Computer-Aided Design 140 (2021) 103088

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Shape-aware Mesh Normal Filtering

Saishang Zhong [a], Zhenzhen Song [b], Zheng Liu [c,*], Zhong Xie [c], Jianguo Chen [a], Lu Liu [b], Renjie Chen [d]

[a] School of Earth Resources, State Key Laboratory of Geological Processes and Mineral Resources, China University of Geosciences, Wuhan 430074, China
[b] School of Geography and Information Engineering, China University of Geosciences, Wuhan 430074, China
[c] School of Geography and Information Engineering, National Engineering Research Center of Geographic Information System, China University of Geosciences, Wuhan 430074, China
[d] School of Mathematical Sciences, University of Science and Technology of China, Hefei 230026, China

ARTICLE INFO

Article history:
Received 21 May 2021
Accepted 22 June 2021

Keywords:
Mesh denoising
ℓ_0 minimization
Spectral analysis
Normal filtering

ABSTRACT

Mesh denoising is a fundamental yet open problem in geometry processing. The main challenge is to remove noise while recovering the shape of the underlying surface as accurately as possible. In this paper, we propose a novel joint bilateral filter on the face normal field. The key is to estimate a reliable guidance normal field by constructing a shape-aware consistent patch for accurately describing the local shape of each face. To this end, we first select a candidate patch for each face by using a newly defined consistent metric considering both patch flatness and face-to-patch orientation similarity. Then, spectral analysis is used in combination with ℓ_0 minimization to refine the candidate patches in a shape-aware manner. The refined patches do not contain any features, and therefore they can accurately describe the local shape of the underlying surface. After smoothing the face normal field, vertex positions are reconstructed to match the filtered face normals. Our mesh denoising method is theoretically rooted and practical for dealing with the meshes containing corners with low sampling rates, multi-scale features, or narrow structure regions. Extensive experimental results demonstrate that our method can significantly improve the feature preserving capability of joint normal filter and outperforms state-of-the-art methods visually and quantitatively.

1. Introduction

Triangular meshes are widely used in various fields, e.g., computer graphics, 3D computer vision, urban modeling, etc. Recently, because of the rapid development of scanning devices (e.g., Microsoft Kinect, Xtion pro), the geometric modeling process has been dramatically simplified, resulting in triangular meshes acquired in quantity from the real world. However, even with high-fidelity devices, the acquired meshes are inevitably contaminated by noise introduced in capturing and reconstruction processes. The noise not only degrades the visual quality of meshes, but also causes troubles in downstream applications [1–3]. Thus, mesh denoising has become an essential and increasingly important task of geometry processing. The main challenge is to remove noise while maximally preserving geometric features. This problem gets more difficult for meshes containing corners with low sampling rates, multi-scale features, or narrow structure regions.

Mesh denoising has been studied in the last decade extensively. Researchers have proposed many remarkable methods [1,4–8] to tackle this problem. Although these state-of-the-art methods have gained success in some respects, they still have limitations when handling meshes with some special characteristics including corners with low sampling rates, multi-scale features, or narrow structure regions. For example, bilateral normal filtering [4] tends to blur sharp features, since it cannot distinguish sharp features from noise clearly; guided normal filtering [6] can preserve sharp features, but it is hard to produce desired results when handling meshes containing narrow structure regions; The sparse optimization methods [1,5] can effectively remove noise while preserving sharp features. However, these methods suffer from staircase artifacts in smoothly curved regions for their sparsity requirements. The performance of the data-driven method [7] is highly dependent on the completeness of training data. The low-rank method [8] can recover pattern similarity patches of the surface, but cannot preserve sharp features, especially in the case of high noise. The limitations of these denoising methods degrade the quality of results produced by them.

* Corresponding author.
E-mail address: liu.zheng.jojo@gmail.com (Z. Liu).

https://doi.org/10.1016/j.cad.2021.103088
0010-4485/

We propose a joint bilateral normal filter because of the issues mentioned above for the existing method denoising methods. The critical part of our joint bilateral filter is to estimate a reliable guidance field from the noisy mesh, which can accurately describe the shape of the underlying surface. To this end, we propose a consistent patch construction scheme including two steps, i.e., candidate patch selection followed by final patch refinement. Specifically, for each face, we select the relatively smooth patch with most faces sharing the orientations similar to the current face, via a new consistency measurement. Then, by using spectral partitioning, we refine the candidate patch (of the current face) to match the local shape of the underlying surface. As a result, the normal guidance field, estimated from the refinement patches, can accurately describe the shape of the underlying surface in the presence of noise. We validate our mesh denoising method on a variety of meshes with either synthetic or raw noise to show its effectiveness and versatility. The main contributions of this paper are summarized as follows:

- We design a simple yet effective consistency measurement, considering both patch flatness and face-to-patch orientation similarity, to select a candidate patch for each face. Compared to the consistent patch created by existing methods, our candidate patch contains more faces having similar orientations with the current face, which can better describe the local shape of the current face.
- We introduce an ℓ_0 minimization method, incorporating the spectral analysis theory, to refine a shape-aware patch for each face from its corresponding candidate patch. The refined patch can more accurately match the local shape in a manner that does not contain any geometric features.
- We conduct a variety of experiments to show that our denoising method outperforms the state-of-the-art methods on synthetic and raw scanned data.

2. Related works

As a vital tool in computer graphics, mesh denoising has been extensively studied in the past decades. There are rich mesh denoising methods in the literature, categorized into three types: filter-based methods, optimization-based methods, and data-driven methods. It is beyond our scope to review all existing work. Here, we mainly review those techniques that are closely relevant to our work.

Filter-based methods for mesh denoising. The filter-based methods, divided into isotropic and anisotropic ones, have dominated mesh denoising for a long time. The key idea is to filter each mesh vertex or face normal with its local neighborhood, which is straightforward and efficient. However, isotropic methods [9,10] tend to over-smooth geometric features while removing noise. Thus, many anisotropic methods [4,11–14] have been proposed to tackle this problem, which can preserve geometric features well when suppressing noise, However, if meshes are corrupted by large-scale noise, these methods may blur features more or less, especially for those sharp features. In order to improve this situation, [6] presents a guided normal filter, which adopts reliable guidance instead of the original signal in the range kernel of the bilateral filter; [15] presents a high-fidelity method by utilizing Tukey's bi-weight function instead of Gaussian function in the range kernel of the bilateral filter. These two approaches preserve sharp features and suppress large-scale noise effectively. But, the former is sensitive to surface sampling, and the latter over-smoothes fine details. Besides, [16,17] study the bilateral filter with proper guidance signals for more geometry applications, such as geometric texture removal and editing.

Optimization-based methods for mesh denoising. Optimization methods have been introduced for feature-preserving mesh denoising with priors about underlying surfaces. These methods formulate the denoising process as an optimization problem to find the solutions that satisfy the given priors. For example, the sparsity prior has been widely used in methods [1,5,18–22], due to its excellent edge-preserving property [23,24]. As we know, methods [1,5,18,19,22] can recover sharp features, but suffer from undesired staircase effects on smooth features due to the sparsity requirements. To address this issue, [20,25] present a high order regularizer that can preserve both sharp and smooth features; [21] combines total variation and anisotropic Laplacian regularizations. However, these two methods cannot handle large-scale noise effectively. Besides the sparsity prior, the nonlocal similarity prior has been recently explored for mesh denoising [8,26]. The method [8] can effectively remove noise while recovering fine details. But it is limited to handle meshes with sharp features. In contrast, the method [26] can keep sharp features. But it is time-consuming for large-scale models. Besides, [27,28] apply low-rank recovery for point cloud denoising and mesh texture smoothing.

Data-driven methods for mesh denoising. Recently, data-driven approaches have been applied to mesh denoising [7,29–34]. [7] presents a cascade normal regression method that can learn the mapping from noisy inputs to their corresponding ground truth. [30] proposes a two-step method that avoids blurring fine details. [32] presents a deep neural network for smoothing face normals. Besides, [29,31,33,34] adopt convolutional neural networks for mesh denoising. In short, without any assumptions about underlying features and noise patterns, these methods can remove noise effectively while recovering features well. However, the performance of these data-driven methods relies on the completeness of the training dataset.

Graph spectral methods for geometry processing. Due to the success of spectral graph in image processing [35,36], it has been extended to 3D data in [37–39]. For example, in the task of mesh denoising, [39] exploits the fact that sharp features reside in a low dimensional structure hidden in the noisy meshes, by cutting off higher frequencies and attenuating frequencies of the spectrum of a graph. [38] transforms feature detection into a graph-cut problem, and iteratively applies the normalized cut (NCut) algorithm to dependent patches according to those detected features; in the task of mesh segmentation, [37] applies a spectral analysis framework on the mesh to obtain a coarse segmentation result. Inspired by [37], we also adopt spectral partitioning with the Fiedler vector for patch refinement. Our method is based on the observation that, the ordered Fiedler vector can be optimized into a piecewise constant solution, where the coordinate values indicate the segmentation result, and the discontinuities match the underlying geometric features. Fig. 1 shows such examples.

3. Preliminaries

3.1. Joint bilateral normal filtering framework

Our mesh normal filtering method is designed based on the well-known joint bilateral mesh normal filter, which has been frequently used in geometry processing problems [6,16,17,38,40], for its simplicity and effectiveness. Specifically, this framework is an iterative scheme, where each iteration consists of two stages: normal filtering followed by vertex updating. Here, we introduce these two stages in detail as follows:

(i) *Normal filtering*. Given the two variables, **N** and **G**, as the face normal field and guidance face normal field, the joint bilateral normal filtering framework first update **N** based on the following formula:

$$\mathbf{N}_i = \Gamma\Big(\sum_{j\in D(i)} a_j W_d(\|c_i - c_j\|) W_s(\|\mathbf{G}_i - \mathbf{G}_j\|)\mathbf{N}_j\Big), \tag{1}$$

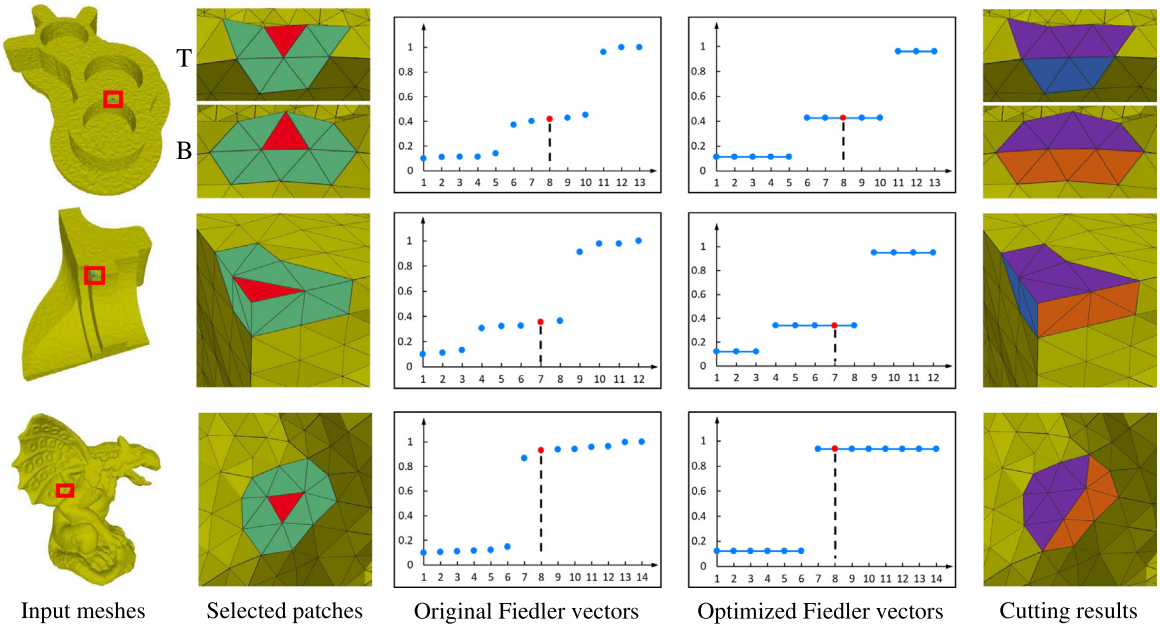

Fig. 1. Shape-aware patch refining via ℓ_0 minimization on the Fiedler vector. All the Fiedler vectors are sorted in increasing order, and each point colored in red represents the central face of the corresponding patch. In the first row, we give the top (T) and bottom (B) side views of the patch for better visualization. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where $\Gamma(\cdot)$ is the normalization operator, $D(i)$ is the 1-ring neighbors of the face i, a_j is the area of face j, c_i, c_j are centroids of faces i and j, and W_d, W_s are two Gaussian functions with kernel sizes σ_d and σ_r, respectively.

(ii) *Vertex updating*. This stage needs to reconstruct vertex positions to match the filtered face normals. Researchers have proposed many vertex updating methods using the orthogonality between the face and its corresponding normal direction [41,42], or by minimizing the difference between the orientation of the face and its corresponding normal direction [21]. In this work, we adopt the vertex updating method presented in [42] for its efficiency handling flipped triangles.

From (1), we can see that the performance of the filtering process relies on guidance face normal field $\mathbf{G}$. The reason is that the proximity between the guidance signal values $\mathbf{G}_i$ and $\mathbf{G}_j$ provides the prediction for the proximity between desired output signals $\mathbf{N}_i$ and $\mathbf{N}_j$. Compared to bilateral mesh normal filter [4], this proximity can preserve geometric features much better. Though many methods [6,16,38] have tried to compute a proper guidance signal within the context of mesh normal filtering, it is still challenging to estimate an accurate guidance field when handling meshes containing corners with low sampling rates, multi-scale features or narrow structure regions. Therefore, in this paper, we focus on computing the shape-aware guidance field from the noisy input, which is capable of providing more reliable normal information about the local shape of the desired result.

3.2. Spectral graph partitioning with fiedler vector

Our method produces shape-aware consistent patches from candidate patches by utilizing spectral graph partitioning with Fiedler vector. Here, we briefly introduce the background of our method, which includes some necessary notations and results.

Let $G = (V, E)$ be a general, undirected, and non-negatively weighted graph, where V and E are the sets of vertices and edges, respectively. Assume $|V|$ be the number of vertices, and $|E|$ be the number of edges. Edges are represented by pairs of vertices (i, j) for $i, j \in V$. The edges are associated with non-negative weights $\omega = \{w(i, j), \forall (i, j) \in E\}$. To capture information of graph G, spectral related methods need to define its adjacency matrix A_G as $A_G(i, j) \triangleq w(i, j)$, its diagonal degree matrix D as $D_{ii} \triangleq \sum_{j \in V} A_G(i, j)$, and its Laplacian matrix L_G as $L_G \triangleq A_G - D_G$. As well known Laplacian matrix L_G is symmetric and positive semi-definite. Assume the spectrum of L_G is denoted by $0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_{|V|}$. Then, the eigenvector corresponding to the smallest nonzero eigenvalue (the second smallest eigenvalue) λ_2 is called the Fiedler vector. The eigensystem of L_G can describe valuable properties of graph G [43]. Some results are given as follows.

Proposition 1 (*[43]*). *The multiplicity k of eigenvalue λ_1 (i.e., 0) equals the number of connected components $G_1, G_2, \ldots, G_k$ in G. Besides, the eigenspace of λ_1 is spanned by indicator vectors $\mathbb{1}_1, \mathbb{1}_2, \ldots, \mathbb{1}_k$ of those components. Formally, the indicator vectors are written as*

$$\mathbb{1}_i \in \mathbb{R}^{|V|} \text{ and } (\mathbb{1}_i)_j = \begin{cases} 1, & \text{if vertex } j \in G_i \\ 0, & \text{otherwise.} \end{cases}$$

In our task, we first construct a graph for each candidate patch, then segment the graph into several subgraphs according to the indicator vectors. Note that each indicator vector corresponds to one subgraph, which can be seen as a connected component.

Suppose that graph G is segmented into k subgraphs $G_1, G_2, \ldots, G_k$, and its vertices are ordered in accordance with the connected components they belong to. Let L_{G_i} be the Laplacian matrix of G_i, $i = 1, 2, \ldots, k$. Suppose that $\breve{G} = \cup_{i=1}^{k} G_i$. Then the Laplacian matrix of $\breve{G}$ is $L_{\breve{G}} = diag(L_{G_1}, L_{G_2}, \ldots, L_{G_k})$. Since graph $\breve{G}$ is derived from graph G by removing the edges with small weights, we can regard the above Laplacian matrix L_G as a perturbation of matrix $L_{\breve{G}}$, i.e.,

$$L_G = L_{\breve{G}} + P,$$

where P is the perturbation matrix. Then, we know that the eigenspace corresponding to the k smallest eigenvalues of L_G is very close to the eigenspace corresponding to the k smallest eigenvalues of $L_{\breve{G}}$ when the perturbation $\|P\|$ is small (i.e., the

eigenvalue λ_1 with multiplicity k). Therefore, by using Davis–Kahan Theorem [43], we can represent the Fiedler vector of Laplacian matrix L_G, called $\mathbf{u}_0$, as

$$\mathbf{u}_0 = c_1\mathbb{1}_1 + c_2\mathbb{1}_2 + \cdots + c_k\mathbb{1}_k + \mathbf{e} \triangleq \mathbf{u} + \mathbf{e}, \tag{2}$$

where $\mathbf{u} \in \mathbb{R}^n$ is a linear combination of the indicator vectors $\mathbb{1}_1, \mathbb{1}_2, \ldots, \mathbb{1}_k$, $c_1, c_2, \ldots, c_k$ are values defined on subgraphs, and $\mathbf{e} \in \mathbb{R}^n$ is a small error vector (a residual vector). Since $\mathbf{u}$ is the linear combination of the indicator vectors, it is a piecewise constant vector. Therefore, if $\mathbf{u}$ can be decomposed from $\mathbf{u}_0$, we can segment graph G by checking the entry values of $\mathbf{u}$. Specifically, for $\forall j \in V$, we have

$$j \in G_i \;\Leftrightarrow\; \mathbf{u}_j = c_i,\; i = 1, 2, \ldots, k.$$

The aforementioned prior knowledge of the Fiedler vector has been proved helpful for mesh segmentation, even in the presence of noise [37]. Inspired by that, we segment each candidate patch into a refined shape-aware patch, which can accurately describe the local shape of the underlying surface.

4. Guided normal filtering via shape-aware patches

Following works [6,17,40,44], we compute each guidance face normal based on a properly constructed patch. For example, given a well-constructed patch P_i of face i, its guidance normal can be computed as:

$$\mathbf{G}_i = \Gamma\Big(\sum_{j\in P_i} a_j\mathbf{N}_j\Big),$$

where a_j is the area of face j. As can be seen, guidance normals are computed based on the constructed patches. Hence, the guidance normal needs to be computed for each face by building a consistent local patch around the face, which does not contain any geometric features. However, how to build a consistent patch for each face in the presence of noise is difficult, because the mesh may contain corners with low sampling rates, multi-scale features, or narrow structure regions. To address these issues, we propose a two-stage consistent patch construction method. In the first stage, we can select a candidate patch for each face by introducing a novel patch consistency measurement. In the second stage, based on spectral analysis theory, we optimize these candidate patches to generate refined patches in a shape-aware manner for estimating more robust guidance normals. Note that, for each face, its candidate patch would be not only as flat as possible, but also have the orientation most similar to it.

4.1. Candidate patch selection

We compute candidate patches under the assumption: the underlying surface of a noisy mesh consists of many piecewise smooth regions, and each feature lies at the intersection of multiple smooth regions. Furthermore, the faces belonging to the same smooth region tend to have similar orientations. With these prior knowledge, we first decompose the input mesh into many overlapping small patches, then select a patch from those as the candidate patch for each face. Details are elaborated as follows.

First, we define the patch as a union of one face with its 1-ring neighbor faces that share vertices with it. Based on this definition, for each face i, we can collect a set of nearby patches $C(i) = \{P_j | i \in P_j\}$. We can observe that, if patch P_j does not contain any geometric features, all the faces of P_j have similar orientations with face i. Thus, for face i, we compute its candidate patch by selecting one patch from $C(i)$ which is as flat as possible. To meet these requirements, we perform a patch-shift procedure on patch set $C(i)$ with the newly defined patch consistency measurement $\mathcal{R}(P_i)$, which is written as

$$\mathcal{R}(P_i) = \mathcal{F}(P_i) \cdot \mathcal{S}(P_i), \tag{3}$$

where $\mathcal{F}(P_i)$ is the flatness term used to estimate surface variations within patch P_i, $\mathcal{S}(P_i)$ is the similarity term for measuring face-to-patch orientation similarity between the current face and patch P_i.

Flatness term $\mathcal{F}(P_i)$: The flatness term $\mathcal{F}(P_i)$, taking both local and global surface variations within patch P_i into account, is designed as:

$$\mathcal{F}(P_i) = F_L(P_i) \cdot F_G(P_i), \tag{4}$$

where $F_L(P_i)$ and $F_G(P_i)$ are two functions defined on the face normal field of patch P_i for measuring its degree of local and global surface variations. To achieve this, $F_L(P_i)$ is defined as the average magnitude of face normal differences over the edges of patch P_i. Formally, it can be written as:

$$F_L(P_i) = \frac{1}{4\sum_{e\in E(P_i)} l_e} \sum_{e\in E(P_i)} l_e \|\mathbf{N}_{e,1} - \mathbf{N}_{e,2}\|_2,$$

where $E(P_i)$ is the set of edges contained in patch P_i, $\mathbf{N}_{e,1}$ and $\mathbf{N}_{e,2}$ are normals of two faces sharing the common edge e, and l_e is the length of edge e. The value of $F_L(P_i)$ can be used to describe the local flatness of patch P_i. Furthermore, considering the accumulation of surface variances, we define global flatness $F_G(P_i)$ as the maximum normal difference between two faces:

$$F_G(P_i) = \max_{j,k\in P_i} \|\mathbf{N}_j - \mathbf{N}_k\|_2.$$

As a result, a small value of $F(P_i)$ indicates that the patch is smooth and does not contain features, while a large value implies the patch may contain extra features.

Orientation similarity term $\mathcal{S}(P_i)$: We further measure the orientation similarity between face i and patch P_i using a newly defined metric $\mathcal{S}(P_i)$, which is called face-to-patch similarity:

$$\mathcal{S}(P_i) = S_L(P_i) \cdot S_G(P_i), \tag{5}$$

where $S_L(P_i)$ and $S_G(P_i)$ are used to describe the local and global orientation similarity between face i and the other faces contained in patch P_i. Specifically, $S_L(P_i)$ measures the local similarity, which directly computes the sum of variations between face i and the other faces. We formulate $S_L(P_i)$ as:

$$S_L(P_i) = \frac{1}{2\sum_{k\in P_i} a_k} \sum_{k\in P_i} a_k \|\mathbf{N}_i - \mathbf{N}_k\|_2.$$

A smaller value of $S_L(P_i)$ means that more faces have orientations similar to face i. Ideally, when $S_L(P_i) = 0$, all the other faces of P_i have the same orientation as face i. That also means patch P_i is a flat region. However, when the $S_L(P_i)$ value is slightly greater than zero, patch P_i may contain small-scale noise or shadow features. Thus, this metric cannot clearly distinguish the above two surface variations. To address this ambiguity, we further introduce metric $S_G(P_i)$ to measure the global similarity, which is written as:

$$S_G(P_i) = \frac{1}{2}\|\mathbf{N}_i - \overline{\mathbf{N}}_{P_i}\|_2,$$

where $\overline{\mathbf{N}}_{P_i}$ is the area-weighted average normal of patch P_i. A lower $S_G(P_i)$ value is likely to indicate the patch with a more similar orientation to face i. Thus, the global similarity amplifies the difference between these two types of patches, which can solve the above ambiguity problem. As a result, by using the similarity term, we can identify the patch with more faces having orientation similar to the current face, and be capable of distinguishing shadow edges from small-scale noise.

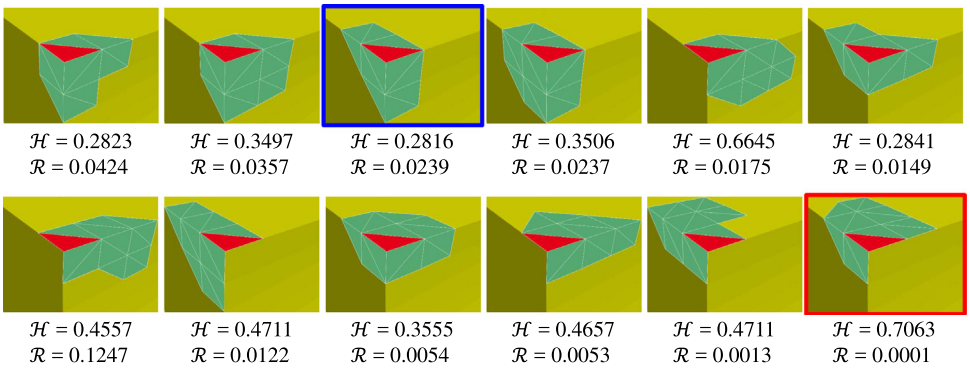

Fig. 2. Comparison of the selected candidate patch using consistency measurement $\mathcal{H}$ in [6] and $\mathcal{R}$ proposed in this paper. The current face is colored in red. For each patch of current face, we calculate its corresponding two measurement values of $\mathcal{H}$ and $\mathcal{R}$. The patch bounded with a blue box is the candidate patch identified using the measurement $\mathcal{H}$, while the patch bounded with a red box is the candidate patch selected using our consistency measurement $\mathcal{R}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

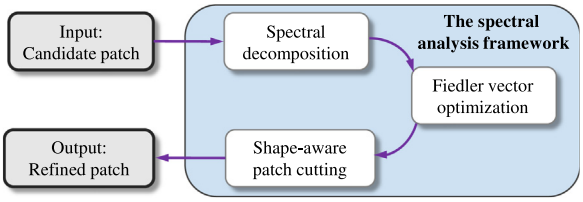

Fig. 3. The pipeline of the proposed spectral analysis framework.

Remark. By combining flatness term $\mathcal{F}(\cdot)$ and orientation similar term $\mathcal{S}(\cdot)$, the proposed patch consistency measurement $\mathcal{R}(\cdot)$ can be used to select the candidate patch with the smallest value of $\mathcal{R}(\cdot)$. The selected patch contains as few geometric features as possible. Compared to the patch consistency measurement $\mathcal{H}(\cdot)$ proposed in [6], our measurement is more robust in handling meshes with complicated geometric features. The reasons are as follows. Although $\mathcal{H}(\cdot)$ can estimate the global flatness of a patch, it cannot fully measure the local flatness. The reason is that $\mathcal{H}(\cdot)$ computes the maximum edge saliency instead of the sum of the local surface variations. A comparison of selecting the candidate patch using these two metrics is demonstrated in Fig. 2. As can be seen, the candidate patch selected using $\mathcal{R}(\cdot)$ contains fewer features than the patch selected by $\mathcal{H}(\cdot)$. On the other hand, due to the proposed face-to-patch similarity $\mathcal{S}(\cdot)$, we select the candidate patch containing as many faces as possible that have orientations similar to the current face. Thus, by using $\mathcal{S}(\cdot)$, the selected candidate patch can provide more faces to compute the guidance normal, making the guidance normal to be more accurate.

4.2. Shape-aware patch refinement

Due to the worse-case mesh quality or topology, although we have obtained the candidate patch for each face, some of these patches still contain geometric features that need to be preserved; see Fig. 2 for example. As a result, the guidance signal, directly estimated from these patches, may blur features or even destroy geometric structures. To circumvent this limitation, we further present a shape-aware patch refinement method using the spectral analysis framework. The key idea is that, for each face i, we cut its candidate patch into multiple sub-patches according to the shape of the underlying surface, and choose the sub-patch containing face i as its shape-aware patch that does not contain any geometric features. The pipeline of our spectral analysis framework is shown in Fig. 3. Details of each module in the pipeline are elaborated as follows.

Algorithm 1: Fiedler vector optimization

Input: Fiedler vector $\hat{\mathbf{x}}'$;
Output: Optimized Fiedler vector $\hat{\mathbf{x}}$;
Initialization: $\beta = 10^{-3}$;
repeat
  fix $\hat{\mathbf{x}}$, solve for $\mathbf{z}$ in (9).
  fix $\mathbf{z}$, solve for $\hat{\mathbf{x}}$ in (10).
  $\beta = 2 \times \beta$.
until $\beta \geq 10^3$;

Spectral decomposition. We perform spectral analysis on each selected candidate patch. We first build a proper Laplacian matrix of each patch reflecting its local geometric and topological attributes, and then generate the refined patch by optimizing the Fiedler vector of the Laplacian matrix. In the following, we elaborate on spectral analysis of each candidate patch.

Now we construct the Laplacian matrix, which serves as the primary tool for spectral analysis. First, we construct dual graph $G_d = (V_d, E_d)$, where vertex set V_d is the face set of patch P, and E_d is the set of dual edges, i.e., every pair of adjacent faces in patch P corresponds to an edge of G_d. Note that, we build the Laplacian matrix based on faces instead of vertices, since first-order face normal variations can better describe surface variations than vertex position variations [2]. Then, we construct the Laplacian matrix of G_d. The key is to define the weights associated with edges E_d. To do that, we measure the difference between a pair of adjacent faces i, j as

$$d(i,j) = \frac{\xi}{2}\|\mathbf{N}_i - \mathbf{N}_j\|_2^2,$$

where $\mathbf{N}_i$ and $\mathbf{N}_j$ are two normals of faces i and j, ξ is a positive weight. In this work, we typically set $\xi = 1.0$, if $\angle(\mathbf{N}_i, \mathbf{N}_j) \geq \sigma_\theta$; otherwise, $\xi = 0.1$. σ_θ is a user-specified angle threshold for recognizing features from noise. Let $\bar{d}$ be the average value of all the differences. Let e be the common edge of a pair of adjacent faces i, j. We define the weight associated with edge e as

$$w(i,j) = l_e \exp(-d(i,j)/\bar{d}).$$

After obtaining all the weights, we further compute Laplacian matrix L_{G_d} of graph G_d, by the way introduced in Section 3.2. By performing eigen-decomposition on Laplacian matrix L_{G_d}, we can obtain its spectrum, i.e., eigenvalues.

Fiedler vector optimization. Let the Fiedler vector of Laplacian matrix L_{G_d} be $\mathbf{x}' = \{x'_1, x'_2, \ldots, x'_{|P|}\}$, where $|P|$ is the number of

faces contained in patch P. According to (2), we have

$$\mathbf{x}' = \mathbf{x} + \mathbf{e}_p,$$

where $\mathbf{x}, \mathbf{e}_p \in \mathbb{R}^{|P|}$ are piecewise constant vector and error (residual) vector. In order to recover $\mathbf{x}$ from $\mathbf{x}'$, we optimize $\mathbf{x}'$ using ℓ_0 minimization. However, it is time-consuming to directly solve ℓ_0 minimization problem over the mesh. Thus, for better efficiency, we use one approximation strategy as follows. First, we sort the elements of the Fiedler vector $\mathbf{x}'$ in increasing order, and denote the sorted vector as

$$\hat{\mathbf{x}}' = \{\hat{x}'_1, \hat{x}'_2, \ldots, \hat{x}'_{|P|}\}.$$

Then $\hat{\mathbf{x}}'$ is a 1-dimensional vector, where similar values have been grouped together. Then, we induce the sparsity on $\hat{\mathbf{x}}'$, and formulate the problem as

$$\min_{\hat{\mathbf{x}}} \frac{1}{2}\|\hat{\mathbf{x}} - \hat{\mathbf{x}}'\|_2^2 + \lambda \sum_{i=1}^{|P|-1} |\hat{x}_{i+1} - \hat{x}_i|_0, \tag{6}$$

where $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{|P|})^{\mathrm{T}}$. The first term of (6) is a fidelity term minimizing the approximation error between $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$, the second term is a regularization term measuring the sparsity of the vector, and λ is a positive parameter balancing these two terms.

Due to the nondifferentiability of problem (6), it is challenging to directly solve it. The studies [1,23] show that variable-splitting and alternating optimization solver have achieved great success in solving ℓ_0 related problem. Here, we first introduce auxiliary variable $\mathbf{z} = \{z_1, z_2, \ldots, z_{|P|-1}\}$, and reformulate (6) as

$$\min_{\hat{\mathbf{x}},\mathbf{z}} \frac{1}{2}\|\hat{\mathbf{x}} - \hat{\mathbf{x}}'\|_2^2 + \lambda \sum_{i=1}^{|P|-1} |z_i|_0 + \beta \sum_{i=1}^{|P|-1} |\hat{x}_{i+1} - \hat{x}_i - z_i|^2. \tag{7}$$

The above problem can be solved using alternating optimization. First, we fix $\hat{\mathbf{x}}$ constant and perform minimizing for $\mathbf{z}$ as

$$\min_{\mathbf{z}} \lambda \sum_{i=1}^{|P|-1} |z_i|_0 + \beta \sum_{i=1}^{|P|-1} |\hat{x}_{i+1} - \hat{x}_i - z_i|^2. \tag{8}$$

This problem is easy to solve since (8) can be spatially separated, where the minimization problem with respect to each first-order variation is performed individually. Thus, for each z_i, we just need to solve the following problem

$$\min_{z_i} \lambda |z_i|_0 + \beta |\hat{x}_{i+1} - \hat{x}_i - z_i|^2. \tag{9}$$

In this minimization, each entry z_i will either be 0 or $\hat{x}_{i+1} - \hat{x}_i$. When $\sqrt{\frac{\lambda}{\beta}} > \hat{x}_{i+1} - \hat{x}_i$, $z_i = 0$; otherwise $z_i = \hat{x}_{i+1} - \hat{x}_i$. Next, we hold $\mathbf{z}$ fixed and optimize $\hat{\mathbf{x}}$ by solving the following problem

$$\min_{\hat{\mathbf{x}}} \frac{1}{2}\|\hat{\mathbf{x}} - \hat{\mathbf{x}}'\|_2^2 + \beta \sum_{i=1}^{|P|-1} |\hat{x}_{i+1} - \hat{x}_i - z_i|^2. \tag{10}$$

This is a quadratic problem, which can be solved by existing sparse linear solvers, such as MKL and Taucs. The two optimizations (8) and (10) alternate until convergence, and parameter β is multiplied by 2 in each iteration to eventually force z_i to match $\hat{x}_{i+1} - \hat{x}_i$. The whole procedure is outlined in Algorithm 1.

Shape-aware patch cutting. The solution $\hat{\mathbf{x}}$ is a piecewise constant vector due to its sparsity prior. We segment each candidate patch into some sub-patches according to the entry value of $\hat{\mathbf{x}}$, and then select the sub-patch that contains the current face as the refined patch; see Fig. 1 for example.

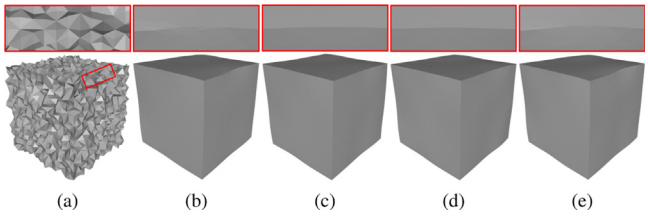

Fig. 4. Denoising results for different σ_θ with other parameters fixed. From left to right: noisy input ($\sigma = 0.5\bar{l}_e$), and results with $\sigma_\theta = 0.01\pi, 0.1\pi, 0.4\pi, \pi$.

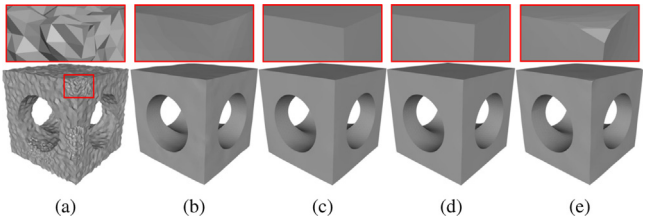

Fig. 5. Denoising results for different λ with other parameters fixed. From left to right: noisy input ($\sigma = 0.15\bar{l}_e$), and results with $\lambda = 0.01, 0.4, 0.7, 10$.

5. Experimental results

In this section, we report numerical experiments on various kinds of CAD, nonCAD and raw scanned meshes. The synthetic noise is generated by a zero-mean Gaussian function with standard deviation proportional to the mean edge length of the ground truth shape. We denote this standard deviation as σ. We compare the proposed method with seven state-of-the-arts including local bilateral filtering [4], ℓ_0 minimization [1], robust and high fidelity mesh denoising [15], low-rank based normal filtering [8], cascade normal filtering [7], and guided normal filtering [6], abbreviated as LBF, L0, RHM, LRNF, CNR, and GNF. For RHM, LRNF, CNR, and GNF, we run the codes provided by their authors; For the other three methods, we have implemented them based on their published articles in C++. To show the faceting effect, all of the examples are rendered in the flat-shading model.

5.1. Parameter choosing

Like previous methods, to generate satisfactory results, our method also needs to tune parameters including σ_θ, α, σ_d, and σ_r. Since Gaussian kernel sizes σ_d and σ_r are parameters of the joint bilateral filter, their effects on filtering results have been fully explored in previous works [4,6]. Thus, we only discuss the effects of the first two parameters on final denoising results.

σ_θ is used to construct Laplacian matrices. The value of σ_θ is highly related to feature recognition. Fig. 4 shows results of different σ_θ with other parameters fixed. As we can see, if σ_θ is too small or too large, sharp features will be blurred; see Figs. 4b and 4e. This is because that, if σ_θ is too small or too large, most edges will be recognized as features or non-features, and our algorithm cannot identify the shape of the underlying surface correctly. Besides, there is a range of σ_θ for our algorithm to produce visually good results, shown in Figs. 4c and 4d. Empirically, we set $\sigma_\theta = 0.1\pi$ in all the experiments of this paper for generating visually appealing results.

λ is used to constrain the sparsity of the Fiedler vector. Fig. 5 shows results of different λ with other parameters fixed. As we can see, if λ is too small, our method cannot recover the flat regions well; see Fig. 5b. The reason is that, most faces have the different values from the current face in the optimized Fiedler vector. On the contrary, if λ is too large, the corner feature is

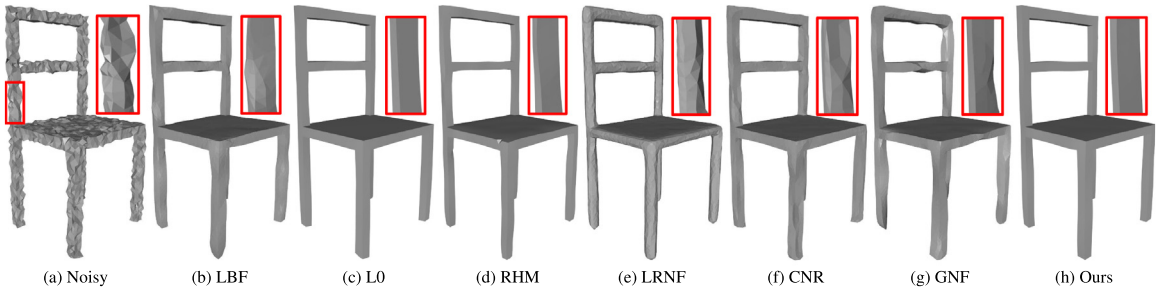

Fig. 6. Denoising results of Chair ($\sigma = 0.2\bar{l}_e$). The zoomed views highlight that our method is more effective in keeping corners with low sampling rates and narrow structure regions.

destroyed; see Fig. 5e. The reason is that the optimized patch is actually the input candidate patch, since the Fiedler vector will be optimized into a constant vector with strong sparsity constraints. Similarly, there is also a range of λ for our algorithm to produce visually good results, shown in Figs. 5c and 5d.

5.2. Qualitative comparisons

First of all, we compare our method with the mentioned state-of-the-arts on CAD and nonCAD meshes as well as raw scanned meshes. We carefully tune the parameters of each method for producing visually appealing results.

Fig. 6 demonstrates results on Chair containing corners with low sampling rates and narrow structure regions. As can be seen, all the methods can remove noise effectively. However, LBF, RHM and CNR blur the sharp features in vary degrees, due to that these methods are designed based on the bilateral filter, which cannot predict the desired features at corners and narrow structure regions well as seen in Figs. 6b, 6d and 6f. Similarly, LRNF also fails to keep sharp features at narrow structure regions, because it uses a simple average conversion between vertex normals and face normals; see Fig. 6e. GNF also blurs these sharp features as Fig. 6g shows. The reason is that, GNF uses fixed-size patches, which leads to failure in the estimation of the reliable guidance normals at corners with low sampling rates and narrow structure regions. Unlike the above methods, L0 and ours can keep all the sharp features; see Figs. 6c and 6h. This is because that L0 benefits from the edge-preserving property of sparsity norms, and our method uses shape-aware guidance normals. However, from quantitative comparison in the next subsection, we can see that the error of our method is smaller than that of L0.

Fig. 7 demonstrates results on Table and Casting, both of which contain narrow structure regions and smooth features. Obviously, RHM, LBF, LRNF and CNR recover smooth features well. However, they blur sharp features in various degrees; see Figs. 7b, 7d, 7e and 7f. The reasons are as follows. RHM, LBF, and CNR are based on the bilateral filter, which cannot accurately predict the geometric features of underlying surfaces. This situation is more serious in the case of large noise. Though RHM adopts a robust statistics framework for preserving sharp features, it is hard to recover all the features with a hard threshold. Besides, from zoomed view in Fig. 7g, GNF destroys some sharp features, because it hardly estimates reliable guidance normals at narrow structure regions. On the contrary, though L0 keeps sharp features well, it generates obvious staircase effects on smooth features, because of its highest sparsity requirements; see Fig. 7c. Different from the above methods, our method can not only restore sharp features at narrow structure regions, but also keep smooth features well as seen in Fig. 7h.

Fig. 8 shows results on Fandisk containing corners with low sampling rates and complicated narrow structure regions. As can be seen that all methods can effectively remove noise. Similarly to the previous examples, LBF, LRNF, and CNR blur features at corners or narrow structure regions; see Figs. 8b, 8e, and 8f. On the contrary, L0 and RHM can keep sharp features. Nevertheless, L0 inevitably produces staircase effects in smooth features, and RHM suffers from relatively large shape distortions; see Figs. 8c and 8d. Besides, although GNF can keep sharp features at narrow structure regions well, it blurs corners; see Fig. 8g. Compared to these methods, our method produces the visually best result with all the sharp features well preserved; see Fig. 8h.

Fig. 9 shows results on Gargoyle and Lion, both of which contain multi-scale features. As can be seen, LBF over-smoothes all the features; seen the zoomed view in Fig. 9b. In contrary, L0 sharpens small-scale features, and generates staircase effects in large-scale features; see Fig. 9c. Though RHM, LRNF, and CNR recover large-scale features well, they over-smooth small-scale features; see Figs. 9d, 9e, and 9f. Besides, GNF sharpens these small-scale features; see Fig. 9g. Compared to the above methods, our method keeps both large-scale and small-scale features well; see Fig. 9h.

Fig. 10 shows results on Max-planck with irregular sampling rates. As we can see that, most methods can remove noise effectively, except the CNR which remains some noise on its denoising result. For other methods, LBF, RHM, LRNF, and GNF blur features in vary degrees; see Figs. 10b, 10d, 10e, and 10f. Besides, L0 generates staircase effects on smooth features; see Fig. 10c. Compared to these methods, our method produces the best results with most features preserved; see Fig. 10h.

Moreover, in Fig. 11, we show results on Hand acquired by Laser scanners. As can be seen that, all the methods can eliminate noise effectively. Similarly, LBF, RHM, and CNR over-smooth fine details in varying degrees; see Figs. 11b, 11d, and 11f. L0 sharpens the geometric details; see Fig. 11c. Besides, LRNF and GNF as well as our method can preserve fine details, especially our method recovers the shadow edge better; see Figs. 11e, 11g, and 11h. Thus, for laser-scanned meshes, our method also produces the more appealing result compared to the other methods.

Recently, lots of meshes have been acquired using consumer-grade scanner devices, e.g., Microsoft Kinect. So we also compare our method with the state-of-the-arts on this type of data, and show the results in Fig. 12. From these results, we can observe that some bumps still exist on the results of LBF, RHM, LRNF, and CNR; see Figs. 12b, 12d, 12e, and 12f. Compared to these results, the results of L0, GNF and our method are more smooth; see Figs. 12c, 12g, and 12h. However, L0 suffers the staircase effects on smooth features. Besides, both GNF and our method can generate visually satisfactory results with well-recovered smooth features; see Figs. 12g and 12h. However, from the quantitative comparison in the next subsection, we can find that the error of our method is small than that of GNF. Thus, this example shows the effectiveness of our method in dealing with Kinect-scanned meshes.

In the next two paragraphs, we demonstrate the performances of our method against different levels of noise and two kinds

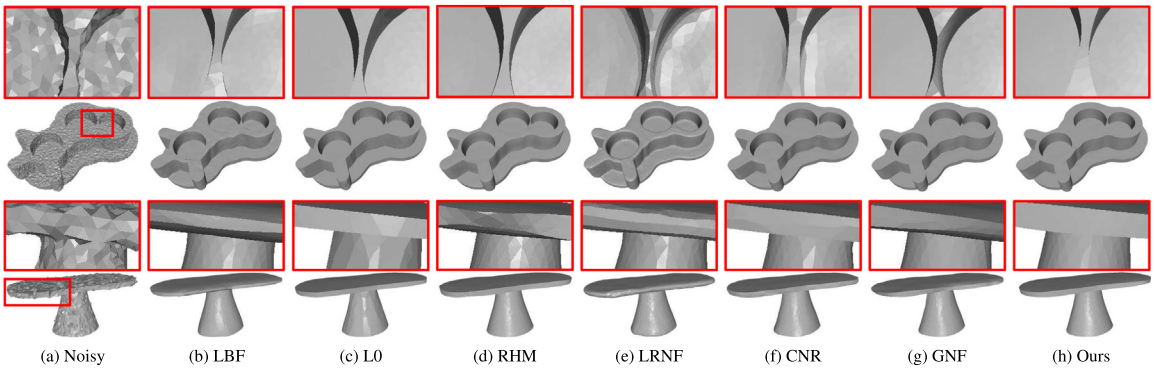

Fig. 7. Denoising results of Table ($\sigma = 0.25\bar{l}_e$) and Casting ($\sigma = 0.2\bar{l}_e$). The zoomed views highlight that our method better keeps sharp features at narrow structure regions as well as smooth features.

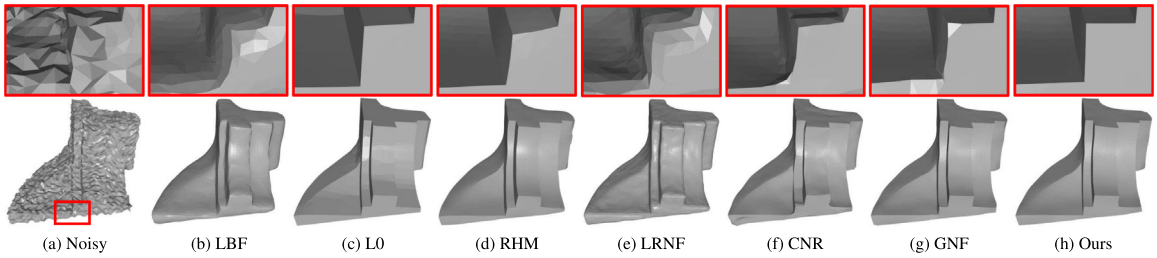

Fig. 8. Denoising results of Fandisk ($\sigma = 0.3\bar{l}_e$). The zoomed views highlight that our method is faithfully able to preserve sharp features at corners and complicated narrow structure regions.

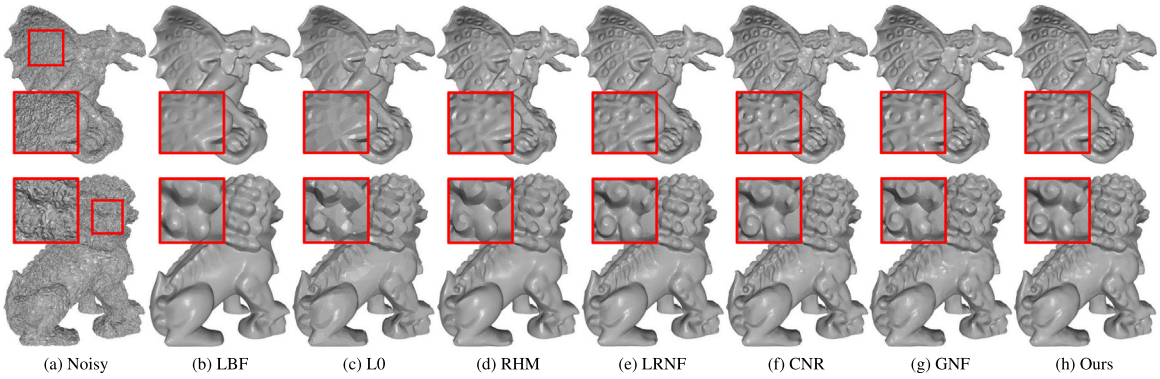

Fig. 9. Denoising results of Gargoyle ($\sigma = 0.35\bar{l}_e$) and Chinese lion ($\sigma = 0.3\bar{l}_e$). The zoomed views highlight that our method better keeps multi-scale features.

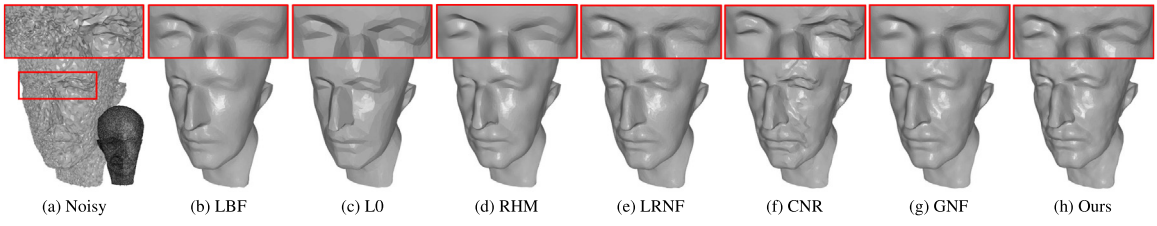

Fig. 10. Denoising results of Max-planck ($\sigma = 0.3\bar{l}_e$). The zoomed views highlight that our method better handles the mesh with irregular sampling rates.

of noise including impulsive noise and mix Gaussian-impulsive noise.

Figs. 13 and 14 show the robustness of our method and the state-of-the-arts against impulsive noise and mixed Gaussian-impulsive noise. Our method can effectively suppress impulsive noise and mixed Gaussian-impulsive noise, which demonstrates the capacity of our method for dealing with different kinds of noise. Our method can simultaneously recover sharp and smooth features while the other competing methods cannot; see Fig. 13. Moreover, as shown in Fig. 14, our method can recover the most geometric details of the underlying surface, while the other competing methods blur these geometric details in varying degrees.

Fig. 15 shows the performances of our method against different levels of noise. As can be seen in Figs. 15a, 15b, and 15c, our method effectively removes noise while preserving sharp features. However, when the noise level is too high, our method

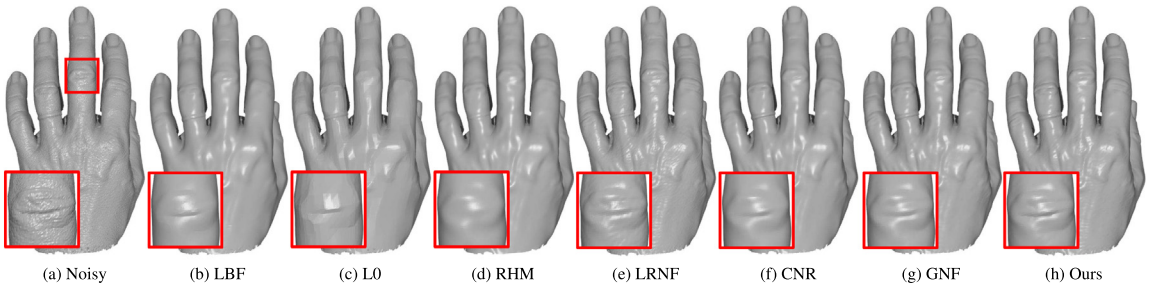

Fig. 11. Denoising results of Hand acquired by Laser scanners.

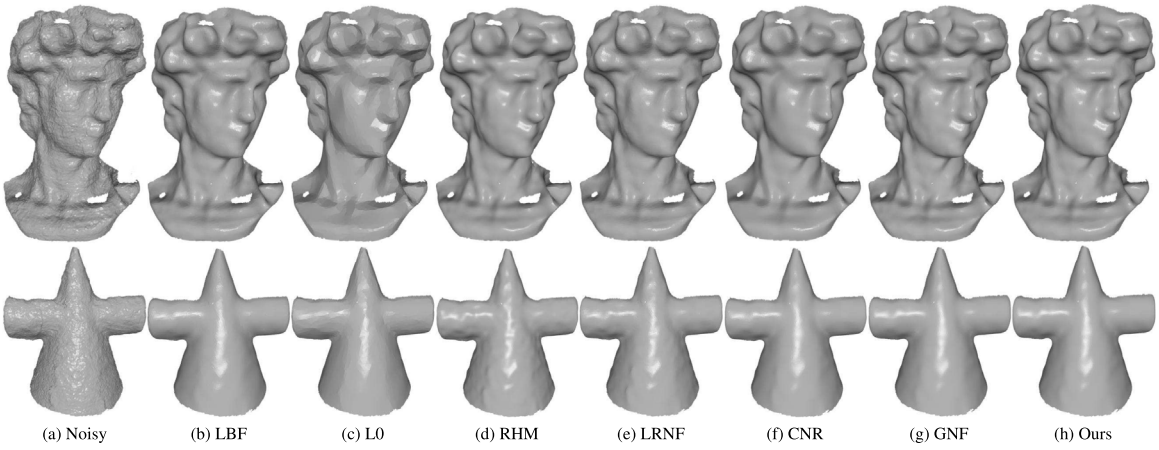

Fig. 12. Denoising results of David and Cone, both of which are scanned by Kinect.

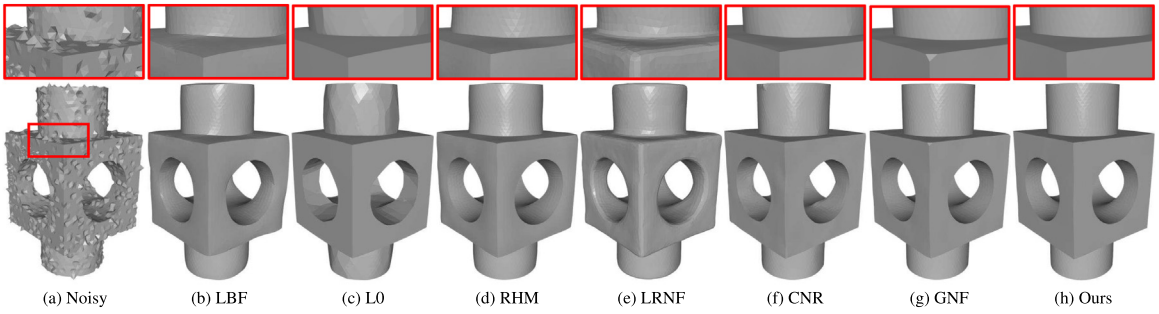

Fig. 13. Denoising results of Block corrupted by impulsive noise (20% impulsive noise with $\sigma = 0.5\bar{l}_e$).

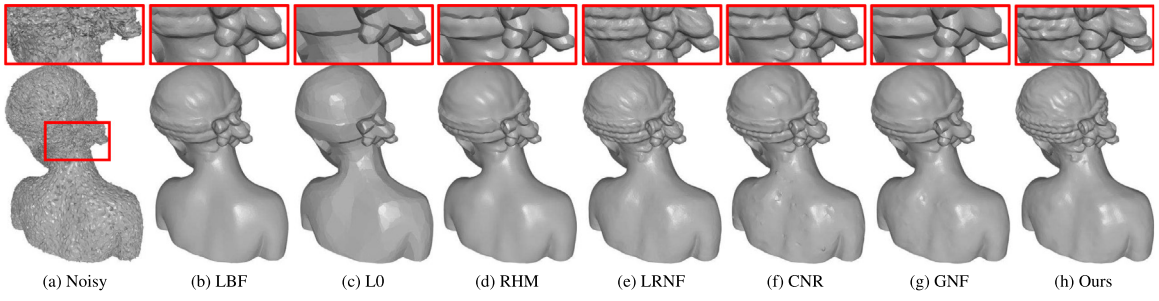

Fig. 14. Denoising results of Child with mixed Gaussian-impulsive noise (20% of impulsive noise with $\sigma = 0.5\bar{l}_e$ and Gaussian noise with $\sigma = 0.2\bar{l}_e$).

cannot faithfully produce a satisfactory result; see Fig. 15d for example.

It is worthwhile to compare our method with GGNF proposed by Zhao et al. [38]. Their method consists of two stages: graph-based feature detection and feature-aware guided normal filtering. Specifically, their method first needs to iteratively perform normalized cut algorithm on patches to detect features. Although their method can detect the geometric features of the underlying surface effectively, it is computational expensive. Besides, in Fig. 16, we visually compare the denoising results of our

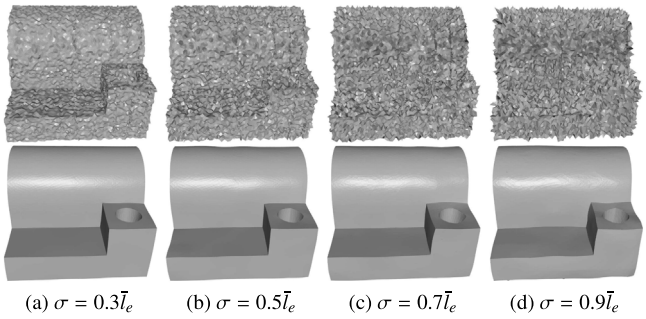

Fig. 15. Denoising results of Joint corrupted by different levels of noise. The first row shows noisy meshes, and the second row shows the corresponding results generated by our method.

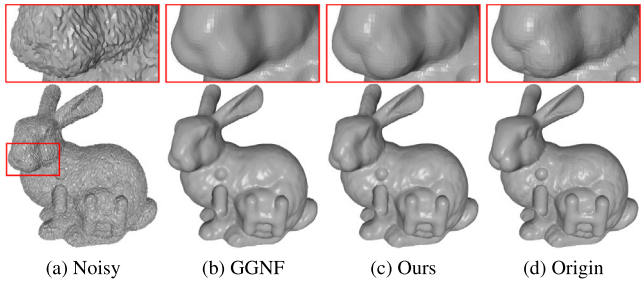

Fig. 16. Denoising results of BunnyIH. From left to right: noisy mesh, results produced by the method in [38] and ours, and the original mesh.

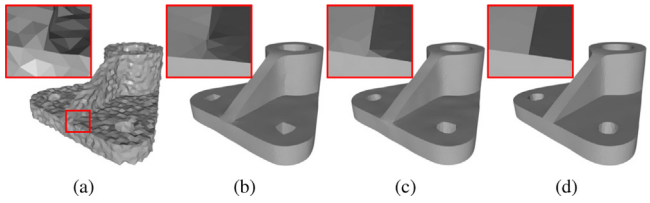

Fig. 17. Denoising results produced based on three types of patches. From left to right: the noisy input ($\sigma = 0.2\bar{l}_e$), and results produced based on CAP, TVP, and SAP patches.

method and GGNF. As we can see, our method recover small-scale details better than GGNF.

5.3. Quantitative comparisons

The above visual comparisons demonstrate that our method can produce better results than the compared state-of-the-arts. Here, for an objective evaluation, we further quantitatively compare our method to others on synthetic data. To achieve this, two error metrics, including mean square angular error (MSAE) and L_2 vertex-based error ($E_{v,2}$), are utilized to measure the fidelity of the denoising result compared to the ground truth. These two measurements are designed with respect to face normals and vertex positions. Specifically, MSAE is used to estimate the mean square angular error between face normals of the denoising result and the corresponding ground truth shape, while $E_{v,2}$ measures the positional error between the clean mesh and the denoised one. More details about these two metrics can be found in the works [4,5]. We compare our method to the other methods using these two metrics on the comparison examples shown in the previous subsection, except for Hand which does not have corresponding ground truth. The comparison results are demonstrated in Table 1. As we can see, compared to the other methods, our method has the smallest MSAE values on the most examples except for Fandisk, where the MSAE value of our method is closest to the best one. Similarly, for most cases, the $E_{v,2}$ values of our method are smaller than the compared state-of-the-arts. Thus, these quantitative comparisons show that, our method still outperforms the compared state-of-the-art methods.

Additionally, we list the mesh sizes and runtime for all the testing methods in Table 1. Moreover, note that the guidance normal construction processes of different faces are separable, hence it can be parallel processed on multiple cores for further speed up, we also implement a fast version of our method and list its runtime in the last column. As can be seen, LBF is always faster than other methods, due to its simplicity. Since our method need to iteratively decompose Laplacian matrices and solve ℓ_0 minimization problems, the cost of our method is relatively higher than the other methods. Fortunately, the fast version of our method is not only able to achieve 3–4 times speed up than the original version, but also faster than most of the other methods. Nevertheless, the computational time of our method is acceptable.

5.4. Comparisons of several patch construction strategies

As we known that, the performance of joint bilateral filter is limited by guidance signals, which should accurately indicate the geometric features of the underlying surface of the noisy mesh. Many researchers have computed guidance signals using patch-based strategies [6,17,40]. These strategies are designed by combining the patch-shift scheme and some novel patch consistency measurement. For example, Zhang et al. [6] propose the well-known patch construction strategy which combines the patch-shift algorithm with the modified relative total variation (mRTV). Different from the existing strategies, for each face, the proposed patch construction strategy first computes a candidate patch based on the patch-shift algorithm and a newly proposed patch consistency measurement, then refines the candidate patch by a shape-aware operation. Here, we discuss the effectiveness of the proposed patch construction strategy.

Fig. 17 demonstrates a comparison of denoising results, which are generated by applying the joint bilateral filter with multiple guidance signals estimated using three types of patch, including the candidate patch, the patch generated by algorithm in [6], and the proposed shape-aware patch. For brevity, we denote these types of patch as CAP, TVP, and SAP. Note that, for a fair comparison, we also refine the TVP patches using the proposed shape-aware operation. As can be seen that, the sharp features of the result, obtained based on the CAP patches, are blurred a lot; see Fig. 17b. This means, due to the irregular sampling of the input mesh, some CAP patches still contain one or more features. As a result, the guidance signals estimated based on CAP patches cannot clearly indicate the geometric features on the irregular sampled regions of the underlying surface. Fig. 17c shows the denoising results generated based on the TVP patches. Obviously, the sharp features are blurred too. This is because that, after the shape-aware operation, some TVP patches cannot provides sufficient number of triangles for robust estimation of the guidance signals. Compared to the above two results, the result computed using the proposed SAP patches preserves all the sharp features well as seen in Fig. 17d.

6. Conclusion

In this paper, we present a shape-aware mesh denoising approach based on the joint bilateral filtering framework, which applies a bilateral filter to smooth face normals, followed by vertex reconstruction to match the filtered face normals. Because

Table 1
Quantitative evaluation results. Note that (·) are the costs of the speeded version of our method.

Mesh	Size($\|V\|$; $\|F\|$)	MSAE($\times 10^{-3}$), $E_{v,2}$($\times 10^{-3}$); Time (in Seconds)						
		LBF	L0	RHM	LRNF	CNR	GNF	Ours
Chair	(3.3k; 6.6k)	28.1, 5.10; 0.08	17.4, 4.34; 1.03	23.9, 3.43; 2.58	112, 10.6; 1.53	20.0, 4.81; 0.71	101.9, 11.5; 0.63	**11.7**, **3.34**; 2.31 (0.63)
Table	(4.6k; 9.1k)	38.1, 5.73; 0.09	13.4, 2.90; 1.37	8.50, 3.83; 3.91	35.8, 3.26; 2.26	22.6, 3.16; 0.80	33.2, 4.85; 0.87	**6.80**, **1.84**; 3.43 (1.01)
Casting	(19.3k; 38.7k)	8.07, 1.25; 0.35	6.67, 1.93; 12.4	5.23, 1.35; 15.2	46.4, 2.10; 8.65	4.64, 1.56; 1.86	8.83, 2.42; 3.29	**4.09**, **0.83**; 10.4 (2.29)
Fandisk	(6.4k; 12.9k)	47.1, 6.13; 0.06	21.3, 5.15; 4.21	13.5, 3.49; 4.94	46.8, 4.78; 2.35	**11.2**, 4.22; 0.88	16.1, 3.99; 1.05	12.8, **3.05**; 3.81 (0.85)
Gargoyle	(85.5k; 171.1k)	91.4, 4.88; 1.45	72.4, 3.11; 47.3	65.1, 1.83; 97.9	43.3, 3.11; 49.0	60.4, 1.77; 9.10	75.9, 1.74; 22.9	**38.1**, **1.16**; 24.2 (6.15)
Lion	(50.0k; 100.0k)	85.2, 3.60; 0.96	69.5, 2.19; 27.5	100.2, 2.35; 55.2	30.3, 1.30; 25.8	39.4, 1.46; 5.79	56.9, 1.51; 7.86	**25.5**, **1.17**; 19.9 (6.35)
Max–Planck	(50.0k; 100.0k)	41.9, 2.65; 0.91	41.5, 2.16; 32.3	32.9, 2.24; 49.8	31.4, 2.22; 24.5	32.5, 2.04; 4.38	40.9, 2.41; 13.4	**27.5**, **1.92**; 27.7 (8.06)
David	(51.8k; 102k)	108, 7.28; 1.28	94.6, 7.03; 22.6	87.2, **5.94**; 37.7	96.0, 6.24; 22.2	99.3, 6.20; 4.69	91.9, 6.29; 14.1	**81.3**, 6.14; 30.2 (14.5)
Cone	(31.1k; 61.3k)	70.7, 9.96; 0.50	58.7, 9.31; 23.0	63.0, **8.43**; 21.6	65.3, 8.78; 10.4	67.7, 8.61; 2.61	64.8, 9.02; 19.5	**55.9**, 8.87; 43.4 (13.1)
Block	(8.7k; 17.5k)	19.7, 5.83; 0.26	30.4, 6.04; 6.11	7.61, 4.37; 6.94	52.8, 5.05; 3.72	6.52, 4.35; 1.04	12.5, 2.40; 1.53	**4.92**, **2.18**; 7.64 (2.31)
Child	(50k; 100k)	60.1, 1.46; 0.94	99.1, 2.42; 32.6	57.1, 1.15; 56.9	33.0, 0.94; 25.6	47.8, 1.17; 5.51	67.6, 1.15; 12.4	**27.7**, **0.93**; 19.6 (6.23)

the performance of joint bilateral filter highly depends on the quality of the guidance signal, we propose a novel patch construction strategy for producing a shape-aware guidance normal field. Our guidance normal field can robustly indicate the underlying surface features in the presence of noise, even for the meshes containing corners with low sampling rates, multi-scale features, or narrow structure regions. As demonstrated in extensive experimental results, our method outperforms the state-of-the-art approaches in preserving sharp and multi-scale features. For future research, we plan to extend our work to handle more comprehensive problems, such as bas-relief modeling, point cloud denoising, and urban modeling.

Declaration of competing interest

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.

Acknowledgments

This work was supported by NSF of China (Nos. U1711267, 41671400, and 62072422), National Key R&D Program of China (Nos. 2017YFC0601500, 2017YFC0601504), NSF of Anhui Province, China (No. 2008085MF195), and Zhejiang Lab (No. 2019NB0AB03).

References

[1] He L, Schaefer S. Mesh denoising via ℓ_0 minimization. ACM Trans Graph 2013;32(4):1–8.
[2] Wei M, Yu J, Pang W-M, Wang J, Qin J, Liu L, Heng P-A. Bi-normal filtering for mesh denoising. IEEE Trans Vis Comput Graphics 2015;21(1):43–55.
[3] Yadav SK, Reitebuch U, Polthier K. Mesh denoising based on normal voting tensor and binary optimization. IEEE Trans Vis Comput Graphics 2017;24(8):2366–79.
[4] Zheng Y, Fu H, Au O-C, Tai C-L. Bilateral normal filtering for mesh denoising. IEEE Trans Vis Comput Graphics 2011;17(10):1521–30.
[5] Zhang H, Wu C, Zhang J, Deng J. Variational mesh denoising using total variation and piecewise constant function space. IEEE Trans Vis Comput Graphics 2015;21(7):873–86.
[6] Zhang W, Deng B, Zhang J, Bouaziz S, Liu L. Guided mesh normal filtering. Comput Graph Forum 2015;34(7):23–34.
[7] Wang P-S, Liu Y, Tong X. Mesh denoising via cascaded normal regression. ACM Trans Graph 2016;35(6):232:1–232:12.
[8] Li X, Zhu L, Fu C-W, Heng P-A. Non-local low-rank normal filtering for mesh denoising. Comput Graph Forum 2018;37(7):155–66.
[9] Taubin G. A signal processing approach to fair surface design. In: Proceedings of the 22nd annual conference on computer graphics and interactive techniques. 1995, p. 351–8.
[10] Desbrun M, Meyer M, Schröder P, Barr A-H. Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proceedings of the 26th annual conference computer graphics and interactive techniques. 1999, p. 317–24.
[11] Wang C. Bilateral recovering of sharp edges on feature-insensitive sampled meshes. IEEE Trans Vis Comput Graphics 2006;12(4):629–39.
[12] Jones T-R, Durand F, Desbrun M. Non-iterative, feature-preserving mesh smoothing. ACM Trans Graph 2003;22(3):943–9.
[13] Fleishman S, Drori I, Cohen-Or D. Bilateral mesh denoising. ACM Trans Graph 2003;22(3):950–3.
[14] Li T, Liu W, Liu H, Wang J, Liu L. Feature-convinced mesh denoising. Graph Models 2019;101:17–26.
[15] Yadav S, Reitebuch U, Polthier K. Robust and high fidelity mesh denoising. IEEE Trans Vis Comput Graphics 2018;PP(99):1.
[16] Wang P-S, Fu XM, Liu Y, Tong X, Liu SL, Guo B. Rolling guidance normal filter for geometric processing. ACM Trans Graph 2015;34(6):1–9.
[17] Wei M, Feng Y, Chen H. Selective guidance normal filter for geometric texture removal. IEEE Trans Vis Comput Graphics 2020;1.
[18] Wu X, Zheng J, Cai Y, Fu C-W. Mesh denoising using extended ROF model with ℓ_1 fidelity. Comput Graph Forum 2015;34(7):35–45.
[19] Lu X, Deng Z, Chen W. A robust scheme for feature-preserving mesh denoising. IEEE Trans Vis Comput Graphics 2016;22(3):1181–94.
[20] Liu Z, Lai R, Zhang H, Wu C. Triangulated surface denoising using high order regularization with dynamic weights. SIAM J Sci Comput 2019;41(1):1–26.
[21] Zhong S, Xie Z, Wang W, Liu Z, Liu L. Mesh denoising via total variation and weighted Laplacian regularizations. Comput Animat Virtual Worlds 2018;29(3–4):e1827.
[22] Baumgärtner L, Bergmann R, Herrmann M, Herzog R, Schmidt S, Vidal-Núñez J. Mesh denoising and inpainting using the total variation of the normal. 2020, arxiv preprint arXiv:2012.11748.
[23] Xu L, Lu C, Xu Y, Jia J. Image smoothing via ℓ_0 gradient minimization. ACM Trans Graph 2011;30(6):174:1–174:12.
[24] Wu C, Zhang J, Duan Y, Tai X-C. Augmented Lagrangian method for total variation based image restoration and segmentation over triangulated surfaces. J Sci Comput 2012;50(1):145–66.
[25] Liu Z, Zhong S, Xie Z, Wang W. A novel anisotropic second order regularization for mesh denoising. Comput Aided Geom Design 2019;71:190–201.
[26] Wei M, Huang J, Xie X, Liu L, Wang J, Qin J. Mesh denoising guided by patch normal co-filtering via kernel low-rank recovery. IEEE Trans Vis Comput Graphics 2019;25:2910–26.
[27] Chen H, Wei M, Sun Y, Xie X, Wang J. Multi-patch collaborative point cloud denoising via low-rank recovery with graph constraint. IEEE Trans Vis Comput Graphics 2019;26(11):3255–70.
[28] Chen H, Huang J, Remil O, Xie H, Qin J, Guo Y, Wei M, Wang J. Structure-guided shape-preserving mesh texture smoothing via joint low-rank matrix recovery. Comput Aided Des 2019;115:122–34.
[29] Arvanitis G, Lalos A, Moustakas K. Feature-aware and content-wise denoising of 3D static and dynamic meshes using deep autoencoders. In: 2019 IEEE international conference on multimedia and expo (ICME). 2019, p. 97–102.
[30] Wang J, Huang J, Wang FL, Wei M, Qin J. Data-driven geometry-recovering mesh denoising. Comput Aided Des 2019;114:133–42.
[31] Zhao W, Liu X, Zhao Y, Fan X, Zhao D. Normalnet: Learning based guided normal filtering for mesh denoising. 2019, arxiv preprint arXiv:1903.04015.
[32] Li X, Li R, Zhu L, Fu C-W, Heng P-A. DNF-Net: a deep normal filtering network for mesh denoising. IEEE Trans Vis Comput Graphics 2020;1.
[33] Arvanitis G, Lalos AS, Moustakas K. Image-based 3D MESH denoising through a block matching 3D convolutional neural network filtering approach. In: 2020 IEEE international conference on multimedia and expo (ICME). IEEE; 2020, p. 1–6.
[34] Nousias S, Arvanitis G, Lalos AS, Moustakas K. Fast mesh denoising with data driven normal filtering using deep variational autoencoders. IEEE Trans Ind Inf 2021;17(2):980–90.
[35] Shi J, Malik J. Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell 2000;22(8):888–905.

[36] Xu Y, Xie Z, Wu L, Chen Z. Multilane roads extracted from the openstreetmap urban road network using random forests. Trans GIS 2019;23(2):224–40.
[37] Tong W, Yang X, Pan M, Chen F. Spectral mesh segmentation via gradient minimization. IEEE Trans Vis Comput Graphics 2020;26(4):1807–20.
[38] Zhao W, Liu X, Wang S, Fan X, Zhao D. Graph-based feature-preserving mesh normal filtering. IEEE Trans Vis Comput Graphics 2019;PP(99):1.
[39] Arvanitis G, Lalos AS, Moustakas K, Fakotakis N. Feature preserving mesh denoising based on graph spectral processing. IEEE Trans Vis Comput Graphics 2019;25(3):1513–27.
[40] Sun Y, Chen H, Qin J, Li H, Wei M, Zong H. Reliable rolling-guided point normal filtering for surface texture removal. Comput Graph Forum 2019;38(7):721–32.
[41] Sun X, Rosin P, Martin R, Langbein F. Fast and effective feature-preserving mesh denoising. IEEE Trans Vis Comput Graphics 2007;13(5):925–38.
[42] Zhang J, Deng B, Hong Y, Peng Y, Qin W, Liu L. Static/dynamic filtering for mesh geometry. IEEE Trans Vis Comput Graphics 2018;25(4):1774–87.
[43] Von Luxburg U. A tutorial on spectral clustering. Stat Comput 2007;17(4):395–416.
[44] Park MK, Lee SJ, Jang IY, Lee YY, Lee KH. Feature-aware filtering for point-set surface denoising. Comput Graph 2013;37(6):589–95.