Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Efficient mesh denoising via robust normal filtering and alternate vertex updating*

Tao LI‡[1], Jun WANG[2], Hao LIU[2], Li-gang LIU[3]

([1]*College of Mathematics and Physics, Suzhou University of Science and Technology, Suzhou 215007, China*)

([2]*College of Mechanical and Electronics, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China*)

([3]*College of Mathematics Sciences, University of Science and Technology of China, Hefei 230026, China*)

E-mail: litao@mail.usts.edu.cn; wjun@nuaa.edu.cn; liuhao-01@nuaa.edu.cn; lgliu@ustc.edu.cn

Received May 4, 2016; Revision accepted Sept. 27, 2016; Crosschecked Nov. 8, 2017

Abstract: The most challenging problem in mesh denoising is to distinguish features from noise. Based on the robust guided normal estimation and alternate vertex updating strategy, we investigate a new feature-preserving mesh denoising method. To accurately capture local structures around features, we propose a corner-aware neighborhood (CAN) scheme. By combining both overall normal distribution of all faces in a CAN and individual normal influence of the interested face, we give a new consistency measuring method, which greatly improves the reliability of the estimated guided normals. As the noise level lowers, we take as guidance the previous filtered normals, which coincides with the emerging rolling guidance idea. In the vertex updating process, we classify vertices according to filtered normals at each iteration and reposition vertices of distinct types alternately with individual regularization constraints. Experiments on a variety of synthetic and real data indicate that our method adapts to various noise, both Gaussian and impulsive, no matter in the normal direction or in a random direction, with few triangles flipped.

Key words: Mesh denoising; Guided normal filtering; Alternate vertex updating; Corner-aware neighborhoods
https://doi.org/10.1631/FITEE.1601229 **CLC number:** TP391.7

1 Introduction

Triangular mesh is a significant surface modeling method due to its simplicity, topological adaptivity, and convenient availability. However, noise from different sources is inevitable, and this has great impact on the downstream applications, such as visualization, reconstruction, and shape analysis. Therefore, the past two decades have seen a lot of study on mesh denoising.

Many traditional denoising methods may miss sharp features on the input meshes. Therefore, state-of-the-art works on mesh denoising have focused on maintaining sharp features while eliminating noise on account of the ambiguity between them, which are both high-frequency parts from the signal processing point of view (Taubin, 1995). Among the numerous mesh denoising approaches, one of the most representative is filter-based anisotropic denoising. Vertex filtering methods (Fleishman *et al.*, 2003; Jones *et al.*, 2003) directly denoise mesh vertices, and are efficient but not effective. Normal filtering-based methods (Ohtake *et al.*, 2002; Yagou *et al.*, 2003; Zheng *et al.*, 2011; Wang *et al.*, 2012; Wei *et al.*, 2015; Zhang WY *et al.*, 2015) first filter noisy normals and then update vertices with refined normals, so they are more robust than vertex filtering methods. As the first-order differential geometry property is more sensitive to noise, how to filter face normals robustly is a crucial step for these methods. The patch-based

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 61402224 and 61222206), the Natural Science Foundation of Jiangsu Province, China (No. BK2014833), and the Natural Science Foundation of Suzhou University of Science and Technology, China (No. XKZ201611)

ORCID: Tao LI, http://orcid.org/0000-0002-7247-6126

guided normals estimation method (Zhang WY *et al.*, 2015) can greatly improve the reliability of filtered face normals. However, it still lacks robustness at corners for the adoption of conventional isotropic neighborhoods, though an anisotropic bilateral filtering operator is employed. In addition, the consistency measure is unreliable for it considers only face normals in a patch as a whole and omits the special influence of interested faces. To accurately capture local structures around features, we propose a novel scheme for finding neighborhoods, corner-aware neighborhoods (CANs), and estimate guided normals with them. Furthermore, we define a more reliable consistency measure for all CANs and choose one with the best consistency to estimate the guided normal.

As an important aspect of mesh quality, regularity has great impact on visualization and shape analysis. However, many mesh denoising methods overlook the regularity of denoised meshes, leading to the coexistence of superior smoothness and inferior triangulation. Though filtering-based methods move vertices along the normal direction in the vertex updating process, the influence of noise and poor mesh quality often cause distortion or even flipped triangulation. Similar to Ohtake *et al.* (2001), we introduce tangential Laplacian constraints to improve mesh regularity. Owing to the heavy dependence on vertex normals, tangential Laplacian constraints adapt to only vertices with reliable normals, i.e., non-feature vertices. Therefore, different from the indiscriminative abuse of normals of all vertices, we impose individual regularization constraints for distinct vertices. Then another tricky problem appears: how to identify feature vertices? Although many researchers (Shen *et al.*, 2005; Fan *et al.*, 2010; Bian and Tong, 2011; Wang J *et al.*, 2012; Wang RM *et al.*, 2014; Wei *et al.*, 2015) have attempted to address this issue, none of them could achieve reliable classification results as they conducted feature recognition before mesh denoising and they were not immune to the interference of noise. Since noise removal and feature recognition are themselves a coupled 'chicken and egg' hard problem, we take an absolutely different measure in this study. As the purpose of vertex assortment is to aid mesh regularization, or more specifically, to ensure the reliability of the vertex normal while calculating the tangential Laplacian operator, we simply take a vertex as a feature if the filtered normals of its adjacent faces are significantly different. Like most previous works, we further classify feature vertices into edge features and corners according to the number of neighboring features. A feature is called an edge feature if it has exactly two neighboring features; otherwise, it is called a corner. Since we classify vertices and update vertex positions iteratively at each normal filtering step, most misclassification can be rectified automatically.

The contributions of this paper are as follows:

1. A robust guided normal estimation method is proposed based on CANs. CANs are of better adaptivity to local structures around corners than the conventional neighborhoods.

2. Individual regularization terms are integrated into vertex updating models according to vertex classification, which makes our method suitable for multifarious noise but immune to flipping.

2 Related work

2.1 Isotropic smoothing

Early work on mesh smoothing or fairing decimated all high-frequency information blindly, be it feature or noise. The motivation often came from aesthetic or visual requirements. As one of the most representative methods, Laplacian smoothing is both simple and efficient. However, since the Laplacian operator pushes a vertex to the convex combination of its neighbors, mesh shrinkage is inevitable. Therefore, it has been a long effort to investigate methods to resist shrinkage while smoothing. Taubin (1995) proposed a signal processing approach for mesh denoising, where volume preserving constraint was used to prevent shrinkage. In the curvature flow based mesh fairing method, Desbrun *et al.* (1999) adopted a rescaling technique to prevent shrinkage. These methods are isotropic filtering, so features are indiscriminately blurred.

2.2 Anisotropic mesh denoising

There have been a lot of studies on anisotropic mesh denoising, which put much emphasis on keeping features while denoising. A thorough review is beyond the scope of this study, and we list only some representative and related work.

2.2.1 Filtering-based mesh denoising

Fleishman *et al.* (2003) and Jones *et al.* (2003) first brought bilateral filtering into the digital geometric processing realm independently. They all conducted filtering directly on vertex positions. Jones *et al.* (2003) used face normal as a first-order predictor of the local geometry while Fleishman *et al.* (2003) resorted to vertex normals. They performed well for meshes with low-level noise. Ohtake *et al.* (2002) and Yagou *et al.* (2003) used mean, median, and alpha-trimming filters for normal refining. Shen *et al.* (2005) took advantage of a fuzzy vector median filter to compute normals of non-feature vertices. Chen *et al.* (2005) employed a sharpness-dependent weighting function to determine a polygon face normal lying between the mean normal and the closest normal of its neighboring faces. The degree of sharpness is sensitive to noise and mesh uniformity. Sun *et al.* (2007) filtered a face normal by averaging all its neighboring faces' normals whose differences with the considered normal were within a threshold, which exaggerated original noisy normals and degraded smoothness of the denoised mesh. Furthermore, Sun *et al.* (2008) proposed a random walks model to filter the normal and solved the vertex optimization problem with a conjugate gradient approach.

In the local scheme, Zheng *et al.* (2011) carried out bilateral filtering on face normals first, and then updated vertices with filtered normals. Wei *et al.* (2015) first filtered face normals with methods of Zheng *et al.* (2011) and then combined both face normals and vertex normals to construct a vertex optimization model. Solomon *et al.* (2014) presented a theoretically sound generalization of bilateral and mean shift filter within a unified framework via heat diffusion, which can be used for feature-preserving mesh denoising. Recently, Wang *et al.* (2015) transplanted the rolling guidance idea from image processing to geometry processing, and they simply took as guidance the filtered normals of the last iteration. As the authors said, this method could not preserve features well while being used for mesh denoising. Inspired by Cho *et al.* (2014), Zhang WY *et al.* (2015) estimated the guided normal for each face according to consistency measures of all patches containing it, which greatly improved the accuracy of filtered normals.

2.2.2 Optimization-based mesh denoising

To keep the fidelity of noisy meshes to the original ones, additional geometrical constraints were often imposed on the vertex optimization model. Liu *et al.* (2007) pushed the centroid and feature position constraints onto a global Laplacian operator to prevent contraction and feature blurring. In the global scheme of Zheng *et al.* (2011), bilateral normal filtering was taken as a Laplacian operator and an objective function was formulated with a data term as a soft constraint. Based on the sparsity optimization idea prevalent in image processing recent years, He and Schaefer (2013) defined an edge differential operator and imposed a sparsity constraint by minimizing an ℓ_0 norm. Wang *et al.* (2014) first constructed a C^2 base mesh and then decomposed feature and noise progressively on residuals via ℓ_1-analysis compressed sensing. Zhang HY *et al.* (2015) looked on face normals as a piecewise constant function on mesh and defined its gradient on edges. An optimization model was set up by minimizing total variation (TV) of the first-order information and solved via variable splitting and an augmented Lagrange method. Though novel in methodology, all these methods need to solve a complex optimization model. Thus, their time complexity is much higher than that of iterative methods, which is a great obstacle for their applications.

2.2.3 Features vs. mesh denoising

As discussed in the last section, the ambiguity between features and noise is the largest challenge in mesh denoising. To preserve features well, researchers took various measures for different types of vertices. However, it is hard to classify vertices robustly with noise. Shen *et al.* (2005) partitioned feature and non-feature regions simply by an angle threshold, while Fan *et al.* (2010) categorized vertices using a density-based clustering algorithm via shared nearest neighbors. Bian and Tong (2011) classified vertices according to integral invariant values of mesh and individual updating strategies were adopted for distinct vertices. Wang *et al.* (2012) identified feature faces by a sharpness indicator and anisotropic neighborhoods of feature faces were constructed according to a weighted dual graph. Wei *et al.* (2015) categorized vertices with normal tensor voting method first, and then rectified the

categorization by adjacent face clustering. All these methods group vertices directly on the noisy meshes. To improve robustness of vertex classification, Lu *et al.* (2016) detected features on the preprocessed mesh of the noisy input, and Wang *et al.* (2014) recovered features from the residual vector between input data and the last estimation in a progressive manner.

2.2.4 Regularity vs. mesh denoising

Many mesh denoising methods consider regularity of the denoised meshes as of great importance. Ohtake *et al.* (2001) introduced a tangential Laplacian constraint for mesh smoothing. As the operator was added to all vertices updating equations indiscriminately, no features would be preserved. He and Schaefer (2013) integrated a triangle shape regularizer for each edge in the optimization model and improved the denoised mesh greatly. Wei *et al.* (2013) conducted constrained Laplacian smoothing independently after bilateral mesh filtering. Actually, denoising and regularizing can be unified into a single framework. In the variational minimization method of Wang *et al.* (2014), a discrete Laplacian regularization term was used to measure smoothness of the denoised mesh. Reliability of the vertex normal has a great impact on the robustness of the algorithm.

3 Joint bilateral normal filtering with robust guidance

3.1 Joint bilateral normal filtering

Joint bilateral normal filtering has been widely applied in digital geometry processing recently:

$$\boldsymbol{n}_i^{k+1} = \frac{1}{W_i} \sum_{f_j \in N(f_i)} w_j K_{\mathrm{s}}(\|\boldsymbol{c}_i^k - \boldsymbol{c}_j^k\|) \cdot K_{\mathrm{r}}(\|J(\boldsymbol{n}_i^k) - J(\boldsymbol{n}_j^k)\|)\boldsymbol{n}_j^k, \quad (1)$$

where $\boldsymbol{c}_i^k$ and $\boldsymbol{n}_i^k$ are the centroid and normal of face f_i in the kth iteration, $K_{\mathrm{s}}(x) = \exp(-x^2/(2\sigma_{\mathrm{s}}^2))$, $K_{\mathrm{r}}(x) = \exp(-x^2/(2\sigma_{\mathrm{r}}^2))$, $W_i = \sum_{f_j \in N(f_i)} w_j K_{\mathrm{s}}(\|\boldsymbol{c}_i^k - \boldsymbol{c}_j^k\|) K_{\mathrm{r}}(\|J(\boldsymbol{n}_i^k) - J(\boldsymbol{n}_j^k)\|)$ is a normalization factor, and w_j is an additional weight which will be explained later. If $J(\cdot)$ is a constant operator, it degenerates into the general isotropic Gaussian filtering; if $J = I$ is an identity operator, it is the bilateral normal filtering (Zheng *et al.*, 2011). If $J(\boldsymbol{n}_i)$ denotes the last filtered result of $\boldsymbol{n}_i^k$, it becomes the rolling guidance filter (Wang *et al.*, 2015); if $J(\boldsymbol{n}_i) = \boldsymbol{g}_i$ is a guidance of $\boldsymbol{n}_i$, it is the guided normal filtering (Zhang WY *et al.*, 2015). Thus, various filtering operators can be deduced when different guidance signals $J(\boldsymbol{n}_i)$ are chosen.

Thanks to the great importance of guidance on joint bilateral filtering, Zhang WY *et al.* (2015) proposed a patch-based guided normals estimation method, which produces better denoising results than bilateral normal filtering (Zheng *et al.*, 2011). However, on account of the usage of isotropic neighborhoods, the guided normals may be erroneous for some faces around corners, as illustrated in Fig. 1a, where the dark face is incorrectly grouped into the left patch. This is possible as the consistency measures of the marked patches in Figs. 1a and 1b are equal according to Eq. (7) of Zhang WY *et al.* (2015). Even if we were lucky enough and we caught the right representative patch (Fig. 1b), we could not obtain appropriate guidance for the inclusion of at least two faces of different structures. Thus, the method of Zhang WY *et al.* (2015) cannot always preserve corners well, and this will be illustrated in Section 5.

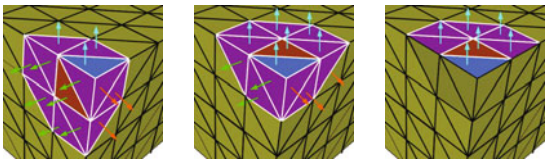

Fig. 1 Guided normal estimation for a face near a corner. The interested face (dark) contains a corner and two edge feature vertices. A wrong patch may be chosen as guidance with the method of Zhang WY *et al.* (2015) (a), even the right patch (b) cannot provide the accurate guided normal. With the CAN, we can catch the right structure (c)

3.2 Corner-aware neighborhood

Before elaborating on how to construct reliable guided normals, we present a novel neighborhood to adapt to the local structure of triangular meshes. First, we review the general definition of neighborhoods for triangular meshes. Let $\mathcal{M} = \{\mathcal{V}, \mathcal{F}\}$ be a triangular mesh with vertex set $\mathcal{V}$ and face set $\mathcal{F}$, and the neighborhood of a face $f \in \mathcal{F}$, denoted by $N(f)$, is the set of faces which have at least one common vertex or edge with f. Unless otherwise specified, all neighborhoods in this study are the first type, i.e., vertex-based neighborhoods. Likewise, the

neighborhood $N(\boldsymbol{v})$ of vertex $\boldsymbol{v} \in \mathcal{V}$ is the set of all faces sharing vertex $\boldsymbol{v}$. For simplicity, we also use these symbols to denote sets of vertices of these faces, which can be easily distinguished according to the context. These neighborhoods are topologically related. There are also geometrical neighborhoods (Zhang WY *et al.*, 2015), which are confined by the distances between the centroid of the central face and those of its surrounding faces. All these neighborhoods are symmetric about the central face, forming a disk-shaped, isotropic structure.

To capture local structures around features, we propose a new, anisotropic neighborhood construction scheme. Each face is assigned several neighborhoods, which are not necessarily symmetric about the face of interest. First, we introduce a vertex-based construction method. For each reference face, there are in total six neighborhoods attached. Three are vertex-based neighborhoods centered at its three vertices, and three are unions of the reference face with another vertex-based neighborhoods, which are centered at the opposite vertices of three edge neighboring faces of the reference face. In Fig. 2, the yellow faces are the reference faces and the faces within the red polygons form the neighborhoods. With such neighborhoods, we can easily catch local structures around corners in the example (Fig. 1c). Owing to the adaptivity to structures around corners, we call such neighborhoods CANs, the first three as Type I CANs and the second three as Type II CANs.

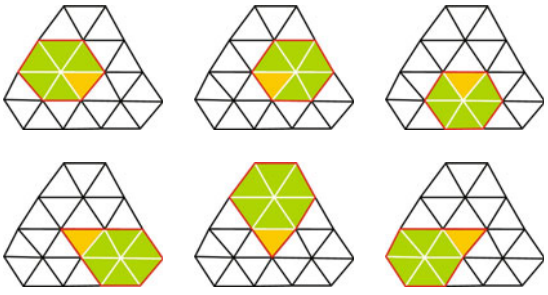

Fig. 2 Vertex-based CANs (1st row: type I CANs; 2nd row: type II CANs). References to color refer to the online version of this figure

Despite the perfect adaptivity to local substructures, the vertex-based CANs are sensitive to large noise as there are only six or seven faces in each CAN. To improve reliability, we further introduce face-based CANs. For each face f, we call all the (face-based) neighborhoods containing f as Type I CANs, and choose some patches as Type II CANs. In Fig. 3, each Type II CAN has an edge neighboring face f' of f (yellow faces), and f' is a vertex-based neighboring face of the patch center (dark green faces) and is at least three faces away from the patch center in the rotational sense. There are about three face-based Type II CANs for each face.

Fig. 3 Face-based Type II CANs. References to color refer to the online version of this figure

To capture local small structures around features more accurately, we further define edge-based CANs. In Fig. 4, edge-based CANs are constructed by discarding one of the neighboring faces of the interested face's (edge-based) one-ring neighborhood. Though less robust than the other two kinds, edge-based CANs are especially fit for the situation when one triangular face strides over two feature lines, which often occurs in meshes with slim structures. We do not suggest using the edge-based CANs independently, but take them as a supplement to the other two methods if the mesh has some slim structures. Fig. 12 shows a successful application of edge-based CANs.

Fig. 4 Edge-based CANs

3.3 CANs guided normal estimation

3.3.1 Consistency measure

On account of the misguidance of the consistency measure given in Zhang WY *et al.* (2015), we propose a new definition for it. Consider the set of related CANs of face f_i: $\mathcal{S}(f_i) = \{\mathcal{P}_k,\ \mathcal{P}_k$ is a CAN of $f_i,\ k = 1, 2, \ldots, n_i\}$. First, we define the inner

consistency measure of $\mathcal{P}_k$ as

$$\mathcal{I}(\mathcal{P}_k) = \sqrt{\frac{1}{(|\mathcal{P}_k|-1)\bar{A}_k}\sum_j A_j \|\boldsymbol{n}_j - \bar{\boldsymbol{n}}_k\|^2} \cdot \max_{f_l, f_j \in \mathcal{P}_k} \|\boldsymbol{n}_l - \boldsymbol{n}_j\|, \quad (2)$$

where $|\mathcal{P}_k|$ is the number of faces in $\mathcal{P}_k$, A_j and $\boldsymbol{n}_j$ are area and normal of face f_j, $\bar{A}_k = \frac{1}{|\mathcal{P}_k|}\sum_j A_j$, $\bar{\boldsymbol{n}}_k = \boldsymbol{a}_k/\|\boldsymbol{a}_k\|$, and $\boldsymbol{a}_k = \sum_{f_j \in \mathcal{P}_k} \boldsymbol{n}_j A_j$. $\mathcal{I}(\mathcal{P}_k)$ describes the distribution of face normals of $\mathcal{P}_k$ as a whole, which can be seen as an intrinsic measure of consistency. Furthermore, we construct a measure characterizing the outer influence of f_i on its CAN $\mathcal{P}_k$ by

$$\mathcal{O}(\mathcal{P}_k, f_i) = \phi(\|\boldsymbol{n}_i - \bar{\boldsymbol{n}}_k\|). \quad (3)$$

To not exaggerate the influence of the reference face normal, we set $\phi(\cdot)$ to be a monotonically increasing, concave function in interval $[0, 2]$; $\phi(x) = \sqrt{x}$ is a good choice in all our experiments.

The total consistency measure of $\mathcal{P}_k$ corresponding to face f_i is defined as

$$M_{ki} = \mathcal{I}(\mathcal{P}_k) \cdot \mathcal{O}(\mathcal{P}_k, f_i). \quad (4)$$

Note the inverse relationship between the value of M_{ki} and the consistency. Large M_{ki} manifests bad consistency, which means $\mathcal{P}_k$ may contain features or large noise, while small M_{ki} indicates good consistency and faces in $\mathcal{P}_k$ form a relatively flat region.

3.3.2 Guided normal estimation

For each face, we calculate all the consistency measures M_{ki} of its CANs and choose as guidance the mean normal of the CAN with least measure. If the reference face is far away from features and with little noise, which means all the consistency measures are small enough, there is no further need to calculate and compare all of them. As the mesh gets smoother and smoother during the filtering process, the smoothness remains in the following iterations. Therefore, we set a flag S_i for each face to label the smoothness. For a smooth face, we simply calculate its consistency measure and mean normal with conventional neighborhoods.

To measure the smoothness of a face, we introduce a threshold τ. We say a face is smooth if all of its CANs satisfy $M_{ki} < \tau$. However, it is not easy to choose a proper τ for meshes with different levels of noise. To adapt to multifarious data, we set τ with a statistical method. Sort the consistency measures of all faces in ascending order, and take τ to be the $[\alpha \cdot n_i]$th value in the sequence. Here, n_i is the size of the sequence and α an empirical parameter. We can obtain perfect results when $\alpha = 0.6$.

Taking the representative Fandisk model as an example, Fig. 5 compares the differences between the guided normals and ground-truth ones before the first normal iteration. Zheng *et al.* (2011) used the noisy normals directly, and the normal differ-

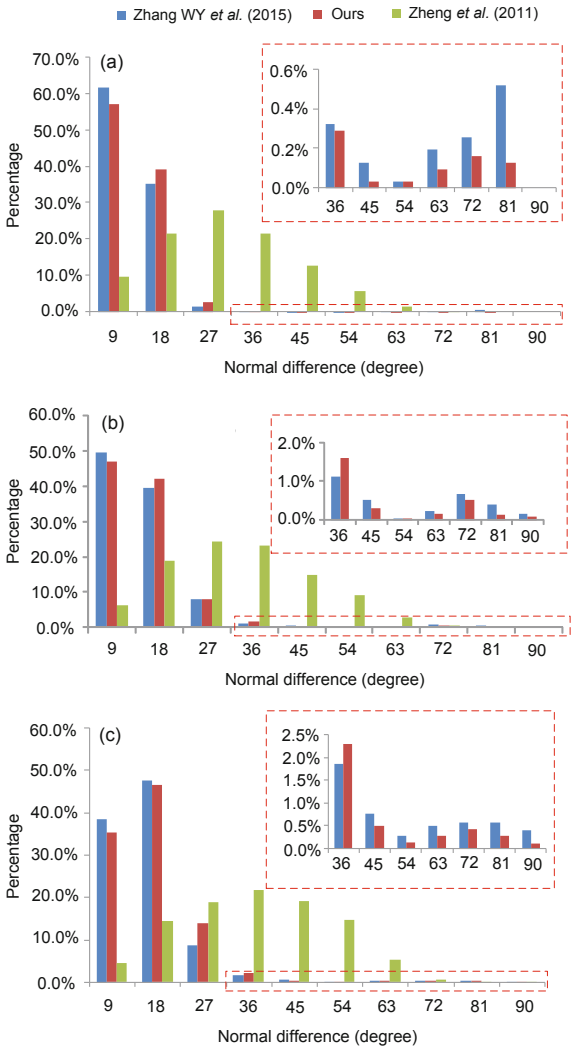

Fig. 5 Comparison of differences between guided normals and ground-truth ones before the first normal iteration. The histograms of normal differences of the Fandisk model are with levels 0.25 (a), 0.30 (b), and 0.35 (c) of Gaussian noise distributed in the normal direction. The subplots are the close-up view of the dashed differences of Zhang WY *et al.* (2015) and our method

ences of their method are indeed the original normal noise, which provides a reference for the other methods. As can be seen from Fig. 5b, our method produces less misleading guided normals (with difference more than 45 degrees) than that of Zhang WY *et al.* (2015). Note that the errors listed here are about the mesh never denoised, which are caused mainly by large noise near features. Inaccurate guided normals tend to push corresponding faces to the false structures in the vertex updating process. Fortunately, most such inaccuracy will be remedied automatically as iteration proceeds.

The CANs based guided normal estimation process is illustrated in Algorithm 1.

Algorithm 1 Robust estimation of guided normals

Input: Mesh $\mathcal{M} = \{\mathcal{V}, \mathcal{F}\}$ with smoothness flag S_i for each face f_i.

Output: Guided normal $\boldsymbol{g}_i$, consistency measure M_i, and updated flag S_i for each face in $\mathcal{F}$.

```
for each face f_i ∈ F do
    if S_i == false then
        Calculate consistency measure M_ki for all CANs according to Eq. (4);
        Set g_i and M_i for f_i;
    else
        Calculate g_i and M_i directly with the conventional neighborhood N(f_i);
    end if
end for
Estimate tolerance τ with M_i of all faces;
Update flags S_i for all faces with τ.
```

Remark 1 The differences are twofold on guided normal estimation between our method and Zhang WY *et al.* (2015). First, we take distinguishing anisotropic neighborhoods, CANs, which have better adaptivity to local substructures near features. Second, the computational methods on the consistency measure are distinct. In Zhang WY *et al.* (2015), only normal differences in a patch were considered. In our scenario, however, both normal distribution of the patch as a whole and the normal consistency between the interested face and the patch are involved. So our method is more robust.

Remark 2 As for the additional weights w_j in Eq. (1), Zheng *et al.* (2011) and Zhang WY *et al.* (2015) simply took it to be the area of f_j, which characterized face sampling rate from a geometric perspective. Here, we further enrich it from an algebraic aspect. Since the consistency measure assesses the reliability of the face normal, the smaller, the better. We endow larger weights for faces with smaller consistency measures, and smaller weights for faces with larger consistency measures. So, w_j is defined as

$$w_j = A_j(M_{\mathrm{U}} - M_j), \tag{5}$$

where M_{U} is an upper bound of all consistency measures M_j. We can obtain decent results while taking $M_{\mathrm{U}} = \max_j M_j + \epsilon$ with $\epsilon = 0.01$.

Remark 3 The new neighborhoods CANs are applied to only estimate guided normals for their adaptivity to local structures near features. As to normal filtering and vertex updating, we still take advantage of the conventional neighborhoods.

3.3.3 Acceleration via rolling guidance

The introduction of flags S_i makes the calculation of guided normals more efficient. However, it is still a relatively costly process. Fortunately, as noise decimates in the first few iterations, we do not need to estimate guided normals via CANs throughout the whole normal filtering process. A natural selection is to leverage the filtered normals of the last iteration as guidance, which happens to coincide with the rolling guidance idea (Wang *et al.*, 2015). Since the purpose of normal filtering was to remove geometric features of different scales and the rolling guidance went with trivial zero initialization, their method did not preserve sharp features well when applied to mesh denoising. In our scenario, however, we estimate guided normals with CANs for the first few iterations, and then resort to the rolling guidance method for the remaining iterations when all large noise has been decimated. We obtain ideal results by taking a half-and-half partition of iterations for two manners: robust guidance filtering and rolling guidance filtering.

4 Alternate vertex updating

4.1 Optimization model

In many normal filtering-based mesh denoising methods (Sun *et al.*, 2007; Zheng *et al.*, 2011; Zhang WY *et al.*, 2015), vertices were updated by the

following iterative scheme:

$$\boldsymbol{v}_i^{k+1}=\boldsymbol{v}_i^k+\frac{1}{|N(\boldsymbol{v}_i)|}\sum_{f_j\in N(\boldsymbol{v}_i)}[\bar{\boldsymbol{n}}_{f_j}^k\cdot(\boldsymbol{c}_j^k-\boldsymbol{v}_i^k)]\bar{\boldsymbol{n}}_{f_j}^k, \quad (6)$$

where $\boldsymbol{c}_j^k$ and $\bar{\boldsymbol{n}}_{f_j}^k$ are the centroid and filtered normal of face f_j in the kth iteration, respectively. This formula was derived by a gradient descent method with step size $\frac{1}{|N(\boldsymbol{v}_i)|}$ while minimizing the error function (Taubin, 2001):

$$E(\boldsymbol{v})=\frac{1}{2}\sum_{f_k\in\mathcal{F}}\sum_{(i,j)\in\partial f_k}[\boldsymbol{n}_{f_k}\cdot(\boldsymbol{v}_i-\boldsymbol{v}_j)]^2. \quad (7)$$

Here, $(i,j)\in\partial f_k$ means that $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ form an edge of face f_k. Since this formula is concerned with filtered face normals but not vertex normals, it fits both feature and non-feature vertices. However, for meshes with large noise, vertices may not always move along their normal directions as expected in theory. Therefore, it is possible for the appearance of poor triangulation in the denoised meshes, some even being overlapped or flipped.

To improve the regularity of the denoised mesh, we integrate a regularization term $R(\boldsymbol{v})$ to the error function $E(\boldsymbol{v})$ and formulate an optimization model:

$$\boldsymbol{v}^*=\arg\min_{\boldsymbol{v}}\ E(\boldsymbol{v})+\beta R(\boldsymbol{v}), \quad (8)$$

where $\beta>0$ is a factor which balances the influence of the regularization term and the error term. The construction of the regularization term $R(\boldsymbol{v})$ will be elaborated in the next subsection.

4.2 Individual regularization constraints

To preserve regularity of the denoised mesh, we prefer to leverage the tangential Laplacian operator for its good performance in homogenizing mesh density (Ohtake *et al.*, 2001):

$$\mathcal{T}(\boldsymbol{v}_i)=\mathcal{L}(\boldsymbol{v}_i)-(\mathcal{L}(\boldsymbol{v}_i)\cdot\boldsymbol{n}_{\boldsymbol{v}_i})\boldsymbol{n}_{\boldsymbol{v}_i}, \quad (9)$$

i.e., the tangential component of Laplacian operator:

$$\mathcal{L}(\boldsymbol{v}_i)=\sum_{\boldsymbol{v}_{ij}\in N(\boldsymbol{v}_i)}w_{ij}\boldsymbol{v}_{ij}-\boldsymbol{v}_i, \quad (10)$$

where $\boldsymbol{n}_{\boldsymbol{v}_i}$ is the normal of vertex $\boldsymbol{v}_i$ and w_{ij} the Laplacian weights. There are many different Laplacian weights, e.g., constant weights, edge length related weights, and cotangent weights (Taubin, 1995; 2001; Desbrun *et al.*, 1999). Since the functionality of the tangential Laplacian operator is to improve mesh regularity, we choose the umbrella operator (Taubin, 1995), i.e., $w_{ij}=1/|N(\boldsymbol{v}_i)|$. For detailed comparison of different Laplacian operators, please refer to Desbrun *et al.* (1999).

The high dependence on vertex normals makes the tangential Laplacian operator suitable only for vertices with definite and reliable normals, i.e., non-feature vertices. We define the regularization term for non-feature vertices by

$$\begin{aligned}R(\boldsymbol{v})&=\frac{1}{2}\sum_{\boldsymbol{v}_i\in\boldsymbol{v}}\|\mathcal{T}(\boldsymbol{v}_i)\|^2\\&=\frac{1}{2}\sum_{\boldsymbol{v}_i\in\boldsymbol{v}}\|\mathcal{L}(\boldsymbol{v}_i)\|^2-\frac{1}{2}\sum_{\boldsymbol{v}_i\in\boldsymbol{v}}[\mathcal{L}(\boldsymbol{v}_i)\cdot\boldsymbol{n}_{\boldsymbol{v}_i}]^2.\end{aligned} \quad (11)$$

Clearly, the quadratic objective function (Eq. (8)) induces a sparse linear system. Though there are a lot of efficient libraries, such as Intel Math Kernel Library (Intel MKL), Taucs, and Eigen, to solve such a system, it is still a little expensive in temporal and spatial efficiency as the number of vertices increases. A simple and efficient way is to iterate with the gradient decent method (Sun *et al.*, 2007), i.e.,

$$\begin{aligned}\boldsymbol{v}_i^{k+1}=\boldsymbol{v}_i^k&+\frac{1}{|N(\boldsymbol{v}_i)|}\sum_{f_j\in N(\boldsymbol{v}_i)}[\bar{\boldsymbol{n}}_{f_j}^k\cdot(\boldsymbol{c}_j^k-\boldsymbol{v}_i^k)]\bar{\boldsymbol{n}}_{f_j}^k\\&+\beta\mathcal{T}(\boldsymbol{v}_i^k).\end{aligned} \quad (12)$$

For feature vertices, owing to the G^1 discontinuity, there is no rigorous definition of normal therein, so we resort to other regularization methods. For edge features, we encourage them to move along the feature lines they attach to. Similar to Eq. (11), we construct a regularization term by

$$R(\boldsymbol{v})=\frac{1}{2}\left\|\left(\frac{\boldsymbol{v}_{i1}^k+\boldsymbol{v}_{i2}^k}{2}-\boldsymbol{v}_i\right)\cdot\boldsymbol{n}_{12}\right\|^2, \quad (13)$$

where $\boldsymbol{v}_{i1}^k$ and $\boldsymbol{v}_{i2}^k$ are two neighboring features of edge feature $\boldsymbol{v}_i^k$ and $\boldsymbol{n}_{12}=\dfrac{\boldsymbol{v}_{i1}^k-\boldsymbol{v}_{i2}^k}{\|\boldsymbol{v}_{i1}^k-\boldsymbol{v}_{i2}^k\|}$. The updating formula for edge features is then formulated as

$$\begin{aligned}\boldsymbol{v}_i^{k+1}=\boldsymbol{v}_i^k&+\frac{1}{|N(\boldsymbol{v}_i)|}\sum_{f_j\in N(\boldsymbol{v}_i)}[\bar{\boldsymbol{n}}_{f_j}^k\cdot(\boldsymbol{c}_j^k-\boldsymbol{v}_i^k)]\bar{\boldsymbol{n}}_{f_j}^k\\&+\beta\left[\left(\frac{\boldsymbol{v}_{i1}^k+\boldsymbol{v}_{i2}^k}{2}-\boldsymbol{v}_i\right)\cdot\boldsymbol{n}_{12}\right]\boldsymbol{n}_{12}.\end{aligned} \quad (14)$$

For corners, as there are no appropriate tangent lines or tangent planes to guide, we simply update them with Eq. (6).

4.3 Detection of features

As the purpose of vertex classification is merely to assess the reliability of vertex normals, we do it in a simple and efficient way. At each iteration, we compute the differences of all filtered face normals of a vertex's neighboring faces. If the differences are all smaller than tolerance τ_a, we take the vertex as a non-feature vertex. Otherwise, it is looked on as a feature vertex. Now that classification is carried out in each iteration, it does not matter if there is some misclassification in the first few iterations. Then different types of features (edge features and corners) can be obtained according to the number of neighboring features as discussed in Section 1.

4.4 Alternate vertex updating

As mentioned in Section 2, most existing denoising methods work well for meshes without features. Therefore, instead of updating all vertices in a batch manner as in previous methods (Sun *et al.*, 2007; 2008; Zheng *et al.*, 2011; Wei *et al.*, 2015; Zhang WY *et al.*, 2015), we propose an alternate vertex updating strategy. Specifically, we perform three rounds of cycles for three different types of vertices: first for non-feature vertices with Eq. (12), then for edge features with Eq. (14), and last for corners with Eq. (6). Since the iteration proceeds from more credible non-feature vertices to less credible feature vertices, we expect a greater convergence speed than the conventional batch manner. Although there is no rigorous theoretical proof, a number of experiments verify this observation. Fig. 6 demonstrates the outperformance of the alternate method over the batch method by comparing mean normal error δ and mean distance error D_{mean} (to be explained in Section 5.2) on the cube model of Fig. 11.

Remark 4 The tangential Laplacian introduced in Ohtake *et al.* (2001) indiscriminately handles all vertices, which makes it inaccurate at features for the uncertainty of normals there. In our method, instead, we classify vertices at each iteration and impose original tangential Laplacian constraints on only non-feature vertices.

Remark 5 In the bi-normal filtering (Wei *et al.*, 2015), both face normal and vertex normal were involved. For feature vertices, they took as normals the mean normals of the corresponding substructures, the reliability of which is questionable. In our vertex updating process, we use only normals of non-feature vertices and avoid abusing normals of features for their G^1 discontinuity.

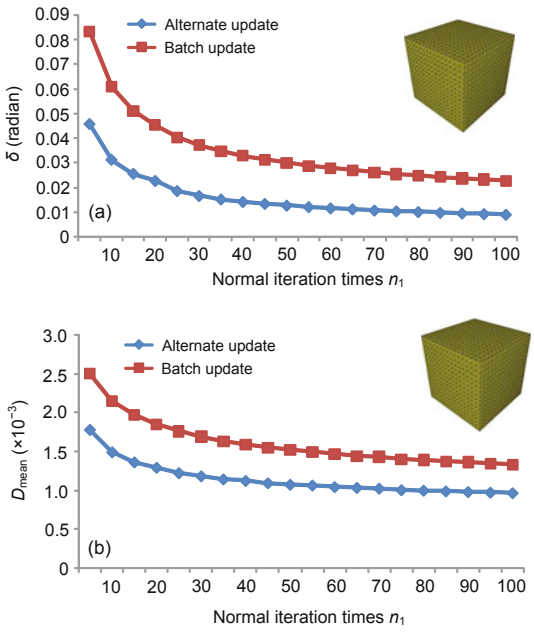

Fig. 6 Convergence comparison of the alternate method and the batch method for vertex updating: (a) mean normal error; (b) mean distance error

5 Experiments and discussion

To verify the effect of our proposed method, we have implemented it on a laptop with a 2.5-GHz Intel Core i5 CPU and 6-GB RAM with the C++ programming language and experimented on various data, both synthetic and real scanned. To clearly demonstrate the denoising effect, all models were rendered with flat shading mode. For synthetic data, two types of noise, Gaussian and impulsive noise were added to the ground-truth models, which may be along the normal direction or a random direction. The noise level means the ratio of the standard deviation of noise to the mean edge length of the corresponding meshes.

5.1 Parameter setting

There are eight parameters to be modulated in our algorithm, which are the two standard

deviations σ_s and σ_r in the bilateral filtering (Eq. (1)), the normal iteration time n_1, the vertex iteration time n_2, the cut-off parameter α for threshold τ, the normal difference tolerance τ_a while recognizing feature vertices, the balancing factor β in vertex updating (Eqs. (8) and (12)), and CANs type t. Fortunately, many parameters have specific geometric meanings and can be determined empirically. We can obtain ideal results when setting $\alpha = 0.8$, $n_2 = 10$, and $\sigma_s = r$, where r is the mean distance between the centroids of all edge neighboring faces. As these parameters are less sensitive to the final denoising results, unless specified exceptionally, we fix them in all our experiments. The normal difference tolerance τ_a affects the classification of vertices. The larger the tolerance, the more non-feature vertices are obtained. Parameter β balances the displacements of vertices in the normal direction and the tangential direction, which corresponds to the noise removal and mesh regularization process respectively. For noisy meshes with bad triangulation, we increase β; otherwise, we decrease it. Parameter σ_r controls the influence of normal deviation in the bilateral normal filtering process. The smaller, the more features are preserved. Similarly, we adjust the value of σ_r according to the scale of features to be preserved. The CANs type is set according to the noise level. For very noisy meshes, we prefer face-based CANs (F), while for meshes with low-level noise, we resort to vertex-based CANs (V). Edge-based CANs (E) are generally taken as a supplement to the first two methods when there are slim structures in the meshes. A number of experiments indicate that we can obtain ideal results when setting $\sigma_r \in [0.3, 0.4]$, $n_1 \in [1, 40]$, $\tau_a \in [0.9, 1.4]$, and $\beta \in [0.001, 0.02]$.

5.2 Results and comparison

We contrast our method the with five most related mesh denoising methods, which are ℓ_0 minimization method (L0M) (He and Schaefer, 2013), fast and efficient mesh denoising method (FED) (Sun *et al.*, 2007), local scheme of bilateral normal filtering (LBNF) (Zheng *et al.*, 2011), guided normal filtering (GNF) (Zhang WY *et al.*, 2015), and bi-normal filtering (BF) (Wei *et al.*, 2015). To make our comparison fair, all these algorithms were implemented with the same software and hardware environment as ours, and we tried our best to tune the parameters of each methods to obtain the best results.

To make our comparison tractable, we took the recommended values of the parameters that have little influence on the results, and considered only the parameters sensitive to the denoising results, which are: balancing parameters λ and α together with the updating ratio μ_α of L0M; n_1 and n_2 of FED; σ_s, n_1, and n_2 of LBNF; σ_r, n_1, n_2, and multiple m for the average distance between neighboring faces of GNF; σ_{s1}, σ_{s2}, n_1, n_2, α, and β of BF; σ_r, n_1, τ_a, β, and t of our method. Here, n_1, n_2, σ_{s1}, and σ_{s2} have the same meanings as in our method.

Since a regularization term is integrated into the optimization model of non-corner vertices, we expected a robust denoising effect for all kinds of noise. Fig. 7 demonstrates an example with Gaussian noise distributed in the normal direction. From the magnified image we see the better performance of our method over the others. Fig. 8 lists

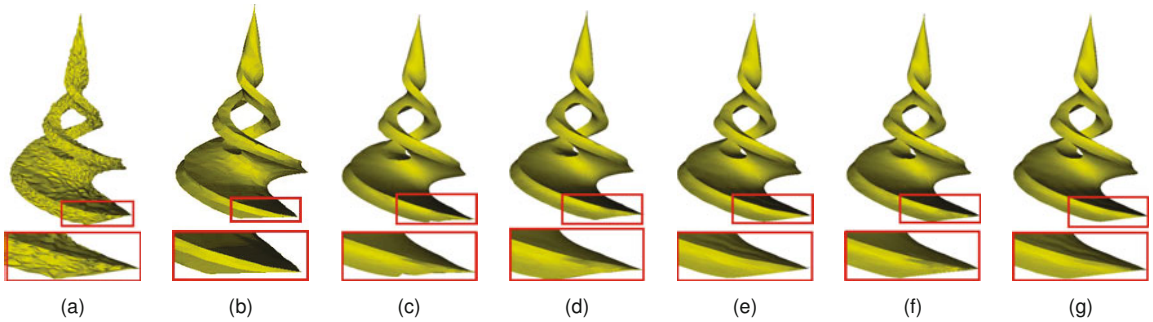

Fig. 7 Denoising a Twirl model with Gaussian noise distributed in the normal direction (noise level 0.2): (a) noisy mesh; (b) denoised results of L0M ($\lambda = 0.01$, $\alpha = 0.003$, $\mu_\alpha = 0.9$); (c) denoised results of FED ($n_1 = 10$, $n_2 = 50$); (d) denoised results of LBNF ($\sigma_s = 0.38$, $n_1 = 16$, $n_2 = 10$); (e) denoised results of GNF ($m = 2.0$, $\sigma_r = 0.35$, $n_1 = 10$, $n_2 = 50$); (f) denoised results of BF ($\sigma_{s1} = 0.4$, $\sigma_{s2} = 0.4$, $n_1 = 8$, $n_2 = 8$, $\alpha = 0.4$, $\beta = 0.2$); (g) denoised results of our method ($\sigma_r = 0.35$, $n_1 = 9$, $\tau_a = 1.2$, $\beta = 0.003$, $t = V$)

denoising results of the six methods on meshes with Gaussian noise added in a random direction, which is a more challenging issue for most methods but does occur in some really scanned data, as can be seen later. Though all the methods remove noise well, only our method avoids flip triangles well. Even for models with large noise, for example, the Fandisk model of Fig. 9 with noise level 0.6, our method can still achieve better performance than other methods. Figs. 10 and 11 show two models with impulsive

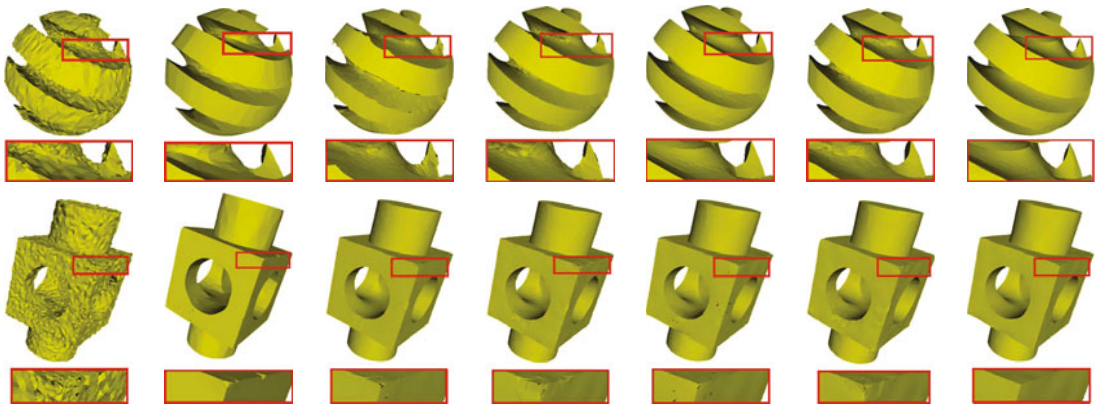

Fig. 8 Denoising meshes with Gaussian noise distributed in a random direction. The noise levels of meshes in (a) are all 0.3, (b) to (g) are denoising results of, for SharpSphere: L0M ($\lambda = 0.03$, $\alpha = 0.003$, $\mu_\alpha = 1.0$), FED ($n_1 = 10$, $n_2 = 20$), LBNF ($\sigma_s = 0.35$, $n_1 = 20$, $n_2 = 10$), GNF ($m = 2.8$, $\sigma_r = 0.25$, $n_1 = 10$, $n_2 = 10$), BF ($n_1 = 10$, $n_2 = 8$, $\sigma_{s1} = 0.4$, $\sigma_{s2} = 0.4$, $\alpha = 0.3$, $\beta = 0.3$), and our method ($\sigma_r = 0.35$, $n_1 = 12$, $\tau_a = 1.25$, $\beta = 0.006$, $t = V$); for Block: L0M ($\lambda = 0.05$, $\alpha = 0.004$, $\mu_\alpha = 1.0$), FED ($n_1 = 20$, $n_2 = 50$), LBNF ($\sigma_s = 0.35$, $n_1 = 15$, $n_2 = 10$), GNF ($m = 2.4$, $\sigma_r = 0.3$, $n_1 = 15$, $n_2 = 10$), BF ($n_1 = 10$, $n_2 = 8$, $\sigma_{s1} = 0.3$, $\sigma_{s2} = 0.3$, $\alpha = 0.3$, $\beta = 0.3$), and our method ($\sigma_r = 0.35$, $n_1 = 12$, $\tau_a = 1.35$, $\beta = 0.016$, $t = V$)

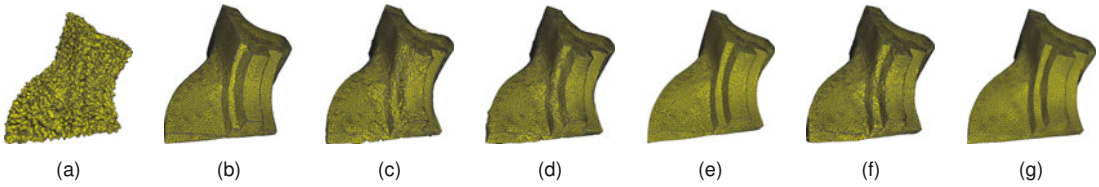

Fig. 9 Denoising meshes with large Gaussian noise distributed in a random direction: (a) noise level 0.6; (b) denoising results of L0M ($\lambda = 0.01$, $\alpha = 0.003$, $\mu_\alpha = 1$); (c) denoising results of FED ($n_1 = 30$, $n_2 = 40$); (d) denoising results of LBNF ($\sigma_s = 0.35$, $n_1 = 30$, $n_2 = 20$); (e) denoising results of GNF ($m = 2.4$, $\sigma_r = 0.33$, $m_s = 1.3$, $n_1 = 20$, $n_2 = 10$); (f) denoising results of BF ($n_1 = 30$, $n_2 = 8$, $\sigma_{s1} = 0.4$, $\sigma_{s2} = 0.35$, $\alpha = 0.3$, $\beta = 0.3$); (g) denoising results of our method ($\sigma_r = 0.38$, $n_1 = 20$, $\tau_a = 1.38$, $\beta = 0.006$, $t = F$)

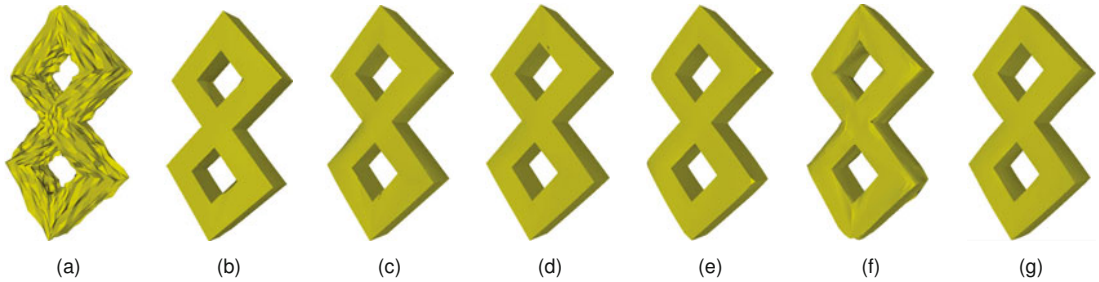

Fig. 10 Denoising a DoubleTorus model with impulsive noise distributed in the normal direction: (a) noise level 0.3 for 50% input data; (b) denoising results of L0M ($\lambda = 0.05$, $\alpha = 0.006$, $\mu_\alpha = 0.9$); (c) denoising results of FED ($n_1 = 80$, $n_2 = 15$); (d) denoising results of LBNF ($\sigma_s = 0.36$, $n_1 = 60$, $n_2 = 15$); (e) denoising results of GNF ($m = 2.6$, $\sigma_r = 0.3$, $n_1 = 25$, $n_2 = 8$); (f) denoising results of BF ($\sigma_{s1} = 0.4$, $\sigma_{s2} = 0.4$, $n_1 = 40$, $n_2 = 10$, $\alpha = 0.3$, $\beta = 0.3$); (g) denoising results of our method ($\sigma_r = 0.36$, $n_1 = 35$, $\tau_a = 1.35$, $\beta = 0.001$, $t = F$)

noise added in the normal and random directions, respectively. It is obvious to see the superiority of our method over the other five. In addition, we can avoid the corner-cutting problem of Zhang WY *et al.* (2015) in both Figs. 10e and 11e caused by inaccurate guided normals.

To compare quantitatively, we introduce three quantities from Belyaev and Ohtake (2003), which are the area weighted mean distance (D_{mean}), the maximal distance ($D_{\max}$) of all denoised vertices to the ground-truth meshes, and the area weighted mean normal deviation (δ) of all faces between the denoised meshes and the ground-truth ones. These metrics were also adopted in existing works, such as Sun *et al.* (2007), Bian and Tong (2011), and Zhang WY *et al.* (2015). Table 1 lists the comparison of all the running results of the six methods on the synthetic examples. The best results of each item are highlighted. We can see that our method outperforms the others for most cases, which coincides with our visual observation.

The total time consumption of these examples is also listed in Table 1, from which we can see that the methods of FED and LBNF are very efficient, which take less than 1 s for a moderate model. However, their denoising effect is limited, especially for meshes with high-level noise. BF is less efficient and it costs over 10 times CPU than the former two methods. L0M can preserve angular distortion well, especially for meshes with many flat regions. However, L0M is the most time-consuming for it has to solve an expensive optimization model. Our method is more efficient than L0M, GNF, and BF, and is of the best denoising effect.

To verify the robustness of our new method, we compare further with these methods on some real

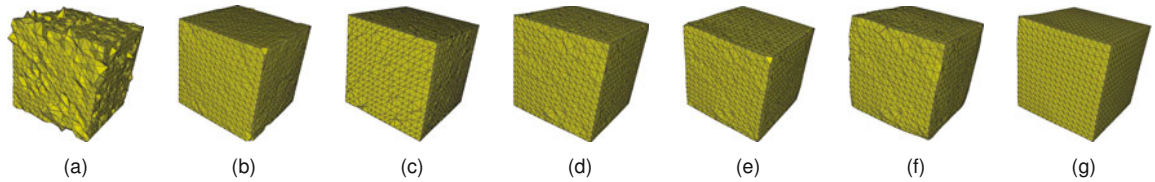

Fig. 11 Denoising a Cube model with impulsive noise distributed in a random direction: (a) noise level 0.4 for 50% input data; (b) denoised results of L0M ($\lambda = 0.002$, $\alpha = 0.008$, $\mu_\alpha = 1$); (c) denoised results of FED ($n_1 = 120$, $n_2 = 40$); (d) denoised results of LBNF ($\sigma_{\text{s}} = 0.35$, $n_1 = 200$, $n_2 = 50$); (e) denoised results of GNF ($m = 2.2$, $\sigma_{\text{r}} = 0.32$, $n_1 = 20$, $n_2 = 10$); (f) denoised results of BF ($\sigma_{\text{s1}} = 0.4$, $\sigma_{\text{s2}} = 0.4$, $n_1 = 10$, $n_2 = 10$, $\alpha = 0.3$, $\beta = 0.3$); (g) denoised results of our method ($\sigma_{\text{r}} = 0.32$, $n_1 = 150$, $\tau_{\text{a}} = 1.3$, $\beta = 0.02$, $t = F$)

Table 1 Performance comparison with other related methods

Model	δ ($\times 10^{-1}$)						D_{mean} ($\times 10^{-3}$)					
	L0M	FED	LBNF	GNF	BF	Ours	L0M	FED	LBNF	GNF	BF	Ours
Twirl	**1.090**	1.610	1.270	1.460	1.260	1.110	12.2	25.7	6.98	9.19	**5.19**	6.75
SharpSphere	1.420	1.520	1.550	1.270	1.800	**0.991**	25.5	12.4	**6.43**	6.46	9.96	6.85
Block	0.930	0.850	0.867	0.825	0.965	**0.645**	15.7	15.5	9.53	10.3	9.58	**7.75**
Fandisk	1.290	1.990	1.970	1.210	1.730	**1.180**	9.07	6.90	5.54	5.02	**4.56**	5.36
DoubleTorus	0.788	1.150	0.905	0.908	1.710	**0.756**	**12.9**	15.8	14.3	18.0	16.3	16.8
Cube	**0.533**	0.869	0.996	0.937	1.190	0.823	2.44	0.835	**0.771**	1.19	9.62	0.912

Model	$D_{\max}$ ($\times 10^{-2}$)						t (ms)					
	L0M	FED	LBNF	GNF	BF	Ours	L0M	FED	LBNF	GNF	BF	Ours
Twirl	15.30	47.8	9.99	14.10	72.40	**8.63**	16 958	**346**	503	4268	6310	1255
SharpSphere	17.80	15.3	10.70	9.40	10.60	**8.99**	27 622	**271**	722	5607	10 297	1633
Block	23.00	33.8	15.60	12.90	14.40	**8.51**	23 628 (41)	**371** (72)	415 (112)	2704 (140)	11 251 (120)	1315 (**1**)
Fandisk	8.40	80.7	6.89	4.75	5.18	**4.67**	14 052 (9)	**283** (37)	930 (15)	2390 (2)	7665 (80)	2578 (**0**)
DoubleTorus	14.50	35.9	12.60	18.30	20.20	**12.50**	3023 (**0**)	**192** (8)	325 (8)	1677 (18)	1245 (32)	933 (2)
Cube	2.04	0.602	0.687	1.08	1.79	**0.564**	2062 (7)	**186** (21)	721 (33)	723 (26)	746 (30)	2616 (**0**)

The numbers in the brackets describe the number of edges with dihedral angle larger than 150 degrees (to be reliable, we omit this item for models with very sharp features); the bold numbers indicate the corresponding best values. Twirl: $|V|$=8700, $|F|$=17 400; SharpSphere: $|V|$=10 443, $|F|$=2088; Block: $|V|$=8771; $|F|$=17 550; Fandisk: $|V|$=6475, $|F|$=12 946; DoubleTorus: $|V|$=2174, $|F|$=4335; Cube: $|V|$=1538, $|F|$=3072

scanned data in Fig. 12. Though there is little visual difference for most parts of the denoised meshes, at the top of the screwdriver as shown in the blow-up image, only our method restores the slim substructure well with the aid of edge-based CANs. Also, a close-up view at some challenging regions of the ManFace model, such as the ear (shown from another viewpoint) and the collar, indicates the superiority of our method over the others. Fig. 13 gives a Chair model with poor triangulation. From the enlarged parts of the wireframe images, we can notice flipping triangles of all the other methods except ours. These examples demonstrate the stronger capability of our new method over state-of-the-art methods in denoising meshes with slim structures and poor quality.

5.3 Limitations and discussion

Although there is an obvious superiority of our method over state-of-the-art mesh denoising methods, there are still some aspects to be polished. First, like all the previous methods, we cannot obtain desired denoising results for meshes with extremely irregular triangulation, as shown in Fig. 14. The reason lies in the fact that both normal filtering (Eq. (1)) and vertex updating (Eqs. (6), (12), and (14)) require the surrounding geometric elements to be symmetric about the central one, while extremely irregular meshes actually destroy the isotropic nature, no matter what neighborhoods are adopted. Resampling may be a good solution to this problem. Second, though we estimate guided normals more robustly than Zhang WY *et al.* (2015) with new CANs and the consistency measure, we cannot get rid of mistakes absolutely, which may push a face near features to a wrong structure and thus cause feature drifting. This is the reason why our method achieves the least angular errors but not the least distance errors in the examples of Fandisk and TrimStar models (Table 1). Last but not least, there are many parameters to be tuned by users, which is a professionally demanding job. The emerging data-driven learning methods will probably provide a perfect handling of this matter.

6 Conclusions and future work

In this paper, we have presented a novel filtering-based mesh denoising framework, which eliminates multifarious noise while preserving features and regularity of original meshes well. Substantial experiments on both synthetic and raw data indicated the superiority of our method over state-of-the-art methods. The good performance of our approach stems from the following three aspects. First, we have employed a novel corner-aware neighborhood and consistency measuring method to estimate guided face normals, which improves robustness greatly. In addition, we have provided an optional rolling guidance method to balance between efficiency and effectiveness. Second, to improve the quality of denoised meshes, we have imposed individual regularization constraints on distinct types of vertices. Third, we have taken an alternate vertex updating strategy according to the reliability of filtered normals, which is more efficient than the batch method. In the future, we will attempt to handle the problems mentioned in the limitation. With the anisotropic neighborhoods and patch-based operation, we want to deal with other problems in digital geometry processing, such as segmentation, feature extraction, and normal estimation.

Acknowledgements

The authors would like to appreciate Wang-yu ZHANG for providing executable programs. The models used in this paper are courtesy of the AIM Shape Repository, the Stanford 3D Scanning Repository, and Laser Design.

References

Belyaev, A., Ohtake, Y., 2003. A comparison of mesh smoothing methods. Proc. Israel-Korea Bi-National Conf. on Geometric Modeling and Computer Graphics, p.83-87.

Bian, Z., Tong, R.F., 2011. Feature-preserving mesh denoising based on vertices classification. *Comput. Aided Geom. Des.*, **28**(1):50-64. https://doi.org/10.1016/j.cagd.2010.10.001

Chen, C.Y., Cheng, K.Y., 2005. A sharpness dependent filter for mesh smoothing. *Comput. Aided Geom. Des.*, **22**(5):376-391. https://doi.org/10.1016/j.cagd.2005.04.003

Cho, H., Lee, H., Kang, H., *et al.*, 2014. Bilateral texture filtering. *ACM Trans. Graph.*, **33**(4):128.1-128.8. https://doi.org/10.1145/2601097.2601188

Desbrun, M., Meyer, M., Schröder, P., *et al.*, 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. Proc. 26th Annual Conf. on Computer Graphics and Interactive Techniques, p.317-324. https://doi.org/10.1145/311535.311576

Fan, H.Q., Yu, Y.Z., Peng, Q.S., 2010. Robust feature-preserving mesh denoising based on consistent

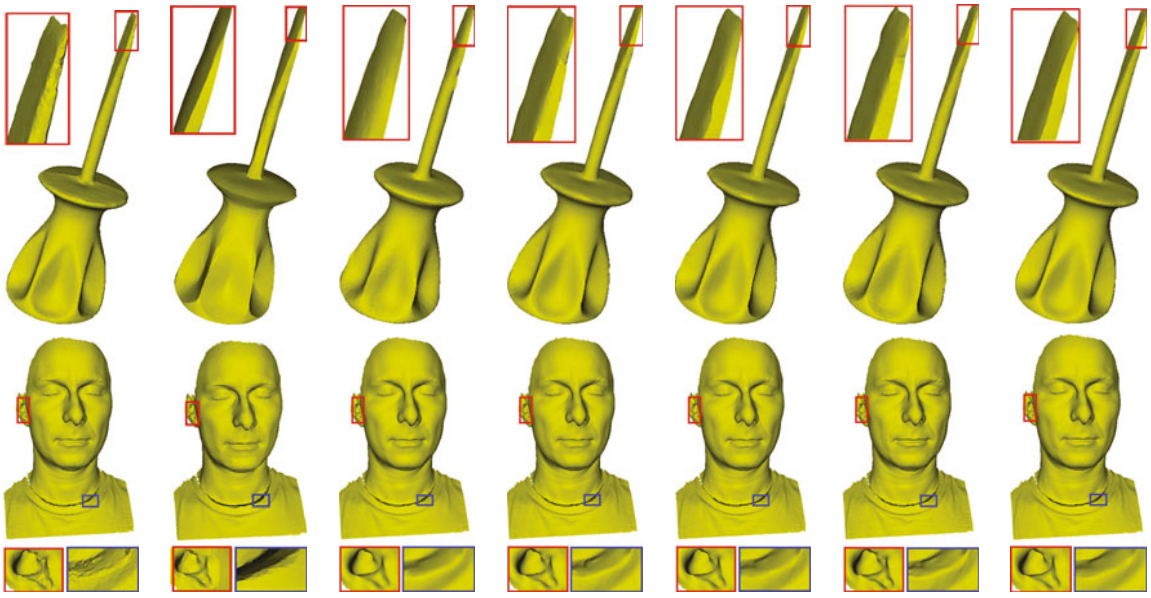

Fig. 12 Comparison with state-of-the-art denoising methods on real scanned data. For the noisy mesh (a), (b) to (g) are the denoising results of, for ScrewDriver: L0M ($\lambda = 2.0 \times 10^{-7}$, $\alpha = 0.004$, $\mu_\alpha = 0.9$), FED ($n_1 = 20$, $n_2 = 50$), LBNF ($\sigma_s = 0.35$, $n_1 = 10$, $n_2 = 10$), GNF ($m = 2.0$, $\sigma_r = 0.35$, $n_1 = 10$, $n_2 = 10$), BF ($\sigma_{s1} = 0.4$, $\sigma_{s2} = 0.4$, $n_1 = 8$, $n_2 = 8$, $\alpha = 0.3$, $\beta = 0.3$), and our method ($\sigma_r = 0.33$, $n_1 = 15$, $\tau_a = 1.3$, $\beta = 0.01$, $t = V$ with the aid of $t = E$); For ManFace: L0M ($\lambda = 0.05$, $\alpha = 0.002$, $\mu_\alpha = 1.1$), FED ($n_1 = 20$, $n_2 = 50$), LBNF ($\sigma_s = 0.35$, $n_1 = 10$, $n_2 = 10$), GNF ($m = 2.8$, $\sigma_r = 0.45$, $n_1 = 3$, $n_2 = 10$), BF ($\sigma_{s1} = 0.4$, $\sigma_{s2} = 0.4$, $n_1 = 8$, $n_2 = 8$, $\alpha = 0.3$, $\beta = 0.3$), and our method ($\sigma_r = 0.4$, $n_1 = 3$, $\tau_a = 1.35$, $\beta = 0.02$, $t = V$)

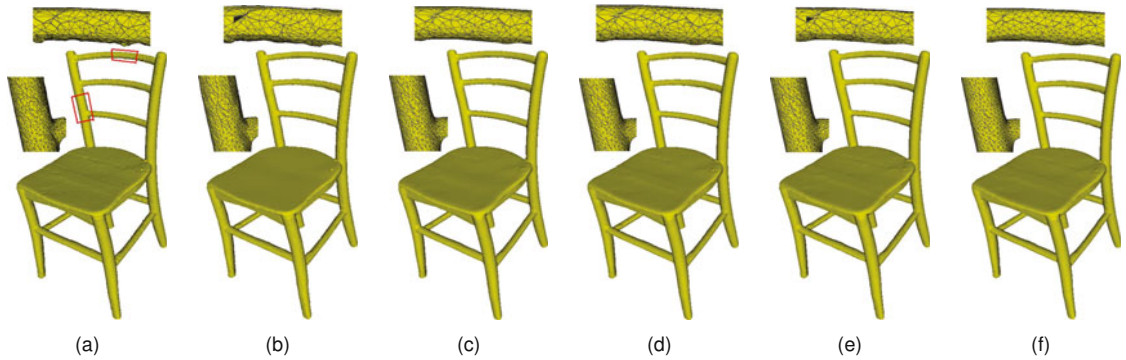

Fig. 13 Another real scanned example with poor quality: (a) noisy mesh; (b) denoising results of L0M ($\lambda = 0.09$, $\alpha = 0.005$, $\mu_\alpha = 1.1$); (c) denoising results of FED ($n_1 = 5$, $n_2 = 10$); (d) denoising results of LBNF ($\sigma_s = 0.36$, $n_1 = 4$, $n_2 = 10$); (e) denoising results of GNF ($m = 2.3$, $\sigma_r = 0.36$, $n_1 = 4$, $n_2 = 10$); (f) denoising results of our method ($\sigma_r = 0.35$, $n_1 = 4$, $\tau_a = 1.0$, $\beta = 0.08$, $t = F$)

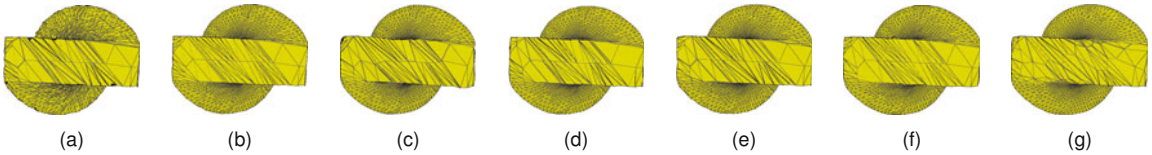

Fig. 14 Another view of Fig. 7. From the wireframe of the ground-truth mesh (a) we can notice very irregular triangulation at the base. (b) to (g) are denoising results of the corresponding six methods

subneighborhoods. *IEEE Trans. Vis. Comput. Graph.*, **16**(2):312-324. https://doi.org/10.1109/TVCG.2009.70

Fleishman, S., Drori, I., Cochen-Or, D., 2003. Bilateral mesh denoising. *ACM Trans. Graph.*, **22**(3):950-953. https://doi.org/10.1145/1201775.882368

He, L., Schaefer, S., 2013. Mesh denoising via ℓ_0 minimization. *ACM Trans. Graph.*, **32**(4):64.1-64.8. https://doi.org/10.1145/2461912.2461965

Jones, T.R., Durand, F., Desbrun, M., 2003. Non-iterative, feature-preserving mesh smoothing. *ACM Trans. Graph.*, **22**(3):943-949. https://doi.org/10.1145/1201775.882367

Liu, L.G., Tai, C.L., Ji, Z.P., *et al.*, 2007. Non-iterative approach for global mesh optimization. *Comput. Aided Des.*, **39**(9):772-782. https://doi.org/10.1016/j.cad.2007.03.004

Lu, X.Q., Deng, Z.G., Chen, W.Z., 2016. A robust scheme for feature-preserving mesh denoising. *IEEE Trans. Vis. Comput. Graph.*, **22**(3):1181-1194. https://doi.org/10.1109/TVCG.2015.2500222

Ohtake, Y., Belyaev, A., Bogaevski, I., 2001. Mesh regularization and adaptive smoothing. *Comput. Aided Des.*, **33**(11):789-800. https://doi.org/10.1016/S0010-4485(01)00095-1

Ohtake, Y., Belyaev, A., Yagou, H., 2002. Mesh smoothing via mean and median filtering applied to face normals. Proc. Geometric Modeling and Processing Conf., p.124-131.

Shen, J., Maxim, B., Akingbehin, K., 2005. Accurate correction of surface noises of polygonal meshes. *Int. J. Numer. Meth. Eng.*, **64**(12):1678-1698. https://doi.org/10.1002/nme.1441

Solomon, J., Crane, K., Butscher, A., *et al.*, 2014. A general framework for bilateral and mean shift filtering. arXiv:1405.4734.

Sun, X.F., Rosin, P., Martin, R., *et al.*, 2007. Fast and effective feature-preserving mesh denoising. *IEEE Trans. Vis. Comput. Graph.*, **13**(5):925-938. https://doi.org/10.1109/TVCG.2007.1065

Sun, X.F., Rosin, P., Martin, R., *et al.*, 2008. Random walks for feature-preserving mesh denoising. *Comput. Aided Geom. Des.*, **25**(7):437-456. https://doi.org/10.1016/j.cagd.2007.12.008

Taubin, G., 1995. A signal processing approach to fair surface design. Proc. 22nd Annual Conf. on Computer Graphics and Interactive Techniques, p.351-358. https://doi.org/10.1145/218380.218473

Taubin, G., 2001. Linear Anisotropic Mesh Filtering. United States Patent Application 20040075659, USA.

Wang, J., Zhang, X., Yu, Z.Y., 2012. A cascaded approach for feature-preserving surface mesh denoising. *Comput. Aided Des.*, **44**(7):597-610. https://doi.org/10.1016/j.cad.2012.03.001

Wang, P.S., Fu, X.M., Liu, Y., *et al.*, 2015. Rolling guidance normal filter for geometric processing. *ACM Trans. Graph.*, **34**(6):17.1-17.9. https://doi.org/10.1145/2816795.2818068

Wang, R.M., Yang, Z.W., Liu, L.G., *et al.*, 2014. Decoupling noise and features via weighted l_1 analysis compressed sensing. *ACM Trans. Graph.*, **33**(2):18.1-18.12. https://doi.org/10.1145/2557449

Wei, M.Q., Shen, W.Y., Qin, J., *et al.*, 2013. Feature-preserving optimization for noisy mesh using joint bilateral filter and constrained Laplacian smoothing. *Opt. Laser Eng.*, **51**(11):1223-1234. https://doi.org/10.1016/j.optlaseng.2013.04.018

Wei, M.Q., Yu, J.Z., Pang, W.M., *et al.*, 2015. Bi-normal filtering for mesh denoising. *IEEE Trans. Vis. Comput. Graph.*, **21**(1):43-55. https://doi.org/10.1109/TVCG.2014.2326872

Yagou, H., Ohtake, Y., Belyaev, A., 2003. Mesh denoising via iterative alpha-trimming and nonlinear diffusion of normals with automatic thresholding. Proc. Computer Graphics Int., p.28-34. https://doi.org/10.1109/CGI.2003.1214444

Zhang, H.Y., Wu, C.L., Zhang, J.Y., *et al.*, 2015. Variational mesh denoising using total variation and piecewise constant function space. *IEEE Trans. Vis. Comput. Graph.*, **21**(7):873-886. https://doi.org/10.1109/TVCG.2015.2398432

Zhang, W.Y., Deng, B.L., Zhang, J.Y., *et al.*, 2015. Guided mesh normal filtering. *Comput. Graph. Forum*, **34**(7): 23-34. https://doi.org/10.1111/cgf.12742

Zheng, Y.Y., Fu, H.B., Au, O.K.C., *et al.*, 2011. Bilateral normal filtering for mesh denoising. *IEEE Trans. Vis. Comput. Graph.*, **17**(10):1521-1530. https://doi.org/10.1109/TVCG.2010.264