers caused by measurement error [1]. Hence, denoising is in great demand for subsequent geometry processing applications, such as segmentation [2, 3], recognition [4, 5], reconstruction [6, 7, 8], etc.

Numerous point cloud denoising techniques have been proposed in the last two decades [1, 9, 10]. Current state-of-the-art denoising algorithms can achieve impressive results in synthetic data [11, 12]. However, the comprehensive performance of existing methods may degrade when facing real-world noisy point clouds with complex structures. Especially, there are still some open challenging problems to be addressed.

First of all, existing methods may suffer from performance degradation when dealing with point clouds with complex structures. The referred complex structures in this paper mean that the noisy point cloud contains not only sharp features (sharp edges, sharp corners, etc.), but also smooth features, fine features, etc. The desired denoising method should retain these inherently complex structures while removing unwanted noise. For example, as shown in Fig. 1,

the noisy point cloud comprises both sharp and round corners. Ideally, the denoised point cloud should simultaneously maintain these sharp and round corners. Unfortunately, existing denoising algorithms may fail when facing this challenging problem. For classical methods, MLS-based [13, 14, 15] and LOP-based [16, 17, 18] methods usually hold the local smoothness assumption, and thus are less able to preserve sharp features well. Sparse and low-rank methods [19, 20, 21, 22, 23, 24] can preserve sharp features well, but may over-sharpen smoothly curved features. For deep learning-based methods [25, 26, 11, 12], they crucially depend on expensive training over massive datasets, and are often impaired when the data to be denoised deviates significantly from the training set.

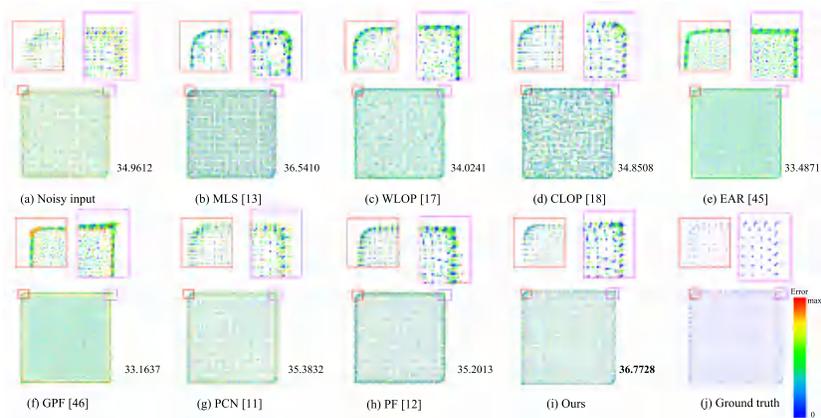


Figure 1: Comparison of denoising results on a noisy point cloud with both sharp and round corners corrupted by impulse noise. Existing denoising algorithms may suffer from performance degradation when facing this challenging problem. Our approach can maintain these inherent structures simultaneously.

Second, few denoising methods are specifically developed for real-world noisy point clouds in the literature. Existing methods primarily focus on the scenario of specific kinds of synthetic noise with different levels, such as Gaussian noise [27, 28, 29, 30, 31] and Poisson noise [32]. To cope with real-world noise, most of existing works choose to directly apply methods originally designed for synthetic noise into the real-world scenario. However, it is difficult to simulate complex

Table 1: Comparison of the comprehensive performance between the state-of-the-art point cloud denoising techniques and our method

Items Methods	Structural feature			Noise type
	Sharp feature	Fine feature	Smooth feature	
MLS [13]	*	**	**	**
LOP [16]/WLOP [17]/CLOP [18]	*	**	**	**
EAR [45]/GPF [46]	***	*	*	**
PCN [11]	**	***	**	*
Pointfliter [12]	***	**	**	**
Ours	***	***	***	***

40 real-world noise with a simple distribution of synthetic noise. Real-world noise is much more complicated and varies with changing scenarios and sensors [1, 33]. Therefore, their performance on real-world noise was unsurprisingly inferior to those on synthetic cases owing to the notable domain shift between synthetic and real-world noise.

45 Third, most of existing point updating methods, which aim to reposition each point to match estimated normal by analyzing the geometry structure of local neighborhood, are not good at restoring feature points, such as edge and corner points. Compared with the normal estimation in the point cloud denoising, point updating has received sparse treatment so far [21, 24, 31, 34, 35, 36, 37].
50 Previous algorithms perform point updating on a neighborhood centering in the point whose position is being estimated [38]. However, unfaithful position estimations will be introduced since the surface of noisy point clouds is not smooth everywhere, particularly the neighborhood of a point nearby the sharp feature area might span borders of different manifolds.

55 Nonlocal self-similarity (NSS) prior has been shown to produce promising results with regard to denoising and feature preserving in point cloud denoising [20, 22, 23, 24], especially in image denoising and restoration [39, 40, 41, 42, 43, 44]. Generally, they first cluster similar patches to form patch groups, and then apply structural sparse representation to achieve promising results for each
60 patch group [47]. Notably, existing NSS-based point cloud denoising solutions

just utilize the internal NSS priors learned from the input degraded (internal) point cloud [20, 22, 24, 23]. However, the internal NSS priors learned from the input noisy point cloud may not be accurate because of the interference noise. In contrast, learning a denoising model from external point clouds has not been
65 thoroughly explored. Recently, deep learning based denoising methods have been becoming popular [11, 12, 26, 25, 48, 49]. These methods usually assume that the training and testing data share the same distribution [33]. Furthermore, their performance heavily depends on high quality datasets which are desired to cover all of the real-world conditions [33]. All in all, internal priors learned from
70 the input noisy point cloud may be inaccurate, while external priors obtained from clean point clouds may not be adaptive to the input noisy point cloud.

To address the issues discussed above, leveraging the NSS priors of both the internal noisy point cloud and external clean point clouds jointly, we propose a novel structure-aware denoising method for real-world noisy point clouds with
75 complex structures. We first group nonlocal similar patches from an independent set of high quality external clean point clouds. Then, the Gaussian Mixture Model (GMM) is employed to learn external NSS priors over patch groups. After that, internal priors of the input noisy point cloud are also obtained in the same way. Both the learned internal and external priors are used to build a
80 set of orthogonal dictionaries to efficiently reconstruct the desired point cloud. Finally, we propose a feature-aware point updating algorithm to recover point positions to better match estimated normals. Our main contributions are as follows:

- We propose a novel structure-aware denoising approach for real-world
85 noisy point clouds with complex structures via jointly learning external and internal priors. To the best of our knowledge, this is the first denoising method that can remove real-world noise while preserving distinctive structural details specific to the given noisy point cloud.
- We propose a feature-aware point updating method through feature point
90 detection and adaptive neighborhood selection.

- Our proposed method has achieved the state-of-the-art comprehensive performance on real-world noisy point clouds with complex structures, as summarized in Table 1.

2. Related work

95 In this section, we first review noticeable point cloud denoising methods closely related to this work from the perspective of synthetic and real-world noise, respectively. Then, we introduce some related works on point updating.

2.1. Synthetic Noisy Point Cloud Denoising

Existing point cloud denoising methods mainly target for synthetic noise, 100 which can be classified into classical methods and deep learning-based methods. Classical point cloud denoising algorithms can be roughly divided into MLS-based, LOP-based, sparse and low-rank-based methods.

MLS-based methods [13, 14, 15] reconstruct a smooth surface from the input point cloud, and iteratively project input points onto the approximated 105 underlying surface. These methods can generate smooth surface robustly from extremely noisy point clouds, but are prone to over-smoothing.

LOP-based techniques [16, 17, 18, 45, 46] aim to describe the underlying surface by projecting the input point cloud to the target point cloud. Lipman et al. [16] proposed the locally optimal projection (LOP) operator, which iteratively 110 projects a subset of the noisy point cloud onto a reference point cloud (e.g., itself) to reduce noise. However, the LOP projection will become non-uniform when the input point cloud is highly uneven. Huang et al. [17] added the local adaptive density weight of each point to the LOP, namely WLOP, to form a uniformly distributed point set. Huang and Lu et al. [45, 46] further improved 115 the technique using anisotropic projection optimization. However, these methods rely on fitting local surfaces, so they can't be extended to different inputs, and the results are often too smooth or sharp after removing noise. More importantly, the smoothing assumption is invalid for some point clouds with sharp features in the real-world.

120 Sparse and low-rank-based methods [19, 20, 21, 22, 23, 24] treat the de-
noising task as an optimization problem and achieves denoising by solving the
optimization problem. Rosman et al. [20] proposed a patch cooperative spectral
point cloud denoising method. They searched similar patches by iterative clos-
est point (ICP) registration for collaborative denoising. Lu et al. [24] proposed
125 a method for estimating normals with low rank matrix approximation, while
using the obtained normals to update point positions for denoising. Chen et al.
[23] proposed a multi-patch cooperative denoising method. They filtered the
noise by applying graph constraints low-rank model to the height map obtained
by projection. However, these methods only considered the noisy model itself,
130 and artifacts may appear in some feature regions.

Recently, deep learning-based denoising methods have been widely applied
to noisy point clouds. PointProNet [25] converts disordered 3D point clouds
to height maps in 2D, using a CNN architecture network for denoising. EC-
Net [26] is suitable for point clouds with sharp features, but training EC-Net
135 requires manual labeling of sharp edges. PCN [11] introduces a network that
removes outliers and noisy points by merging with PCP-Net [50]. By introduc-
ing encoders and decoders, Zhang et al. [12] proposed an automatic and robust
denoising method. However, these methods are highly dependent on the com-
pleteness of the training dataset. Furthermore, it is also impractical to generate
140 high-quality training dataset that covers all of the real-world conditions.

Although the aforementioned state-of-the-art methods can achieve impres-
sive results on synthetic noise, they are either less able to denoise real-world
noisy point clouds effectively, or limited in their ability to handle point clouds
with complex structures.

145 2.2. Real-world Noisy Point Cloud Denoising

In the literature, few point cloud denoising methods are directly designed
for real-world noise. The majority of existing works apply methods originally
designed for synthetic noise into the real-world scenario directly. And, they
mostly utilize internal correlations (NSS priors) within an observed point cloud

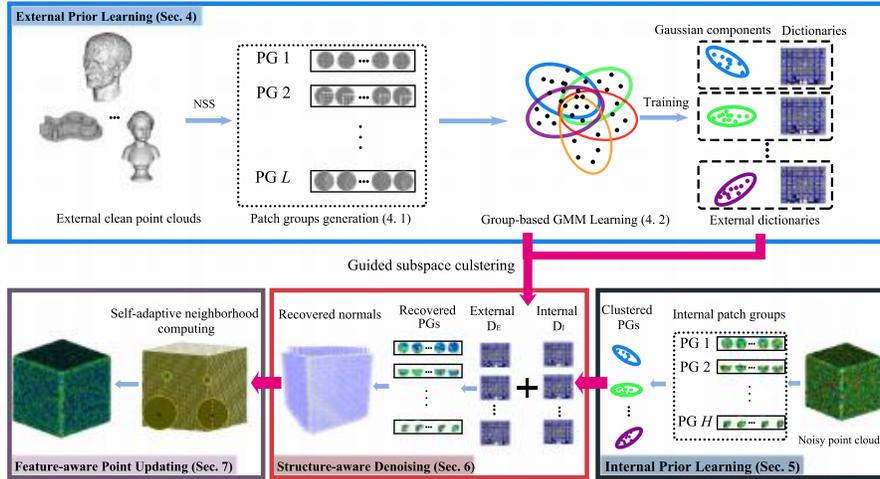


Figure 2: Flowchart of the proposed structure-aware denoising framework. In the external prior learning stage, we extract NSS-based patch groups (PGs) from a set of external clean point clouds. Then, we learn a finite GMM over clean point cloud PGs to describe the external NSS priors. In the internal prior learning stage, we apply the learned GMM model to guide the internal PG prior learning from a noisy point cloud. In the structure-aware denoising stage, we employ the learned external and internal dictionary elements to calculate the denoised normals. Finally, we employ a feature-aware point updating method through feature point calculation and adaptive neighborhood selection.

150 to remove real-world noise.

Hermsilla et al. [48] proposed an unsupervised denoising method that can be trained on noisy data without the ground truth. However, it cannot retain features due to the lack of clear feature information in the training stage. Luo et al. [51] proposed an autoencoder-like unsupervised neural network. They first 155 sampled the noisy point cloud and inferred the underlying manifold by feature embedding, then they obtained a denoised point cloud by resampling on the reconstructed manifold.

Though existing methods have shown excellent performance in removing real-world noise, how to exploit the NSS priors of both internal noisy point 160 cloud and external clean point clouds simultaneously is still an open problem.

2.3. Point Updating

Point updating refers to repositioning points according to estimated normals. Taubin et al. [34] first used point updating in mesh denoising and updating based on the idea that the surface formed by the neighborhood of the point should be as perpendicular to the normal as possible. Sun et al. [35] optimized [34] and accelerated computational efficiency. Sun et al. [21] applied point updating by limiting its point moving direction to normal direction. Finally, the point updating problem is converted into l_0 minimization problem. In [24], the restriction that the point can only move along the normal line is removed, making the updating results better. In order to reduce the ambiguity of the normal in the feature region, Zheng and Liu et al. [31, 36, 52] used multiple normals in the feature region to avoid errors in the normal direction. Yadav et al. [37] proposed to introduce constraints on the quadratic error metric based on different feature points and use distance-based constraints to update point positions. However, existing methods perform point updating on a neighborhood centering in the point whose position is being updated. When the point to be updated is on edges and other sharp feature area, it often leads to inaccurate position estimation.

3. Overview

Our structure-aware denoising framework consists of the following four modules: external prior learning, internal prior learning, structure-aware denoising, and feature-aware point updating, as shown in Fig. 2.

In the external prior learning stage, we first extract NSS-based patch groups (PGs) from an independent set of external high quality clean point clouds. Then, we learn a finite GMM over clean point cloud PGs to describe external NSS priors. Eigenvectors of the covariance matrix of the GMM are taken as elements of the external dictionary.

In the internal prior learning stage, we first generate PGs for the input noisy point cloud. Then, we assign each PG of the noisy point cloud to the most

190 suitable Gaussian component via maximizing a posterior probability, according
to the category information of the learned GMM.

In the structure-aware denoising stage, we first learn a hybrid orthogonal
dictionary by integrating both the external and internal priors. Then, we employ
the learned hybrid dictionary to calculate denoised normals while preserving
195 distinctive structural details specific to the given noisy point cloud.

In the feature-aware point updating stage, we reposition all points to match
the estimated normals through feature point detection and adaptive neighbor-
hood selection.

4. External Prior Learning

200 As discussed above, existing NSS-based point cloud denoising methods just
utilize the internal NSS priors from the input noisy point cloud. This section
presents a group-based GMM learning algorithm to learn external NSS priors
from the PGs of a set of external clean point clouds. The learned priors are then
used to recover the latent clean point cloud from the given noisy point cloud.

205 4.1. Patch Group Generation

We extract NSS-based PGs from a set of external clean point clouds to learn
external priors. A PG is defined as a group of similar patches to a local patch.

Given a clean point cloud $\mathbf{P} = \{\mathbf{p}_i, i = 1, 2, \dots, n\} \subset \mathbb{R}^{n \times 3}$, to extract the
local patch N_i of each point \mathbf{p}_i , we first employ the bilateral normal filter to
210 get its normal \mathbf{n}_i . Then, we find its d nearest neighbors to form the local patch
 $N_i = \{\mathbf{p}_{i,j}, j = 1, 2, \dots, d\}$, where $\mathbf{p}_{i,j}$ is the neighboring point of \mathbf{p}_i .

To extract NSS-based PGs for each local patch N_i , we first find W ($W > d$)
nearest neighbors of \mathbf{p}_i as the non-local searching range. Then, for each point
in the searching region, we find its d nearest neighbors as its local patch. After
215 that, we search the M most similar (overlapped) local patches of N_i to form its
PG \mathbf{Z}_i . Finally, we extract a number of L PGs from all external clean point
clouds.

To find those similar patches, we first denote the characteristics of each patch N_i as:

$$\begin{bmatrix} \mathbf{n}_{i,1} & \cdots & \mathbf{n}_{i,d} \end{bmatrix} = \begin{bmatrix} \mathbf{n}_{i,1}(x), & \cdots, & \mathbf{n}_{i,d}(x) \\ \mathbf{n}_{i,1}(y), & \cdots, & \mathbf{n}_{i,d}(y) \\ \mathbf{n}_{i,1}(z), & \cdots, & \mathbf{n}_{i,d}(z) \end{bmatrix}, \quad (1)$$

where $\mathbf{n}_{i,j}$ is the normal of $\mathbf{p}_{i,j}$. Then, we transform all the normal vectors to the local normal space of \mathbf{p}_i for comparison

$$\mathbf{B}_i = \mathbf{R}_i \begin{bmatrix} \mathbf{n}_{i,1} & \cdots & \mathbf{n}_{i,d} \end{bmatrix}, \quad (2)$$

where the rotation matrix \mathbf{R}_i is composed by the eigenvectors of the covariance matrix \mathbb{C}_i of \mathbf{p}_i , and \mathbb{C}_i is defined as:

$$\begin{aligned} \mathbb{C}_i &= \frac{1}{d} \sum_{j=1}^d (\mathbf{n}_j - \bar{\mathbf{n}}_i)(\mathbf{n}_j - \bar{\mathbf{n}}_i)^\top, \\ \mathbb{C}_i &= \mathbf{R}_i \mathbf{\Lambda}_i \mathbf{R}_i^\top, \end{aligned} \quad (3)$$

where $\bar{\mathbf{n}}_i$ is the mean normal vector of $\begin{bmatrix} \mathbf{n}_{i,1} & \cdots & \mathbf{n}_{i,d} \end{bmatrix}$, $\mathbf{\Lambda}_i$ is the eigenvalue of \mathbb{C}_i . After that, we define the Mahalanobis distance [53] between \mathbf{B}_i and \mathbf{B}_j as:

$$d(\mathbf{B}_i, \mathbf{B}_j) = \sqrt{(\bar{\mathbf{n}}_i - \bar{\mathbf{n}}_j)(\mathbb{C}_i + \mathbb{C}_j)^{-1}(\bar{\mathbf{n}}_i - \bar{\mathbf{n}}_j)^\top}, \quad (4)$$

where $\bar{\mathbf{n}}_i = \frac{1}{d} \sum_{j=1}^d \mathbf{R}_i \mathbf{n}_j$. Finally, the PG \mathbf{Z}_i is defined as:

$$\mathbf{Z}_i = \begin{bmatrix} \mathbf{z}_{i,1} & \cdots & \mathbf{z}_{i,M} \end{bmatrix}, \quad (5)$$

where $\mathbf{z}_{i,m} \in \mathbb{R}^{3d \times 1}$ is a patch vector denoted as:

$$\mathbf{z}_{i,m} = \begin{bmatrix} \mathbf{n}_{i,1}(x), \mathbf{n}_{i,1}(y), \mathbf{n}_{i,1}(z), \cdots, \mathbf{n}_{i,d}(x), \mathbf{n}_{i,d}(y), \mathbf{n}_{i,d}(z) \end{bmatrix}^\top. \quad (6)$$

4.2. Group-based GMM Learning

After obtaining L PGs from external clean point clouds, we learn a finite GMM over PGs $\{\mathbf{Z}_l\}_{l=1}^L$ of clean point clouds to describe the external NSS prior.

The GMM is a parameterized probability density function expressed as a weighted sum of the densities of the Gaussian components [54]. And, it can be defined by the equation

$$P(\mathbf{z}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \quad (7)$$

where K denotes the total number of mixture components, \mathbf{z} is a feature vector, $\{\pi_k\}_{k=1}^K$ are the mixture weights with $\sum_{k=1}^K \pi_k = 1$, $\{\boldsymbol{\mu}_k\}_{k=1}^K$ are the mean vectors, $\{\boldsymbol{\Sigma}_k\}_{k=1}^K$ are the covariance matrices, and $\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ are the Gaussian component densities.

To obtain the centralized data while simplifying the calculation, for the l -th PG \mathbf{Z}_l , we subtract its mean value from all its patch elements. The mean vector of \mathbf{Z}_l is $\boldsymbol{\mu}_l = \frac{1}{M} \sum_{m=1}^M \mathbf{z}_{l,m}$. Then, the group mean subtracted PG $\tilde{\mathbf{Z}}_l$ is defined as $\tilde{\mathbf{Z}}_l \triangleq \{\tilde{\mathbf{z}}_{l,m} = \mathbf{z}_{l,m} - \boldsymbol{\mu}_l\}_{m=1}^M$.

In particular, by utilizing the GMM learning algorithm, the likelihood of the given PGs $\{\tilde{\mathbf{Z}}_l\}_{l=1}^L$ is

$$\begin{aligned} P(\tilde{\mathbf{Z}}_l|\Theta) &= \prod_{m=1}^M p(\tilde{\mathbf{z}}_{l,m}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\ &= \sum_{k=1}^K \pi_k \prod_{m=1}^M \mathcal{N}(\tilde{\mathbf{z}}_{l,m}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \end{aligned} \quad (8)$$

where $\Theta = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k\}_{k=1}^K$. Assuming that all PGs are independent, the overall objective likelihood function \mathcal{L} can be defined as:

$$\mathcal{L} = \prod_{l=1}^L (P(\tilde{\mathbf{Z}}_l|\Theta)). \quad (9)$$

We maximize \mathcal{L} and solve Θ by taking a log of the objective function \mathcal{L}

$$\ln \mathcal{L} = \sum_{l=1}^L \ln \left(\sum_{k=1}^K \pi_k \prod_{m=1}^M \mathcal{N}(\tilde{\mathbf{z}}_{l,m}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right), \quad (10)$$

where Θ can be solved by the Expectation Maximization (EM) algorithm [43, 47, 55, 56] which consists of the following two steps.

(1) In the E-step, the posterior probability for the k -th Gaussian mixture component is defined as:

$$P(k|\tilde{\mathbf{z}}_{l,m}, \Theta) = \frac{\pi_k \cdot \prod_{m=1}^M \mathcal{N}(\tilde{\mathbf{z}}_{l,m}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{t=1}^K \pi_t \cdot \prod_{m=1}^M \mathcal{N}(\tilde{\mathbf{z}}_{l,m}|\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)}. \quad (11)$$

(2) In the M-step, for each PG $\tilde{\mathbf{Z}}_l$, we have $\sum_{m=1}^M \tilde{\mathbf{z}}_{l,m} = \mathbf{0}$ according to the definition of $\tilde{\mathbf{z}}_{l,m}$. Then, we update the model parameters to maximize the posterior probability as follows:

$$\begin{aligned} \boldsymbol{\mu}_k &= \frac{\sum_{l=1}^L P(k|\tilde{\mathbf{z}}_{l,m}, \Theta) \sum_{m=1}^M \tilde{\mathbf{z}}_{l,m}}{\sum_{l=1}^L P(k|\tilde{\mathbf{z}}_{l,m}, \Theta)} = \mathbf{0}, \\ \boldsymbol{\Sigma}_k &= \frac{\sum_{l=1}^L P(k|\tilde{\mathbf{z}}_{l,m}, \Theta) \sum_{m=1}^M \tilde{\mathbf{z}}_{l,m} \tilde{\mathbf{z}}_{l,m}^\top}{\sum_{l=1}^L P(k|\tilde{\mathbf{z}}_{l,m}, \Theta)}, \\ \pi_k &= \frac{\sum_{l=1}^L P(k|\tilde{\mathbf{z}}_{l,m}, \Theta)}{L}. \end{aligned} \quad (12)$$

By alternating between the E-step and M-step, the model parameters can be updated iteratively until convergence is reached. Finally, we learn a GMM model with K Gaussian components with its mean vectors $\{\boldsymbol{\mu}_k\}_{k=1}^K$, covariance matrices $\{\boldsymbol{\Sigma}_k\}_{k=1}^K$ and mixed weights $\{\pi_k\}_{k=1}^K$. The framework of external prior learning is summarized in Algorithm 1.

4.3. External Dictionary Learning

After obtaining the covariance matrix $\boldsymbol{\Sigma}_k$ of the k -th Gaussian component, we decompose $\boldsymbol{\Sigma}_k$ using Singular Value Decomposition (SVD) to obtain the following:

$$\boldsymbol{\Sigma}_k = \mathbf{U}_k \boldsymbol{\Lambda}_k \mathbf{U}_k^\top, \quad (13)$$

where \mathbf{U}_k is an orthogonal matrix consisting of the eigenvectors of $\boldsymbol{\Sigma}_k$, and $\boldsymbol{\Lambda}_k$ is the diagonal matrix of eigenvalues. We employ the eigenvector matrices $\{\mathbf{U}_k\}_{k=1}^K$ as the external orthogonal dictionary, which is helpful to reconstruct the common latent structures of point clouds. Obviously, it is observed that the eigenvectors in \mathbf{U}_k capture the statistical structures of NSS variations in clean point clouds. Therefore, we can use \mathbf{U}_k to represent the structural variations of the PGs in the k -th Gaussian component.

Algorithm 1: External Prior Learning

Input: External clean point clouds $\{\mathbf{P}_i, i = 1, \dots, S\}$, where S is the number of clean point clouds;

Output: The external GMM model;

- 1 **Initialization:** The initialization mean vectors $\boldsymbol{\mu}_k^{(0)}$, covariance matrices $\boldsymbol{\Sigma}_k^{(0)}$, and the number of Gaussian components K ;
 - 2 **for** $i = 1 : S$ **do**
 - 3 | Extract a local patch N_j for each point \mathbf{p}_j in \mathbf{P}_i ;
 - 4 | Extract NSS-based PG \mathbf{Z}_j for each local patch N_j ;
 - 5 **end**
 - 6 Obtain L PGs $\{\mathbf{Z}_l\}_{l=1}^L$ from all external clean point clouds;
 - 7 **repeat**
 - 8 | **for** $l = 1 : L$ **do**
 - 9 | Compute $P(k|\mathbf{z}_{l,m}, \Theta)$ via Eq. (11);
 - 10 | **end**
 - 11 | **for** $k = 1 : K$ **do**
 - 12 | Calculate the group mean $\boldsymbol{\mu}_k$, covariance matrix $\boldsymbol{\Sigma}_k$, and mixed weight π_k via Eq. (12);
 - 13 | **end**
 - 14 **until** *Maximum iteration number or converge*;
 - 15 **Result:** The final mean vectors $\{\boldsymbol{\mu}_k\}_{k=1}^K$, covariance matrices $\{\boldsymbol{\Sigma}_k\}_{k=1}^K$ and mixed weights $\{\pi_k\}_{k=1}^K$.
-

245 **5. Internal Prior Learning**

Based on the priors learned from external clean point clouds, we utilize it to guide our internal prior learning for the noisy point cloud. We first generate PGs from the input noisy point cloud. Then, we select the best k -th Gaussian component for each noisy PG.

Given an input noisy point cloud, similar to the external prior learning stage, we first extract H local PGs denoted as $\tilde{\mathbf{Z}}_h \triangleq \{\tilde{\mathbf{z}}_{h,m} = \mathbf{z}_{h,m} - \boldsymbol{\mu}_h\}_{m=1}^M, h = 1, 2, \dots, H$, where $\mathbf{z}_{h,m}$ is the patch vector, and $\boldsymbol{\mu}_h$ is the mean vector of the h -th PG. Then, we assign the best k -th Gaussian component obtained from the above group-based GMM learning to each noisy PG $\tilde{\mathbf{Z}}_h$ by the following posterior probability:

$$P(k|\tilde{\mathbf{Z}}_h) = \frac{\prod_{m=1}^M \mathcal{N}(\tilde{\mathbf{z}}_{h,m}|0, \boldsymbol{\Sigma}_k)}{\sum_{t=1}^K \prod_{m=1}^M \mathcal{N}(\tilde{\mathbf{z}}_{h,m}|0, \boldsymbol{\Sigma}_t)}. \quad (14)$$

250 It means that the k -th Gaussian component with the highest probability is selected for each noisy PG $\tilde{\mathbf{Z}}_h$ through maximizing Eq. (14).

We assign the internal noise PGs $\{\tilde{\mathbf{Z}}_h\}_{h=1}^H$ to the components with their corresponding maximum probability. For the k -th component, the noise PGs assigned to it are $\{\tilde{\mathbf{Z}}_{k_h}\}_{h=1}^{H_k}$, where $\tilde{\mathbf{Z}}_{k_h} = [\tilde{\mathbf{z}}_{k_h,1}, \dots, \tilde{\mathbf{z}}_{k_h,M}]$ and $\sum_{k=1}^K H_k = H$.

255 **6. Structure-aware Denoising**

After building the external and internal Gaussian mixture models, we first construct a hybrid dictionary to fuse the external and internal prior information for each subspace. Then, we employ the learned hybrid dictionary to calculate the denoised normals while preserving inherent structural characteristics of the
260 noisy point cloud.

6.1. Hybrid Dictionary Learning

We combine the external prior dictionary and the internal prior dictionary into an orthogonal dictionary to efficiently reconstruct the desired point cloud. The learned external dictionary \mathbf{U}_k just represents the structural variations of

the PGs in the k -th latent subspace. Thus, it can not fully reflect the structural characteristics of the given noisy point cloud. Specifically, \mathbf{U}_k may not well characterize some distinctive structural details specific to the given noisy point cloud. Therefore, the external dictionary and the internal dictionary are complementary. Let the hybrid orthogonal dictionary \mathbf{D}_k for the k -th subspace be

$$\mathbf{D}_k \triangleq [\mathbf{D}_{k,E} \ \mathbf{D}_{k,I}] \in \mathbb{R}^{3d \times 3d}, \quad (15)$$

where $\mathbf{D}_{k,E} \in \mathbb{R}^{3d \times r}$ consists of the first r most significant eigenvectors of \mathbf{U}_k , and $\mathbf{D}_{k,I} \in \mathbb{R}^{3d \times (3d-r)}$ is composed of the internal sub-dictionary eigenvectors learned from noisy PGs $\{\tilde{\mathbf{Z}}_{k_h}\}_{h=1}^{H_k}$.

Without loss of generality, we ignore the subscript for $\tilde{\mathbf{Z}}_{k_h}$ and \mathbf{D}_k , etc., in the following steps for notation simplicity. The hybrid orthogonal dictionary \mathbf{D} can be learned by minimizing the following equation:

$$\min_{\mathbf{D}_I, \{\beta_{h,m}\}} \sum_{h=1}^H \sum_{m=1}^M (\|\tilde{\mathbf{z}}_{h,m} - \mathbf{D}\beta_{h,m}\|_2^2 + \sum_{j=1}^{3d} \lambda_j |\beta_{h,m,j}|), \quad (16)$$

where $\mathbf{D}^\top \mathbf{D} = \mathbf{I}$, $\mathbf{I} \in \mathbb{R}^{3d \times 3d}$ is an identity matrix, $\beta_{h,m} = [\beta_{h,m,1} \ \beta_{h,m,2} \ \cdots \ \beta_{h,m,j} \ \cdots \ \beta_{h,m,3d}]$ is the sparse encoding vector of the patch $\tilde{\mathbf{z}}_{h,m}$ in the PG $\tilde{\mathbf{Z}}_h$, and λ_j is the j -th regularization parameter, which is defined as follows:

$$\lambda_j = \frac{\tau}{\sqrt{\mathbf{\Lambda}_k(j)} + \varepsilon}, \quad (17)$$

265 where τ is the sparse regularization parameter, $\mathbf{\Lambda}_k(j)$ is the j -th diagonal element of the matrix $\mathbf{\Lambda}_k$ in Eq. (13), and ε is a small positive number (e.g., $\varepsilon = 1 \times 10^{-5}$) avoiding a denominator of zero.

The optimization problem in Eq. (16) can be solved by alternatively updating $\beta_{h,m}$ and \mathbf{D}_I for T times. For more information on solving, please refer to
270 [42, 43, 55, 47].

6.2. Dictionary Guided Denoising

We simultaneously perform the denoising and internal sub-dictionary learning process. By solving Eq. (16), we can obtain the sparse coding vector $\{\beta_{h,m}^{(T)}\}$

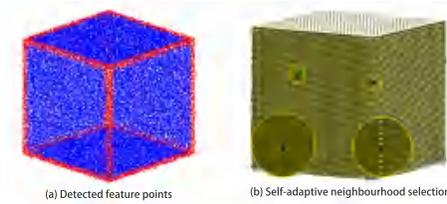


Figure 3: Illustration of feature points selection. (a) Detected feature points (red points) of a Cube model. (b) Self-adaptive neighbourhood selection. Green points are the neighborhoods of a non-feature point. Red points are the neighborhoods of an edge point.

and the orthogonal dictionary $\mathbf{D}^{(T)}$. Then, the noise patch $\mathbf{z}_{h,m}$ in the noise PG $\tilde{\mathbf{Z}}_h$ can be reconstructed by

$$\check{\mathbf{z}}_{h,m} = \mathbf{D}^{(T)}\{\boldsymbol{\beta}_{h,m}^{(T)}\} + \boldsymbol{\mu}_h, \quad (18)$$

where $\check{\mathbf{z}}_{h,m}$ is the latent clean patch, $\boldsymbol{\mu}_h = \frac{1}{M} \sum_{m=1}^M \mathbf{z}_{h,m}$. Then, the latent clean point cloud normal can be recovered by collecting all the reconstructed patches from all PGs. We adopt I_{num} iterations of the above denoising procedures to progressively filter the noisy point cloud for better denoising outputs. Finally, we can obtain the estimated normal of each point in the denoised point cloud. It should be noted that the point included in multiple patches may get more than one estimated normal during the above denoising procedures. We set the average of all its estimated normals as its final normal in this paper.

7. Feature-aware Point Updating

After obtaining estimated normals, we develop a modification of the point updating algorithm in [24] to reposition all points in a feature-aware way.

The original optimization function in [24] is defined as:

$$\sum_i \sum_{j \in N_i} |(\mathbf{p}_i - \mathbf{p}_j) \mathbf{n}_j^\top|^2 + |(\mathbf{p}_i - \mathbf{p}_j) \mathbf{n}_i^\top|^2, \quad (19)$$

where N_i is the neighborhood of point \mathbf{p}_i , \mathbf{p}_i and \mathbf{p}_j are unknown, \mathbf{n}_i and \mathbf{n}_j are normals estimated by our denoising algorithm. Solving the optimization

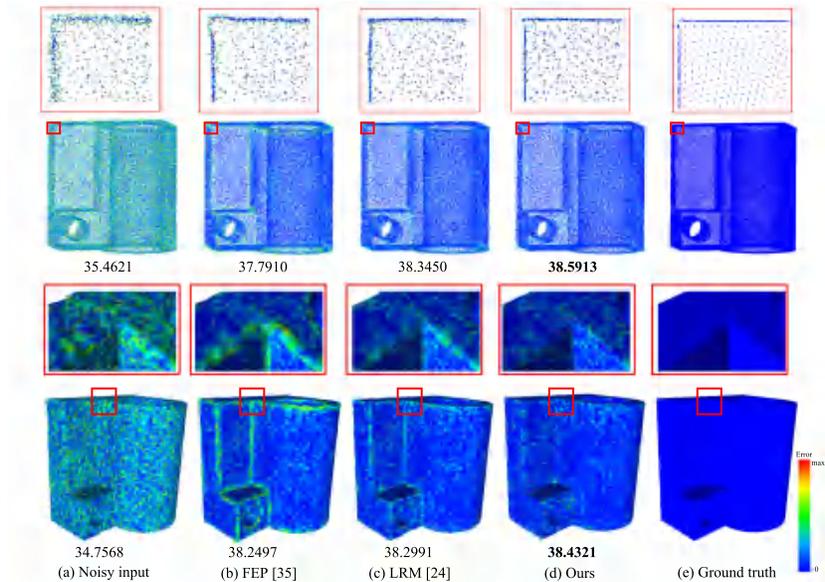


Figure 4: Comparison of position accuracies on the Joint model. The second row: the updated point cloud. The fourth row: the corresponding surface reconstruction results. The figure’s color represents the error between the updated model and the ground truth, with blue indicating a small error and red denoting a large error.

equation by the gradient descent method [24], the new position of \mathbf{p}_i can be computed by

$$\mathbf{p}'_i = \mathbf{p}_i + \eta_i \sum_{j \in N_i} (\mathbf{p}_j - \mathbf{p}_i)(\mathbf{n}_j^\top \mathbf{n}_j + \mathbf{n}_i^\top \mathbf{n}_i), \quad (20)$$

where \mathbf{p}'_i is the new position, and η_i is the step size, which is set to $\frac{1}{3|N_i|}$.

The original method in [24] takes use of all the points of the original neighborhood N_i centering at the point \mathbf{p}_i to update points. However, inaccurate point position estimation will be introduced when \mathbf{p}_i is on corners, edges and other sharp area. Instead of directly using N_i , we employ the neighborhood selection strategies for point normal estimation in [38] to construct an isotropic neighborhood \tilde{N}_i of \mathbf{p}_i to update point position. First, each point \mathbf{p}_i is classified into three categories by analyzing N_i : non-feature points, edge points and corner points. Then, different strategies are designed for each kind of point to adaptively construct qualified neighborhoods \tilde{N}_i [38]. The neighbors selected

according to this strategy are more likely to be isotropic with the candidate points, which can make the point updating more credible. For more details about the neighborhood selection strategies, please refer to [38]. Finally, the constructed isotropic neighborhood \tilde{N}_i is employed by replacing the original neighborhood N_i in Eq. (19) and (20)

$$\sum_i \sum_{j \in \tilde{N}_i} |(\mathbf{p}_i - \mathbf{p}_j) \mathbf{n}_j^\top|^2 + |(\mathbf{p}_i - \mathbf{p}_j) \mathbf{n}_i^\top|^2. \quad (21)$$

Fig. 3(a) shows the detected feature points of our method. And, Fig. 3(b) illustrates the isotropic neighborhoods of an edge point and a non-feature point.

8. Experimental Results and Discussion

8.1. Compared Methods

We compare our structure-aware denoising approach with several popular and state-of-the-art point cloud denoising methods, including the classical MLS [13], WLOP [17], CLOP [18], EAR [45], GPF [46], PointCleanNet (PCN) [11] and Pointfilter (PF) [12].

For the MLS method, we use the relevant function included in the CloudCompare [57]. For WLOP, we implement it based on the CGAL library. The CLOP, EAR and GPF are implemented through source codes released by their corresponding authors. In addition, it is difficult for EAR and GPF to find a suitable radius to balance the removal of noise and gaps near the edges. They all require careful trial-and-error parameter adjustment. For the deep-learning-based methods PCN and PF, we directly utilize the trained models released by the authors. For fair comparison and visualization purposes, we carefully tune the parameters of each method to achieve the best visualization results. We also reconstruct part of the point clouds in order to observe the denoising result, using the method [6] integrated in the MeshLab software [58].

8.2. Parameter Settings

In the external prior learning stage, the main parameters are the patch size d , the number of similar patches M in a PG, and the number of Gaussian

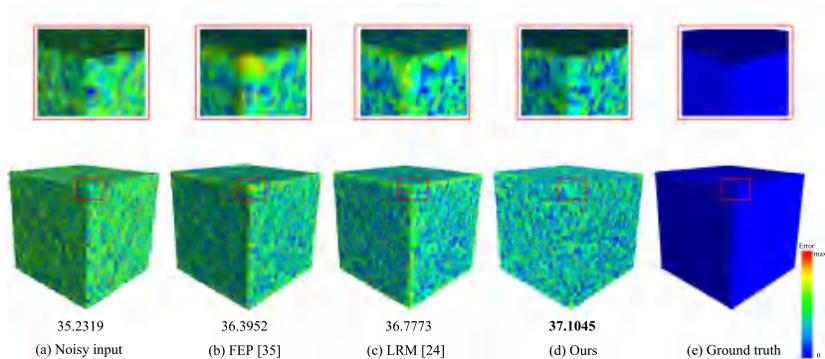


Figure 5: Comparison of position accuracies on the Cube model.

components K in GMM. We also set a non-local searching range W for NSS structures searching, which provides a trade-off between denoising accuracy and speed. In the internal prior learning stage, the main parameters are the number of most important eigenvectors r in the external sub-dictionaries in Eq. (15), the sparse regularization parameter τ in Eq. (17), and the iteration number T for solving Eq. (16). In the denoising and point updating stages, the main parameters are the iteration number I_{num} for the denoising procedures described in Section 6.

Based on our parameter tests and observations, we empirically set: $d = 10$, $M = 10$, $W = 100$, $K = 32$, $r = 15$, $\tau = 0.01$, $T = 3$, and $I_{num} = 5$.

8.3. Dataset

To learn external priors, we prepare a training dataset, covering a wide variety of 3D shapes, from Stanford [59], Georgia and [60], consisting of 10 3D clean models (6 CAD models and 4 non-CAD models). Each point cloud is produced by sampling out 2k-6k points from its mesh. Specifically, we totally generate about half a million patches and about half a million training PGs to learn a finite GMM to describe the external NSS priors.

To demonstrate the performance of our proposed algorithm, our testing dataset includes 20 synthesized noisy point clouds and 8 raw-scan point clouds, which will be introduced in the following experiments.

Table 2: Signal to Noise Ratio (SNR) results of noisy and updated point clouds (dB)

Noise level	Models	Noisy inputs	2007.FEP [35]	2020.LRM [24]	OURS
1.0%	Bunny	34.4858	36.6760	37.5143	37.7012
	Cube	35.2319	36.3951	36.7773	37.1044
	Eight	33.3204	36.7290	38.0518	38.0758
	Joint	35.4621	37.7910	38.3450	38.5913
	Kitten	32.9670	35.3989	36.4117	36.4144
	Plane_sphere	33.4967	36.0665	36.6603	36.8429
2.0%	Bunny	30.8037	34.8236	36.5153	36.5233
	Cube	32.0078	34.6907	35.2720	35.5829
	Eight	29.0676	34.0922	37.5241	37.5023
	Joint	31.8752	36.5959	37.6337	37.7670
	Kitten	29.2216	33.6345	35.8060	35.5842
	Plane_sphere	29.8865	34.8771	35.9046	35.9903
3.0%	Bunny	28.1227	32.6248	35.5319	35.7085
	Cube	29.9663	34.1485	34.8629	34.9918
	Eight	26.1757	30.4682	36.8552	36.8343
	Joint	29.2773	34.8531	37.0556	37.2393
	Kitten	26.5253	31.6380	35.2468	34.8001
	Plane_sphere	27.2009	33.0553	35.2115	35.3058

8.4. Point Updating

To evaluate the performance of our feature-aware point updating method, six clean point clouds (Cube, Eight, Joint, Kitten, Plane_sphere) from Georgia and [60] are selected. And, their corresponding noisy point clouds are synthesized by adding Gaussian noise with standard deviations of 1.0%, 2.0%, and 3.0% of the clean point cloud’s bounding box diagonal length. We compare our feature-aware point updating algorithm with the latest ones, including FEP [35] and LRM [24]. Given the ground truth normals of clean point clouds, which are calculated by the method [61], we employ the three algorithms to update point positions to match ground truth normals. We introduce the Signal to Noise

Ratio (SNR) to evaluate the error between the updated point cloud and its corresponding ground truth (clean point cloud).

Fig. 4 and 5 show the position errors of different techniques on the corrupted Cube and Joint point clouds with the same noise level of 1%, respectively. For visualization purpose, we render the colors of position errors on the updating results. As shown in Fig. 4 and 5, we observe that the results of our feature-aware point updating method are substantially better than those of the state-of-the-art methods, especially in the edge and corner regions. The FEP [35] approach may lead to deformation and over-smoothing in the feature regions. LRM [24] performs better to some extent. However, many noisy points and over-smoothing still exist in the feature regions.

Table 2 lists the SNR results of noisy and updated point clouds, and demonstrates that our approach outperforms the state-of-the-art methods. When the noise level varies, our method is able to achieve better results on most of the point clouds. However, when the noise level gets larger, some of the noise may be treated as feature points, resulting in a degradation in the performance of our algorithm.

8.5. Synthetic Noisy Point Clouds

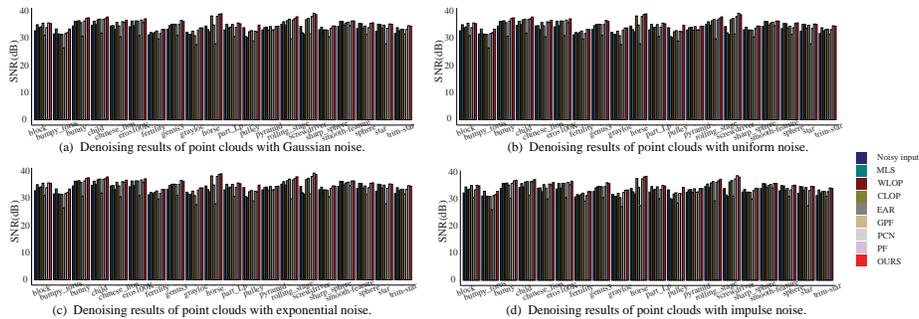


Figure 6: Comparison of SNR results on 20 synthetic noisy point clouds under four different noise types (1% noise).

We also ran experiments on 20 synthetic noisy point clouds, which are specified in Fig. 6. And, the specific values of the first eight point clouds are shown

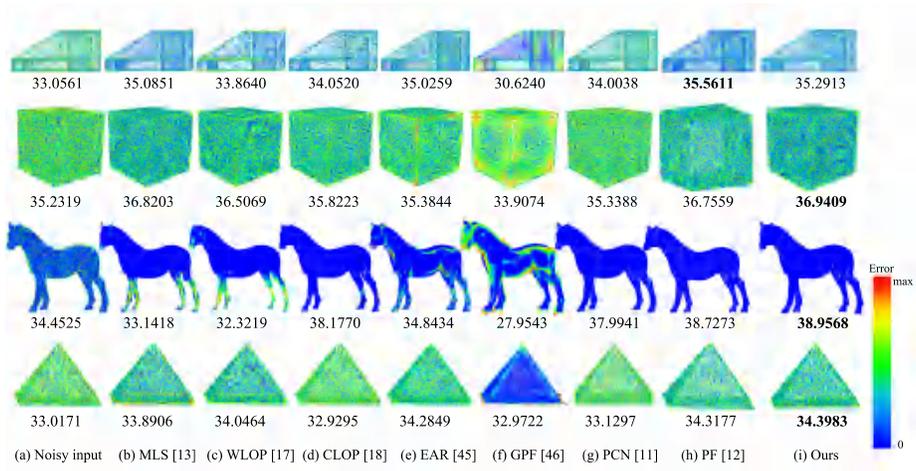


Figure 7: Visual comparison of denoising results on Part_Lp, Cube, Horse and Pyramid with 1% Gaussian noise, respectively.

in Table 3. Four different noise types, i.e., Gaussian noise, uniform noise, exponential noise, and impulse noise, are employed to corrupt the noise-free point clouds.

The denoising results of point clouds with 1.0% of Gaussian noise are shown in Fig. 7. For the point clouds Part_Lp, Cube and Pyramid, which are all geometrically well-structured, our method outperforms most methods in terms of maintaining edges and corners. For the Horse point cloud, our method achieves better results than the comparative methods for the legs and ears, which are more prone to residual noise.

In Fig. 8, we show the denoising results of point clouds with uniform noise and exponential noise. The Chinese-lion and Rolling-stage point clouds have relatively complex structures. First, our method can achieve better results for different noise types. Second, for the fine and weak features in the model, such as the back of Chinese-lion and the slide groove of Rolling-stage, our method gets relatively good retention.

In Fig. 1 and 9, these point clouds are added with impulse noise. Point clouds in Fig. 1 and 9 have sharp and rounded features, and our method can

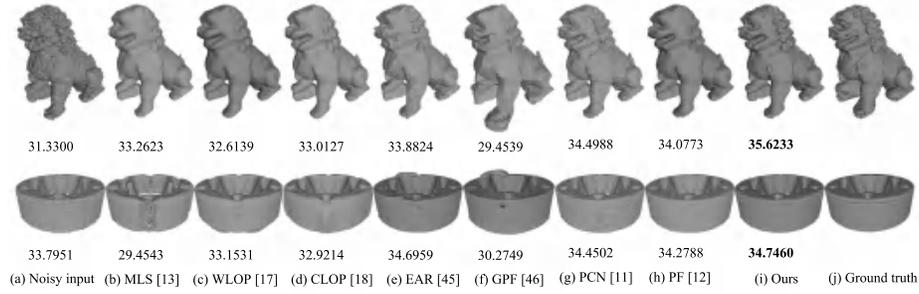


Figure 8: Comparison of denoising results on the Chinese-lion model (1% uniform noise) and the rolling stage model (1% exponential noise).

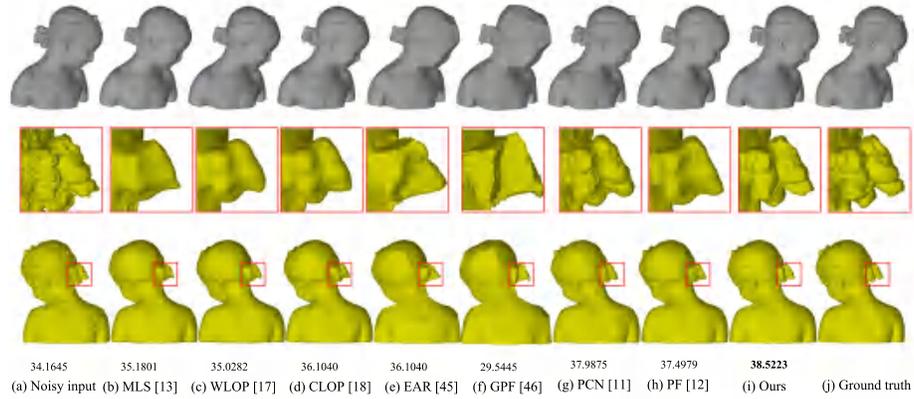


Figure 9: Comparison of denoising results on the Child model. Our approach can better preserve the structural characteristics of the point cloud without over-smoothness and over-sharpening. The first and third rows are the front and back of the model, respectively. The second row is an enlarged view of the red box.

Table 3: SNR results of different denoising methods on the synthetic testing dataset under different noise types (dB)

	Models	Noisy inputs	MLS [13]	WLOP [17]	CLOP [18]	EAR [45]	GPF [46]	PCN [11]	PF [12]	Ours
Gaussian noise	Block(32370)	32.7114	34.9623	33.8586	34.1632	35.4858	30.9949	33.8669	35.5949	35.3690
	Bumpy_torus(16815)	31.5592	33.3654	31.5517	31.4631	31.3829	26.4354	31.7168	32.0648	33.3267
	Bunny(35947)	34.4733	36.2228	36.1757	36.4304	35.7402	30.7267	36.3224	37.2506	37.4883
	Child(50002)	34.7096	36.2311	35.0084	36.7656	37.0004	31.8791	36.9163	37.1544	37.7924
	Chinese_lion(50003)	34.5006	34.6038	33.1656	35.8275	34.5400	30.5265	36.0557	35.8121	36.6001
	Eros100K(50002)	34.1994	36.2750	34.5893	36.3249	36.2848	30.9868	36.6745	35.7685	37.1272
	Fertility(13971)	31.2241	32.0995	31.6577	32.1030	32.6372	29.6767	31.7170	33.2119	33.1843
	Genus3(29663)	33.3042	34.6189	35.0808	35.1433	35.0457	31.0256	35.0977	36.5282	36.1678
Uniform noise	Block(32370)	29.7100	33.5995	33.2701	32.7153	34.5622	30.4165	31.4681	34.5535	34.5394
	Bumpy_torus(16815)	28.6315	31.2963	30.7389	29.8542	29.9909	26.3209	28.9925	30.3107	31.6084
	Bunny(35947)	31.4603	35.1245	35.3983	35.0259	34.6499	29.2762	35.1348	36.4272	36.5218
	Child(50002)	31.4316	35.1697	34.6544	34.8758	35.2553	29.5286	35.6713	36.1458	36.6423
	Chinese_lion(50003)	31.3300	33.2623	32.6139	33.0127	33.8824	29.4539	34.4988	34.0773	35.6233
	Eros100K(50002)	30.8618	34.9164	34.0897	33.9988	34.9504	29.4317	35.2778	34.4956	36.0204
	Fertility(13971)	28.5180	30.7838	31.1286	30.4700	31.8082	26.1932	29.2887	31.8483	32.6583
	Genus3(29663)	30.3328	33.8038	34.5008	33.8901	35.0106	27.3852	33.6784	35.7200	35.7785
Exponential noise	Block(32370)	31.3859	31.4066	31.5925	31.6969	31.3745	30.4165	31.8081	32.4260	32.1109
	Bumpy_torus(16815)	30.1528	30.5191	29.9169	29.9210	30.1816	26.3209	30.2476	30.5246	30.7961
	Bunny(35947)	32.9014	31.8835	33.1195	33.2080	32.4911	29.2762	33.4411	33.4277	33.5672
	Child(50002)	33.1686	31.6938	32.5029	33.0040	32.9181	29.5286	33.8946	33.4676	34.0281
	Chinese_lion(50003)	32.9615	30.3184	31.7411	32.2168	31.2008	29.4539	33.5059	32.9129	33.6877
	Eros100K(50002)	32.3714	31.3972	31.8033	31.9650	33.0237	29.4317	33.0827	32.2126	33.0725
	Fertility(13971)	30.2075	28.0963	29.9985	30.1474	31.0868	26.1932	33.0827	31.3723	31.0269
	Genus3(29663)	31.8061	28.8259	32.1454	32.1928	32.3096	27.3852	32.4359	32.8518	32.6107
Impulse noise	Block(32370)	32.4301	34.8471	33.7464	34.3842	35.7253	30.7493	34.7831	36.6850	36.0761
	Bumpy_torus(16815)	30.9508	33.3195	31.2179	31.5767	31.0776	26.4760	31.3367	32.0269	32.6121
	Bunny(35947)	34.1904	35.3164	36.0098	36.1420	34.1021	30.7202	37.5907	38.2711	37.5518
	Child(50002)	34.1645	35.1801	35.0282	36.1040	34.1190	29.5445	37.9875	37.4979	38.5223
	Chinese_lion(50003)	34.1217	32.4850	33.0609	33.9854	34.5265	30.5276	36.6474	35.4468	37.1732
	Eros100K(50002)	33.6502	35.0984	34.5261	34.7914	34.8136	31.4704	37.5722	35.9032	37.8052
	Fertility(13971)	30.9430	30.5337	31.6330	31.4144	32.2833	26.7520	31.8502	33.9514	33.7002
	Genus3(29663)	32.8802	31.7545	34.9259	35.1025	34.9684	29.0893	36.2427	37.7510	36.6884

achieve the best denoising results.

Table 3 lists the SNR results of different methods on 8 testing point clouds with different noise types. Fig. 6 visually shows the denoising results of each test point cloud in a bar chart. As can be seen from Table 3 and Fig. 6, our method can achieve the best result on most of synthetic noisy point clouds.

8.6. Real-World Noisy Point Clouds

To verify the performance of our method on denoising real-world noisy point clouds, we select two Kinect v1 models, two Kinect v2 models and four Kinect Fusion models from [60], as listed in Table 4. We use the data collected by the high-precision scanner as ground truth.

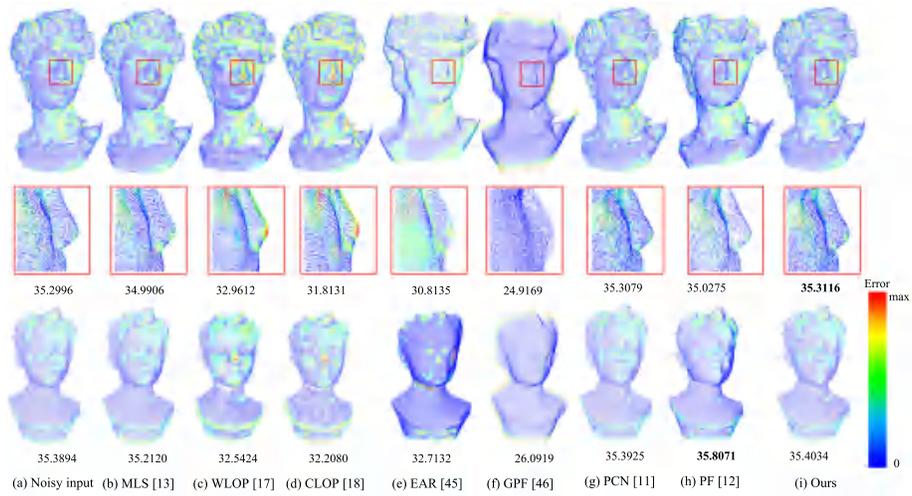


Figure 10: Comparison of denoising results on real-world noisy point clouds David and Boy, respectively.

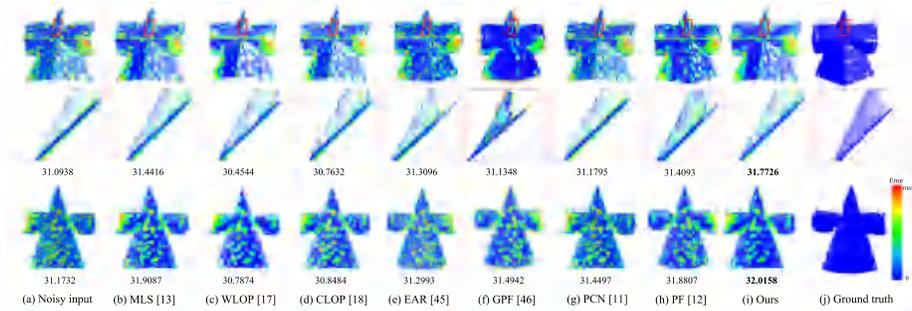


Figure 11: Comparison of denoising results on real-world noisy point clouds Pyramid_02 and Cone_24, respectively.

Table 4: SNR results of different denoising methods on real-world noisy point clouds (dB)

	Models	Noisy inputs	MLS [13]	WLOP [17]	CLOP [18]	EAR [45]	GPF [46]	PCN [11]	PF [12]	Ours
	Big_girl	34.6342	34.7712	32.935	32.6422	31.7982	23.6917	34.6404	34.7903	34.6633
	Boy	35.3894	35.2121	32.5424	32.2080	32.7132	26.0919	35.3925	35.8071	35.4034
	Cone	31.6284	31.6723	31.5938	31.3929	30.2082	20.6547	31.6341	31.5068	31.6560
Raw scan data from Kinect	David	35.2996	34.9906	32.9612	31.8131	30.8135	24.9169	35.3078	35.0276	35.3116
	Cone_24	31.1732	31.9087	30.7874	30.8485	31.2993	31.4942	31.4497	31.8807	32.0158
	Boy_01	33.1496	33.1896	30.1985	32.0080	33.4957	33.2014	33.1955	33.4902	33.1369
	Pyramid_02	31.0938	31.4416	30.4544	30.7632	31.3096	31.1348	31.1795	31.4093	31.7726
	Pyramid_24	31.1967	31.4230	30.6410	30.9788	31.1981	31.3974	31.2450	31.4429	31.7852

Fig. 10 presents the denoising results on two Kinect Fusion models, namely Boy and David, where there are only some fine noises in the Kinect Fusion point clouds. From Fig. 10, we can see that our method can remove these tiny noises while keeping the structural features unaltered. In contrast, WLOP and CLOP significantly deform the origin point clouds. The denoising results of the EAR and GPF methods are not well maintained for the detail part. The results of PCN and PF are closer to ours when the noise level is low.

Fig. 11 provides the denoising results on two point clouds, namely Pyramid_02 and Cone_24. From Fig. 11, we can see that our method is capable of retaining both sharp and structural features. MLS, WLOP, and CLOP perform well in the smooth region, but there is a gap in the feature region. PCN performs poorly in both sharp feature areas and smooth areas. PF does not hold well in areas with edges. Compared to state-of-the-art point cloud denoising techniques, we observe that our algorithm produces visually better results in terms of noise removal and structural feature retention.

Table 4 shows SNR results of all testing real-world noisy point clouds. As can be seen from Table 4, our method is comparable to the state-of-the-art methods on most of the point clouds.

According to the above figures and tables, it can be seen that MLS, WLOP, and CLOP can get good results for models with obscure features, but it does not take into account sharp feature preservation in the denoising process. EAR and GPF are mainly designed to retain sharp features, so they have some advantages in sharp feature retention. However, both methods have problems such as over-sharpening the point cloud, creating holes, and failing to retain tiny details. The robustness of these methods becomes poor when encountering large noise. PCN is a depth frame method, and we only use the denoising module in it. It has a limited ability to maintain complex structures. In addition to that, it is sensitive to the noise type. PF’s denoising result performs well, but it still has shortcomings, such as failure to retain fine features, over-smoothing of certain point clouds, and slight sensitivity to noise. Although our method may not be the best in every aspect, we can still achieve better comprehensive performance

on point clouds with complex structures.

415 8.7. *Complex Structure Preserving*

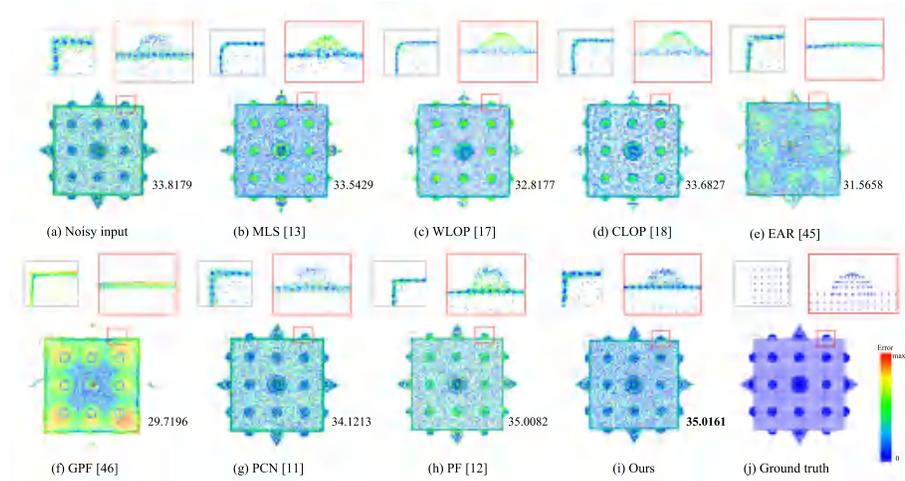


Figure 12: Comparison of denoising results on a synthetic noisy point cloud with complex structure. Our approach can maintain the point cloud’s original characteristics.

In order to demonstrate the superiority of our structure-aware denoising method, we design several point clouds with complex structures. The Cube model with 1% impulse noise in Fig. 1 contains both sharp and rounded corners. The designed point cloud in Fig. 12 characterizes some sharp and small round protrusions on the square with 1% Gaussian noise. Besides, the Child point cloud from [60] with complex structure is also employed, which contains 1% impulse noise.

It is observed from Fig. 1, 9 and 12 that our approach produces generally more desirable results, in terms of simultaneously preserving sharp and round features as well as fine details. In Fig. 1, MLS, WLOP, CLOP and PCN tend to smooth sharp corners evidently for removing noise. PF performs well in both sharp and round corners retention, but there are some noisy points that are not handled in the impulse noisy point cloud. As shown in Fig. 9, the MLS, WLOP, and PF methods produce greater smoothing in the hair part of

430 the Child, while the EAR and GPF methods have varying degrees of outward diffusion. The PCN method is still rough on the Child’s body. Our method not only keeps the smooth body part and the feature-rich bun part, but also leaves a certain degree of fine hair. In Fig. 12, our method is able to preserve the original small round protrusions and without distortion. Despite that MLS, 435 WLOP and CLOP are good at generating smooth results, they still fail to retain sharp features. EAR and GPF preserve sharp features considerably, but lose the small round protrusions. Although the effects of PCN and PF perform well to some extent, the recovery result on small protuberances is slightly less than ours.

440 9. Conclusion

In this paper, we proposed a structure-aware denoising method for real-world noisy point clouds with complex structures by utilizing useful information from external and internal point clouds. We first perform external prior learning from clean point clouds to generate external dictionary which preserves fine-scale 445 structural information. Then, we obtain the internal prior knowledge from the given noisy point cloud, which may not be accurate due to the interference of corrupted noise. Finally, a set of hybrid orthogonal dictionaries are constructed by integrating both external and internal priors for the normal estimation. We also propose a feature-aware point updating method which can adaptively adjust 450 the neighborhood for point updating. Extensive experiments and comparisons on real-world noisy point clouds with complex structures demonstrated that our method achieves better overall performance than state-of-the-art point cloud denoising methods, in terms of both visual quality and evaluation errors.

Acknowledgment

455 The authors sincerely acknowledge the anonymous reviewers for their insights and comments to further improve the quality of the manuscript. This

work was supported by National Key R&D Program of China (2018YFB1004904) and Fundamental Research Funds for the Central Universities (30918012203).

References

- 460 [1] H. Chen, J. Shen, Denoising of point cloud data for computer-aided design, engineering, and manufacturing, *Engineering with Computers* 34 (3) (2018) 523–541. doi:10.1007/s00366-017-0556-4.
- [2] H. Woo, E. Kang, S. Wang, K. H. Lee, A new segmentation method for point cloud data, *International Journal of Machine Tools and Manufacture* 42 (2) (2002) 167–178. doi:10.1016/S0890-6955(01)00120-1.
- 465 [3] L. Landrieu, M. Simonovsky, Large-scale point cloud semantic segmentation with superpoint graphs, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4558–4567. doi:10.1109/CVPR.2018.00479.
- 470 [4] R. Klokov, V. Lempitsky, Escape from cells: Deep kd-networks for the recognition of 3d point cloud models, in: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 863–872. doi:10.1109/ICCV.2017.99.
- [5] S. Xie, S. Liu, Z. Chen, Z. Tu, Attentional shapecontextnet for point cloud recognition, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4606–4615. doi:10.1109/CVPR.2018.00484.
- 475 [6] M. Kazhdan, H. Hoppe, Screened poisson surface reconstruction, *ACM Transactions on Graphics* 32 (3) (2013) 1–13. doi:10.1145/2487228.2487237.
- 480 [7] B. Guo, Q. Li, X. Huang, C. Wang, An improved method for power-line reconstruction from point cloud data, *Remote Sensing* 8 (1) (2016) 36–36. doi:10.3390/rs8010036.

- [8] C.-H. Lin, C. Kong, S. Lucey, Learning efficient point cloud generation for
485 dense 3d object reconstruction, in: Proceedings of the AAAI Conference
on Artificial Intelligence, 2018, pp. 7114–7121.
- [9] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud,
J. A. Levine, A. Sharf, C. T. Silva, A survey of surface reconstruction
from point clouds, *Computer Graphics Forum* 36 (1) (2017) 301–329. doi:
490 10.1111/cgf.12802.
- [10] X.-F. Han, J. S. Jin, M.-J. Wang, W. Jiang, L. Gao, L. Xiao, A review
of algorithms for filtering the 3d point cloud, *Signal Processing: Image
Communication* 57 (2017) 103–112. doi:10.1016/j.image.2017.05.009.
- [11] M. Rakotosaona, V. La Barbera, P. Guerrero, N. J. Mitra, M. Ovsjanikov,
495 Pointcleannet: Learning to denoise and remove outliers from dense point
clouds, *Computer Graphics Forum* 39 (1) (2020) 185–203. doi:10.1111/
cgf.13753.
- [12] D. Zhang, X. Lu, H. Qin, Y. He, Pointfilter: Point cloud filtering
via encoder-decoder modeling, *IEEE Transactions on Visualization and
500 Computer Graphics* 27 (3) (2021) 2015–2027. doi:10.1109/TVCG.2020.
3027069.
- [13] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C. Silva, Com-
puting and rendering point set surfaces, *IEEE Transactions on Visualiza-
tion and Computer Graphics* 9 (1) (2003) 3–15. doi:10.1109/TVCG.2003.
505 1175093.
- [14] S. Fleishman, D. Cohen-Or, C. T. Silva, Robust moving least-squares fitting
with sharp features, *ACM Transactions on Graphics* 24 (3) (2005) 544–552.
doi:10.1145/1073204.1073227.
- [15] A. C. Öztireli, G. Guennebaud, M. Gross, Feature preserving point set
510 surfaces based on non-linear kernel regression, *Computer Graphics Forum*
28 (2) (2009) 493–501. doi:10.1111/j.1467-8659.2009.01388.x.

- [16] Y. Lipman, D. Cohen-Or, D. Levin, H. Tal-Ezer, Parameterization-free projection for geometry reconstruction, *ACM Transactions on Graphics* 26 (3) (2007) 22–es. doi:10.1145/1275808.1276405.
- 515 [17] H. Huang, D. Li, U. Ascher, H. Zhang, D. Cohen-Or, Consolidation of unorganized point clouds for surface reconstruction, *ACM Transactions on Graphics* 28 (5) (2009) 1–7. doi:10.1145/1618452.1618522.
- [18] R. Preiner, O. Mattausch, M. Arikani, R. Pajarola, M. Wimmer, Continuous projection for fast l1 reconstruction, *ACM Transactions on Graphics* 33 (4) 520 (2014) 1–13. doi:10.1145/2601097.2601172.
- [19] Shin Yoshizawa, A. Belyaev, H.-P. Seidel, Smoothing by example: Mesh denoising by averaging with similarity-based weights, in: *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, 2006, pp. 9–9. doi:10.1109/SMI.2006.38.
- 525 [20] G. Rosman, A. Dubrovina, R. Kimmel, Patch-collaborative spectral point-cloud denoising, *Computer Graphics Forum* 32 (8) (2013) 1–12. doi:10.1111/cgf.12139.
- [21] Y. Sun, S. Schaefer, W. Wang, Denoising point sets via l_0 minimization, *Computer Aided Geometric Design* 35-36 (2015) 2–15. doi:10.1016/j.cagd.2015.03.011.
- 530 [22] K. Sarkar, F. Bernard, K. Varanasi, C. Theobalt, D. Stricker, Structured low-rank matrix factorization for point-cloud denoising, in: *2018 International Conference on 3D Vision (3DV)*, 2018, pp. 444–453. doi:10.1109/3DV.2018.00058.
- 535 [23] H. Chen, M. Wei, Y. Sun, X. Xie, J. Wang, Multi-patch collaborative point cloud denoising via low-rank recovery with graph constraint, *IEEE Transactions on Visualization and Computer Graphics* 26 (11) (2020) 3255–3270. doi:10.1109/TVCG.2019.2920817.

- [24] X. Lu, S. Schaefer, J. Luo, L. Ma, Y. He, Low rank matrix approximation
540 for 3d geometry filtering, *IEEE Transactions on Visualization and Computer Graphics* (2020) 1–1. doi:10.1109/TVCG.2020.3026785.
- [25] R. Roveri, A. C. Öztireli, I. Pandele, M. Gross, PointProNets: Consolidation of point clouds with convolutional neural networks, *Computer Graphics Forum* 37 (2) (2018) 87–99. doi:10.1111/cgf.13344.
- [26] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, P.-A. Heng, Ec-net: an edge-aware
545 point set consolidation network, in: *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 386–402. doi:10.1007/978-3-030-01234-2_24.
- [27] A. Adams, N. Gelfand, J. Dolson, M. Levoy, Gaussian KD-trees for fast
550 high-dimensional filtering, *ACM Transactions on Graphics* 28 (3) (2009) 1–12. doi:10.1145/1531326.1531327.
- [28] S. Fleishman, I. Drori, D. Cohen-Or, Bilateral mesh denoising, *ACM Transactions on Graphics* 22 (3) (2003) 950–953. doi:10.1145/882262.882368.
- [29] A. Miropolsky, A. Fischer, Reconstruction with 3d geometric bilateral filter,
555 in: *Proceedings of the Ninth ACM Symposium on Solid Modeling and Applications*, 2004, pp. 225–229. doi:10.5555/1217875.1217910.
- [30] B.-Q. Shi, J. Liang, Q. Liu, Adaptive simplification of point cloud using
k-means clustering, *Computer-Aided Design* 43 (8) (2011) 910–922. doi:10.1016/j.cad.2011.04.001.
- [31] Y. Zheng, G. Li, S. Wu, Y. Liu, Y. Gao, Guided point cloud denoising
560 via sharp feature skeletons, *The Visual Computer* 33 (6) (2017) 857–867. doi:10.1007/s00371-017-1391-8.
- [32] P.-S. Wang, X.-M. Fu, Y. Liu, X. Tong, S.-L. Liu, B. Guo, Rolling guidance normal filter for geometric processing, *ACM Transactions on Graphics*
565 34 (6) (2015) 1–9. doi:10.1145/2816795.2818068.

- [33] C. Chen, Z. Xiong, X. Tian, Z.-J. Zha, F. Wu, Real-world image denoising with deep boosting, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42 (12) (2020) 3071–3087. doi:10.1109/TPAMI.2019.2921548.
- 570 [34] G. Taubin, Linear anisotropic mesh filtering, Res. Rep. RC2213 IBM 1 (4).
- [35] X. Sun, P. L. Rosin, R. Martin, F. Langbein, Fast and effective feature-preserving mesh denoising, *IEEE Transactions on Visualization and Computer Graphics* 13 (5) (2007) 925–938. doi:10.1109/TVCG.2007.1065.
- [36] Y. Zheng, G. Li, X. Xu, S. Wu, Y. Nie, Rolling normal filtering for point
575 clouds, *Computer Aided Geometric Design* 62 (2018) 16–28. doi:10.1016/j.cagd.2018.03.004.
- [37] S. K. Yadav, U. Reitebuch, M. Skrodzki, E. Zimmermann, K. Polthier, Constraint-based point set denoising using normal voting tensor and restricted quadratic error metrics, *Computers and Graphics* 74 (2018) 234–
580 243. doi:10.1016/j.cag.2018.05.014.
- [38] J. Cao, H. Chen, J. Zhang, Y. Li, X. Liu, C. Zou, Normal estimation via shifted neighborhood for point cloud, *Journal of Computational and Applied Mathematics* 329 (2018) 57–67. doi:10.1016/j.cam.2017.04.027.
- 585 [39] K. Dabov, A. Foi, V. Katkovnik, K. Egiazarian, Image denoising by sparse 3-D transform-domain collaborative filtering, *IEEE Transactions on Image Processing* 16 (8) (2007) 2080–2095. doi:10.1109/TIP.2007.901238.
- [40] J. Mairal, F. Bach, J. Ponce, G. Sapiro, A. Zisserman, Non-local sparse models for image restoration, in: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2009, pp. 2272–2279. doi:10.1109/ICCV.2009.5459452.
- 590

- [41] I. Mosseri, M. Zontak, M. Irani, Combining the power of Internal and External denoising, in: IEEE International Conference on Computational Photography (ICCP), 2013, pp. 1–9. doi:10.1109/ICCPHOT.2013.6528298.
- 595 [42] F. Chen, L. Zhang, H. Yu, External patch prior guided internal clustering for image denoising, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 603–611. doi:10.1109/ICCV.2015.76.
- [43] J. Xu, L. Zhang, D. Zhang, External prior guided internal prior learning for
600 real-world noisy image denoising 27 (6) (2018) 2996–3010. doi:10.1109/TIP.2018.2811546.
- [44] Z. Zha, X. Yuan, B. Wen, J. Zhang, J. Zhou, C. Zhu, Simultaneous nonlocal self-similarity prior for image denoising, in: 2019 IEEE International Conference on Image Processing (ICIP), 2019, pp. 1119–1123.
605 doi:10.1109/ICIP.2019.8804272.
- [45] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, H. R. Zhang, Edge-aware point set resampling, ACM Transactions on Graphics 32 (1) (2013) 1–12. doi:10.1145/2421636.2421645.
- [46] X. Lu, S. Wu, H. Chen, S. K. Yeung, W. Chen, M. Zwicker, GPF:
610 GMM-inspired feature-preserving point set filtering, IEEE Transactions on Visualization and Computer Graphics 24 (8) (2018) 2315–2326. doi:10.1109/TVCG.2017.2725948.
- [47] Z. Zha, X. Yuan, J. Zhou, C. Zhu, B. Wen, Image restoration via simultaneous nonlocal self-similarity priors, IEEE Transactions on Image Processing
615 29 (2020) 8561–8576. doi:10.1109/TIP.2020.3015545.
- [48] P. Hermosilla, T. Ritschel, 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. doi:10.1109/ICCV.2019.00014.

- 620 [49] D. Lu, X. Lu, Y. Sun, J. Wang, Deep feature-preserving normal estimation for point cloud filtering, *Computer-Aided Design* 125 (2020) 102860. doi: <https://doi.org/10.1016/j.cad.2020.102860>.
- [50] P. Guerrero, Y. Kleiman, M. Ovsjanikov, N. J. Mitra, Pcpnet: Learning local shape properties from raw point clouds, *Computer Graphics Forum* 37 (2) (2018) 75–85. doi:10.1111/cgf.13343.
- 625 [51] S. Luo, W. Hu, Differentiable manifold reconstruction for point cloud denoising, in: *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1330–1338. doi:10.1145/3394171.3413727.
- [52] Z. Liu, X. Xiao, S. Zhong, W. Wang, Y. Li, L. Zhang, Z. Xie, A feature-preserving framework for point cloud denoising, *Computer-Aided Design* 630 127 (2020) 102857. doi:10.1016/j.cad.2020.102857.
- [53] L. Karacan, E. Erdem, A. Erdem, Structure-preserving image smoothing via region covariances, *ACM Transactions on Graphics* 32 (6) (2013) 1–11. doi:10.1145/2508363.2508403.
- 635 [54] D. A. Reynolds, Gaussian mixture models, *Encyclopedia of biometrics* 741 (2009) 659–663.
- [55] J. Xu, L. Zhang, W. Zuo, D. Zhang, X. Feng, Patch group based nonlocal self-similarity prior learning for image denoising, in: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 244–252. doi:10.1109/ICCV.2015.36.
- 640 [56] D. Zoran, Y. Weiss, From learning models of natural image patches to whole image restoration, in: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 479–486. doi:10.1109/ICCV.2011.6126278.
- 645 [57] D. Girardeau-Montaut, Cloudcompare-open source project, *OpenSource Project* 588.

- [58] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, et al., Meshlab: an open-source mesh processing tool., in: Eurographics Italian chapter conference, 2008, pp. 129–136.
- 650 [59] A. Gardner, C. Tchou, T. Hawkins, P. Debevec, Linear light source reflectometry, *ACM Transactions on Graphics* 22 (3) (2003) 749–758. doi:10.1145/1201775.882342.
- [60] P.-S. Wang, Y. Liu, X. Tong, Mesh denoising via cascaded normal regression, *ACM Transactions on Graphics* 35 (6) (2016) 1–12. doi:10.1145/2980179.2980232.
- 655 [61] J. Zhang, J. Cao, X. Liu, J. Wang, J. Liu, X. Shi, Point cloud normal estimation via low-rank subspace clustering, *Computers and Graphics* 37 (6) (2013) 697–706. doi:10.1016/j.cag.2013.05.008.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: