Computer-Aided Design 121 (2020) 102807

Contents lists available at ScienceDirect

Computer-Aided Design

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

HLO: Half-kernel Laplacian operator for surface smoothing☆

Wei Pan [a], Xuequan Lu [b], Yuanhao Gong [a], Wenming Tang [a], Jun Liu [a], Ying He [c], Guoping Qiu [a,*]

[a] College of Information Engineering, Shenzhen University, Guangdong Key Laboratory of Intelligent Information Processing, China
[b] School of Information Technology, Deakin University, Australia
[c] School of Computer Science and Engineering, Nanyang Technological University, Singapore

ARTICLE INFO

Article history:
Received 11 May 2019
Received in revised form 16 September 2019
Accepted 17 December 2019

Keywords:
Half-kernel Laplacian
Surface denoising
Discrete Laplacian

ABSTRACT

This paper presents a simple yet effective and efficient method for feature-preserving surface smoothing. Through analyzing the differential property of surfaces, we show that the conventional discrete Laplacian operator with uniform weights is not applicable to feature points at which the surface is non-differentiable and the second order derivatives do not exist. To overcome this difficulty, we propose a Half-kernel Laplacian Operator (HLO) as an alternative to the conventional Laplacian. Given a vertex v, HLO first finds all pairs of its neighboring vertices and divides each pair into two subsets (called half windows); then computes the uniform Laplacians of all such subsets and subsequently projects the computed Laplacians to the full-window uniform Laplacian to alleviate flipping and degeneration. The half window with least regularization energy is then chosen for v. We develop an iterative approach to apply HLO for surface denoising. Our method is conceptually simple and easy to use because it has a single parameter, i.e., the number of iterations for updating vertices. We show that our method can preserve features better than the popular uniform Laplacian-based denoising and it significantly alleviates the shrinkage artifact. Extensive experimental results demonstrate that HLO is better than or comparable to state-of-the-art techniques both qualitatively and quantitatively and that it is particularly good at handling meshes with high noise. Also, it outperforms all other compared methods in terms of computational time. We make our executable program publicly available.

1. Introduction

Surface meshes acquired through scanning or sensing equipment are inevitably contaminated with noise. The user has to process them with smoothing techniques before applying to down stream applications, such as shape analysis, animation and rendering. The main technical challenge of surface smoothing is preserving features while removing noise and alleviating shrinkage. The design of robust surface smoothing methods is therefore of particular need nowadays.

The discrete Laplacian operator on surface meshes has proven highly useful in various tasks of digital geometry processing, for example, mesh fairing, parameterization, reconstruction, editing and compressing [1–6], just to name a few. The differential surface representation encodes information about the local shape of a surface such as the curvature and the orientation. Despite that the uniform Laplacian operator can effectively smooth surfaces, it fails to preserve features and leads to shrinkage.

Among the feature-preserving methods, the majority of them typically use the normal information of surfaces, such as face or vertex normals. In contrast, very few works are based on vertex position [7,8]. Some recent works extended these two types by adding more delicate steps [8–15]. However, these methods generally consist of multiple steps and numerous parameters, which are difficult to use/tune especially for complex models. Moreover, the users especially those out of the field may find it difficult and tedious to tune the involved parameters.

To overcome the above issues, we propose a novel, robust approach for feature-preserving mesh smoothing in this paper. Our key idea is to construct a "half-kernel" uniform Laplacian operator (HLO) that can approximate the Laplacians at feature and non-feature points using half windows. Specifically, we first analyze the differential property of feature points and found that it conflicts with the existence assumption of the uniform Laplacian. We then naturally propose a half-window algorithm to generate different pairs of subsets (half windows) for each vertex by pairing one immediate neighbor to the other unique neighbor. We compute the Laplacians of all subsets which are further projected to

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.cad.2019.102807.

* Corresponding author.
E-mail address: guoping.qiu@nottingham.ac.uk (G. Qiu).

https://doi.org/10.1016/j.cad.2019.102807
0010-4485/

the full-window Laplacians to alleviate flipping and degeneration. The final half-kernel Laplacian is automatically determined as the one that incurs the smallest regularization energy. The surface is finally updated with the determined half-kernel Laplacians in an iterative way.

Taking a noisy surface mesh as input, our approach can automatically output a quality version with preserving features and resisting shrinkage. The **main contributions** of this work are:

- mathematical analysis of the differential property at feature points;
- a half-window algorithm to generate multiple half windows (subsets) for each vertex;
- a half-kernel uniform Laplacian operator (HLO) for feature-preserving and shrinkage-resisting surface smoothing.

Our method[1] is conceptually simple and easy to use since it involves only a single parameter (i.e., the number of vertex update iterations). We demonstrate that the proposed HLO substantially outperforms the uniform Laplacian in preserving features and resisting shrinkage. We evaluate our method on both synthetic and real-world models, and observe that our method can produce results with comparable or higher quality than the state-of-the-art methods. We also show that our method is faster than the state-of-the-art methods.

2. Related work

There exists a large body of literature of mesh denoising. Due to space limit, we review only the mostly related works to ours and refer the readers to the comprehensive surveys [16,17].

The classical Laplacian smoothing methods [18,19] are simple and fast. However, its isotropic property leads to feature-wiping and shrinking artifacts. Taubin [20] proposed a non-shrinking, two-step smoothing method with positive and negative damping factors. Desbrun et al. [21] proposed a fairing method based on diffusion and curvature flow to process irregular meshes. Later, various isotropic smoothing methods have been introduced based on volume preservation, pass frequency controlling, differential properties, etc., [22–26].

The above isotropic methods are effective to remove noise, however they also wipe out features. Various anisotropic methods have been proposed to preserve features. Representative methods are diffusion/differential-based methods [7,27–34], bilateral filters [35–39], methods combining normal filtering and vertex update [14,15,35,37,38,40–43,43–49]. Bilateral filtering is initially used in image denoising [50], and it is successfully extended to mesh denoising. Regarding the feature-preserving techniques, the utilization of filtered normals has seen noticeable progress in recent years [14,15,35,37,38,40–43,43–49,51,52]. Some recent methods have focused on vertex and face classification before mesh denoising [9–13,53,54]. Nevertheless, the classification results depend largely on the level of noise. Lu et al. [8,14,15] presented the ideas of pre-filtering before real smoothing and can robustly handle heavy noise with frequent flipped triangle faces. Arvanitis et al. [55] introduced a novel coarse-to-fine graph spectral processing approach for mesh denoising.

Another line of anisotropic methods focused on the sparse perspective. Compared with non-feature vertices, feature vertices in a mesh are usually sparse, which can be reconstructed or detected by solving a sparse problem [56]. Sparsity was introduced into mesh smoothing in some recent works [7,8,57,58]. For instance, He et al. [7] developed a L_0 minimization framework with an area-based edge operator which is generally sparse in a surface mesh. It is, however, non-convex and difficult to solve. Zhao et al. [58] designed an improved alternating optimization strategy to solve the L_0 minimization which incorporates both vertex positions and face normals. Wang et al. [57] proposed a method to decouple noise and features by weighted L_1-analysis compressed sensing, and they prove that the pseudo-inverse matrix of the Laplacian of a mesh is a coherent dictionary for sparsely representing sharp feature signals on the shape. Recently, Lu et al. [8] detected features by introducing a novel L_1 minimization. Lu and his colleagues [59] proposed a low-rank matrix approximation approach for geometry filtering and demonstrated various geometry processing applications. More recently, the low-rank optimization was further extended to mesh denoising [60,61].

3. Half-kernel Laplacian operator

To better understand the proposed half-kernel Laplacian operator (HLO), we briefly introduce the uniform Laplacian on meshes and the Laplacian diffusion flow in the first place. We then analyze the differential property of feature points and finally explain how we construct the HLO.

3.1. Uniform Laplacian

Given a surface mesh $M = (V, E, F)$ with N vertices, we have the set of vertices V, the set of edges E and the set of faces F. The ith vertex $v_i \in V$ is represented by the coordinates $v_i = (x_i, y_i, z_i)$. The differential coordinates (i.e., δ-coordinates) of vertex v_i are defined as

$$\delta_i = v_i - \frac{1}{|NV(v_i)|} \sum_{v_k \in NV(v_i)} v_k, \tag{1}$$

where $\delta_i = (\delta_x, \delta_y, \delta_z)$, $NV(v_i)$ is the set of the neighboring vertices of the vertex v_i, and $|NV(v_i)|$ is the degree (number of neighbors) of v_i. It is also called the uniform Laplacian due to the equal weights. The direction of the differential coordinates approximates the local normal direction and the magnitude linearly approximates the local mean curvature $H(v_i)$ of v_i [4].

3.2. Laplacian diffusion flow

Fairing a mesh can be viewed as a filtering process on mesh signal. In the discretized setting, it is usually done by solving a heat-diffusion-like partial differential equation as follows.

$$\frac{\partial V(x,t)}{\partial t} = \lambda \triangle V(x,t), \tag{2}$$

where λ is the diffusion speed. The Laplacian operator is defined as

$$\triangle V = \nabla^2 V, \tag{3}$$

where $\nabla^2 V = \nabla \cdot \nabla$ is the divergence of the gradient on the vertices of a given mesh. The above equation (Eq. (2)) is usually solved in an iterative way

$$V^{t+1} = V^t + \lambda dt \nabla^2 V^t, \tag{4}$$

where dt is the time step length.

The surface smoothing based on the uniform Laplacian often causes the issues of shrinkage and smoothing out features (Fig. 2a). Motivated by these issues, we attempt to first analyze the differential property of feature points, which will be described in Section 3.3.

[1] Our EXE has been released on https://github.com/WillPanSUTD/hlo. See https://github.com/xuequanlu for more tools and results.

ALGORITHM 1: Half Windows Generation

Input: vertex v_i
Output: all subsets V_{sub}
$v_i' = \frac{1}{|NV(v_i)|}\sum_{v_k \in NV(v_i)} v_k$
for *each* $v_k \in NV(v_i)$ **do**
  compute the distance of other neighbors (except v_k) to the plane (or line in a degenerated case) defined by v_i, v_i' and v_k
  select the neighbor with the shortest distance, and pair it with v_k
  partition $NV(v_i)$ with the line v_i, v_k into the left and right subsets (i.e., two half windows)
end
enumerate all subsets V_{sub}

ALGORITHM 2: Half-kernel Laplacian Operator

Input: vertex v_i
Output: δ–final half-kernel Laplacian for v_i
$E_{init} = 10^6$
compute V_{sub} via Algorithm 1
compute the centroid v_i' with $NV(v_i)$
compute the local normal: $n_i = normalize(v_i - v_i')$
for *each subset* $v_{sub} \in V_{sub}$ **do**
  $d_i = \frac{1}{|V_{sub}|}\sum_{v_m \in v_{sub}}(v_i - v_m)$
  $\delta_i = (d_i \cdot n_i)n_i$
  if $\mathbb{E}(\delta_i) < E_{init}$ **then**
    $E_{init} = \mathbb{E}(\delta_i)$
    $\delta = \delta_i$
  end
end

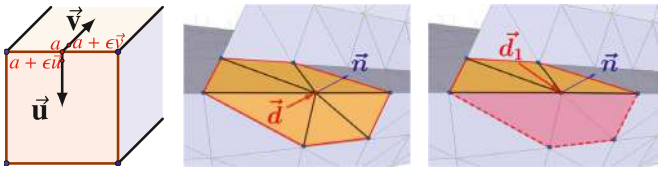
3.3. Differential analysis of feature points

Intuitively speaking, feature points can be classified as the points with abrupt normals (i.e., with large dihedral angles) [7, 14]. Denote by f a surface manifold and a a feature point at a sharp edge on the cube model (Fig. 1(a)). We simply analyze the situation when f crosses the sharp edge (Fig. 1(a)). It is obvious that $f'(a+\epsilon u) \neq f'(a+\epsilon v)$, where f' means the first-order derivative of f and $\epsilon > 0$. Thus, f is not differentiable at the feature point a, which means that the second-order derivative $f''(a)$ does not exist either. This reveals that it does not make sense to apply the uniform Laplacian (i.e., second-order derivative) operator to a feature vertex with all its neighbors (i.e., full window, see Fig. 1(a)). Fig. 2(a) shows the feature-wiping and shrinking results with the uniform Laplacian operator in a full-window sense.

Ideally, the approximations for locations $(a + \epsilon v)$ and $(a + \epsilon u)$ should come from the upper region and the front region, respectively. The approximation for a should base on its two neighboring vertices on the edge since it is differentiable along the edge [57]. Corner, the intersection of several edges, is theoretically non-differentiable and usually fixed in mesh denoising [57]. Sharp edge points and corners are difficult to approximate because they are identified with some means which make surface smoothing less robust due to sensitivity to the noise level. To overcome this difficulty, we attempt to approximate each (feature or non-feature) vertex with a half-kernel Laplacian operator which targets at computing the Laplacians using half-window neighbors. It will be discussed in Section 3.4.

3.4. The half-kernel Laplacian operator

The above observation and analysis indicate that the uniform Laplacian operator should be performed only on half windows (i.e., subsets of neighbors) of the feature point instead of its whole neighborhood. This key insight encourages us to introduce a half-kernel Laplacian operator (HLO) for surface meshes.



(a) Feature point (b) Uniform Laplacian (c) Half-window Laplacian

Fig. 1. (a) Analysis of a feature point a on an edge. The locations $(a + \epsilon v)$ and $(a+\epsilon u)$ should be approximated in the half windows which have the same colors with them, rather than the local full windows centered at them. The illustration of the uniform Laplacian operator (b) and the proposed HLO (c). The blue vector $\vec{n}$ represents the local normal of vertex v_i, and the red vector $\vec{d}$ indicates the orientations of the full-window and half-window Laplacians. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

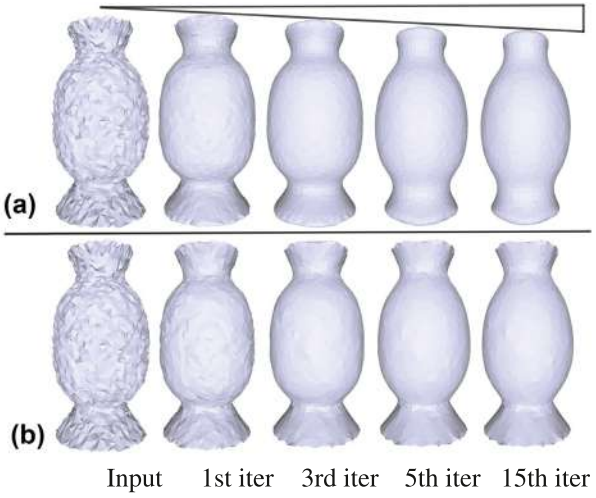


Fig. 2. Comparing the uniform Laplacian operator and the half-kernel Laplacian operator (HLO) on the noisy Vase. One can clearly see the shrinkage artifact of the uniform Laplacian. In contrast, HLO is geometry-aware and feature-preserving.

Our operator is to approximate the Laplacians at vertices with half windows. To produce half windows, each of the immediate neighboring vertex of the current vertex is paired with another neighboring vertex to partition the local neighborhood into two half windows (left and right). The other neighbor paired by the starting neighbor is determined as the neighboring vertex which has the shortest distance to the plane defined by the current vertex, the starting neighbor and the centroid of the neighborhood. Refer to Fig. 4(a). Consider a vertex v_0 with a neighbor v_D. Denote by v_A the starting neighbor to be paired. v_0' is the centroid of the neighborhood of v_0. Fig. 1(b) also shows a half window of a feature vertex. As a result, each vertex v_i and its neighborhood has $|NV(v_i)|$ partition choices which further generate $2|NV(v_i)|$ paired subsets (half windows). The half-window generation algorithm is shown in Algorithm 1.

Remark 1. Theoretically, the full window neighbors can be split into more than two half windows. We simply select the choice of two half windows, because corners are typically far fewer than the edge features in a shape, and recognizing corners may introduce new parameters and result in less robustness. Also, this half-window scheme is sufficient for non-feature vertices.

Remark 2. In general, there might exist multiple candidates with the same shortest distances when determining the other neighbor. We randomly select one candidate in this case, in which we

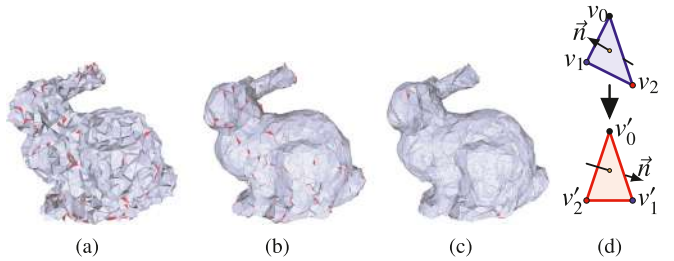


Fig. 3. With and without projection. (a) Noisy bunny. (b) 1 iteration of vertex update of the half-window Laplacian without projecting onto the uniform Laplacian. (c) 1 iteration of vertex update with the half-kernel Laplacian projecting onto the uniform Laplacian. The flipped triangles are rendered in red. (d) A flipped triangle. The clock-wise or counter-clock wise order of vertices has been changed after noise contamination. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

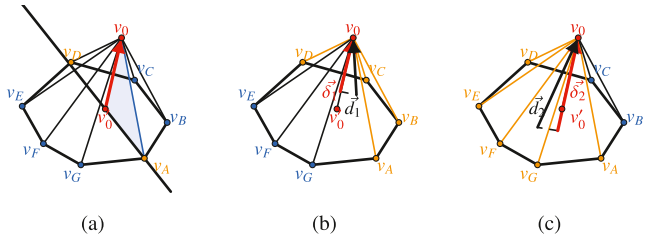


Fig. 4. The half-kernel Laplacian operator on a vertex v_0. (a) v_A, the starting neighbor of v_0, is paired with the other neighbor v_D which has the shortest distance to the plane $v_0v_Av_0'$. v_0' is the centroid of the neighbors. The line v_Av_D partitions the neighborhood into the left and right half windows (subsets). (b) and (c) d_1 and d_2 are computed by performing the uniform Laplacian operator to the half windows. δ_1 and δ_2 are achieved by projecting d_1 and d_2 to the full-window uniform Laplacians, respectively.

found no noticeable differences in mesh denoising. Note that the plane defined by the current vertex, the starting neighbor and the centroid of the neighborhood may degenerate into a line. We simply determine the other neighbor using the distance to the line instead.

We apply the uniform Laplacian operator independently to each of these subsets and compute the half-window Laplacians for each vertex v_i. Because half-kernel Laplacians introduce a shift in tangential direction, directly using the half-kernel Laplacians may result in inferior results like degenerating and flipping (Fig. 3).

We thus project them onto their corresponding full-kernel Laplacians (i.e., using full neighbors) to remove the tangential component and obtain the final half-kernel Laplacians. *This way ensures small Laplacian magnitudes for feature vertices and large Laplacian magnitudes for non-feature vertices, thus smoothing non-features and preserving features.* Fig. 4(b–c) shows two such examples. We define the projection in Eq. (5).

$$\delta = (d \cdot n)n, \tag{5}$$

where d indicates the half-kernel Laplacian and n is the orientation of the full-kernel Laplacian.

We can easily enumerate all the intermediate half-kernel Laplacians for v_i: $\ldots, \delta_L(v_k), \delta_R(v_k), \ldots$, where $\delta_L(v_k)$ and $\delta_R(v_k)$, respectively, denote the half-kernel Laplacians for the left subset and right subset with the starting neighbor v_k ($v_k \in NV(v_i)$). We next need to determine the optimal Laplacian among the $2|NV(v_i)|$ half-kernel Laplacians. The optimal one will be selected based on the regularization energy which may vary in different applications. In this work, we define the regularization energy of each vertex as the sum of the norm of the Laplacian and the distance to the original vertex position, shown as below.

$$\mathbb{E}(\delta_i^t) = \|\delta_i^t\| + \|v_i^t - v_i^0\|, \tag{6}$$

where δ_i^t can be any among the calculated $2|NV(v_i)|$ half-kernel Laplacians in the tth iteration. v_i^t is the position of vertex v_i in the tth iteration and v_i^0 is the initial position of v_i. The former term, which naturally uses the computed δ_i^t and is a feature-preserving term, represents the movement between two consecutive iterations, and the latter term, a typical data term, describes the distance between the position at the tth iteration and the initial position of vertex v_i. This energy enables a closest walking from the initial positions and positions in the previous iteration, thereby preserving features and the original shape. We compute the energy $\mathbb{E}$ for each half-kernel Laplacian of vertex v_i and select the optimal Laplacian that incurs the smallest energy. Algorithm 2 summarizes the proposed half-kernel Laplacian operator. To smooth surfaces, the final half-kernel Laplacians in the tth iteration are used to update vertex positions in the $(t+1)$th iteration via Eq. (4). λdt in Eq. (4) and the number of iterations work in a proportional way to each other. In other words, increasing λdt would generally induce a decrease of the number of update iterations, and vice versa. A larger λdt would also possibly cause instability. To make our method more robust, we empirically set λdt to 1, which works very well in all our experiments. Fig. 2(b) shows an example for our HLO.

4. Experimental results

We first describe the parameter setting, and then compare the proposed HLO with the uniform Laplacian operator (ULO). Finally, We compare our method with the state-of-the-art denoising methods on both synthetic and scanned data, visually and quantitatively.

4.1. Parameter setting

The compared state-of-the-art techniques are: the bilateral mesh filter (BMF) [36], the unilateral normal filter (UNF) [45], the bilateral normal filter (BNF) [37], the L_0 minimization method (L_0) [7], the guided normal filter (GNF) [43], the cascaded normal regression (CNR) [49], and the non-local low-rank normal filtering method (NLLR) [60]. The source code of these methods is available or the authors provide test results. We summarize the parameters of these methods in Table 1. Table 2 shows the parameter values of all methods on most models. *Interested readers are referred to the papers for more specific details.*

4.2. Comparison with the uniform Laplacian

Fig. 2 shows results processed by the uniform Laplacian and our HLO, respectively. We listed the corresponding results after 1, 3, 5, and 15 iterations. It is obvious that our HLO significantly alleviates the shrinking effect. The feature places (e.g., upper and lower parts) of the vase are preserved by HLO.

Fig. 5 visualizes the one-to-one vertex errors of both the uniform Laplacian and our HLO on the Julius model. Our method better preserves surface features and produces more accurate smoothing results than the uniform Laplacian operator. The average vertex errors and mean curvature energy [4,62,63] of Fig. 5 are displayed in Fig. 6. The HLO induces much lower average vertex errors than the uniform Laplacian operator, which means it preserves the shape better. The first iteration sees a remarkable

Table 1
Parameters of the state-of-the-art mesh smoothing methods.

Method	Number of parameters	Parameters
BMF	1	k_{iter}: number of iterations.
UNF	3	T: threshold for controlling the averaging weights. n_{iter}: number of iterations for normal update. v_{iter}: number of iterations for vertex update.
BNF(Local)	3	v_{iter}: number of iterations for vertex update. σ_s: variance parameter for the spatial kernel. n_{iter}: number of iterations for normal update.
L0	6	λ: weight for the L0 term in the target function. α_0, β_0: initial values for α and β μ_α, μ: update ratios for α and β. β_{max}: maximum value of β.
GNF	3	v_{iter}: number of iterations for vertex update. σ_r: variance parameter for the range kernel. n_{iter}: number of iterations for normal update.
NLLR	3	σ_M: the noise variance. N_k: number of similar vertices. v_{iter}: number of iterations for vertex update.
Ours	1	k_{iter}: number of iterations.

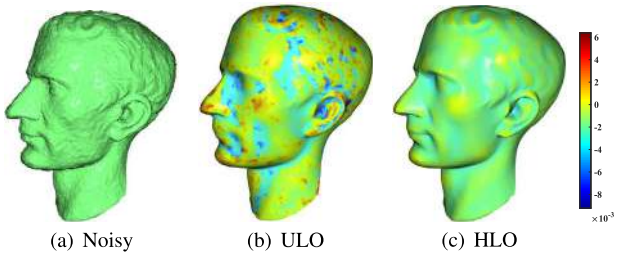


Fig. 5. The visualization of one-to-one vertex errors on the Julius model. (a) Noisy shape ($\sigma_n = 0.2l_e$). (b) 10 iterations of the uniform Laplacian operator (ULO). (c) 10 iterations of HLO. The errors are visualized using color maps, where the warm (resp. cold) colors denote positive (resp. negative) displacements. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

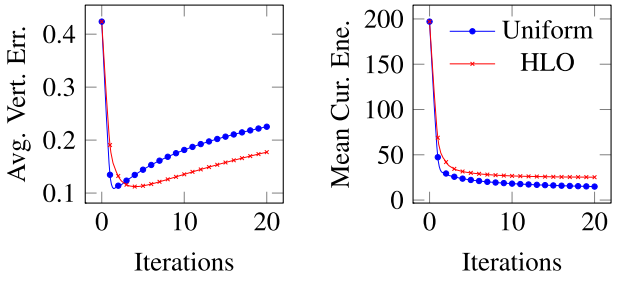


Fig. 6. The average vertex errors (Avg. Vert. Err.) and total mean curvature energy (Mean Cur. Ene.) of Fig. 5 with increasing iterations.

decrease on the mean curvature energy, and the two methods have a similar decreasing trend in mean curvature led by diffusion flow.

We also compared with the cotangent Laplacian. Fig. 7 shows the results of the uniform Laplacian, cotangent Laplacian and our HLO. The cotangent Laplacian is more geometry aware than the uniform Laplacian. However, the cotangent Laplacian tends to generate sharp "thorns" on the surface (please zoom in to observe).

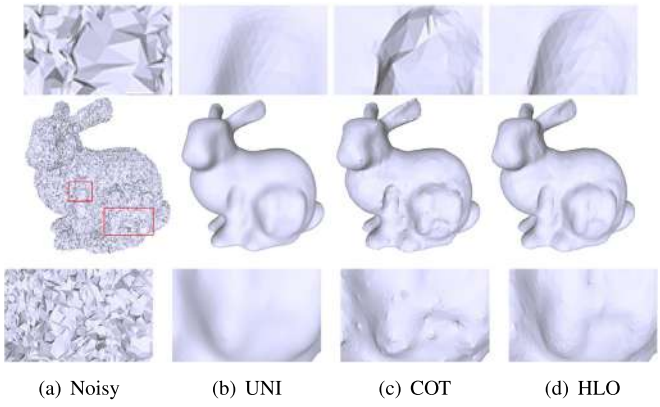


Fig. 7. The comparison of smoothing results after 10 iterations on (a) a noisy bunny model ($\sigma_n = 0.8l_e$). (b) The uniform Laplacian operator. (c) The cotangent Laplacian operator. (d) The half-kernel Laplacian operator (HLO).

4.3. Visual results

Synthetic models. We compare our method with the selected state-of-the-art methods on various models corrupted with synthetic noise. Following state-of-the-art mesh smoothing techniques, we generate synthetic models by adding zero-mean Gaussian noise with standard deviation σ_n to the corresponding ground truth. σ_n is proportional to the mean edge length l_e of the input mesh.

The uniform Laplacian operator can unfold the flipped triangles during surface smoothing. Thanks to this property, our approach is less sensitive to high-level noise than most of the compared methods, as shown in Figs. 8 and 9 (Bunny: $\sigma_n = 0.5l_e$, Nicolo: $\sigma_n = 0.5l_e$, Vaselion: $\sigma_n = 0.8l_e$). *The flipped triangles are rendered in red. See the close-up views.*

Fig. 8 shows visual results over the Armadillo model contaminated by Gaussian noise with $\sigma_n = 0.5e_l$. We can see from the blown-up windows that the state-of-the-art methods tend to oversmooth or oversharpen the fine details, or retain excessive noise in the model. We can observe from the mouth and eyes that our method well preserves the small-scale features which are usually lost to some extent in the results by the other methods. Fig. 9 (Bunny: $\sigma_n = 0.5l_e$ and Nicolo: $\sigma_n = 0.5l_e$) demonstrates that our approach preserves better surface details. Some methods

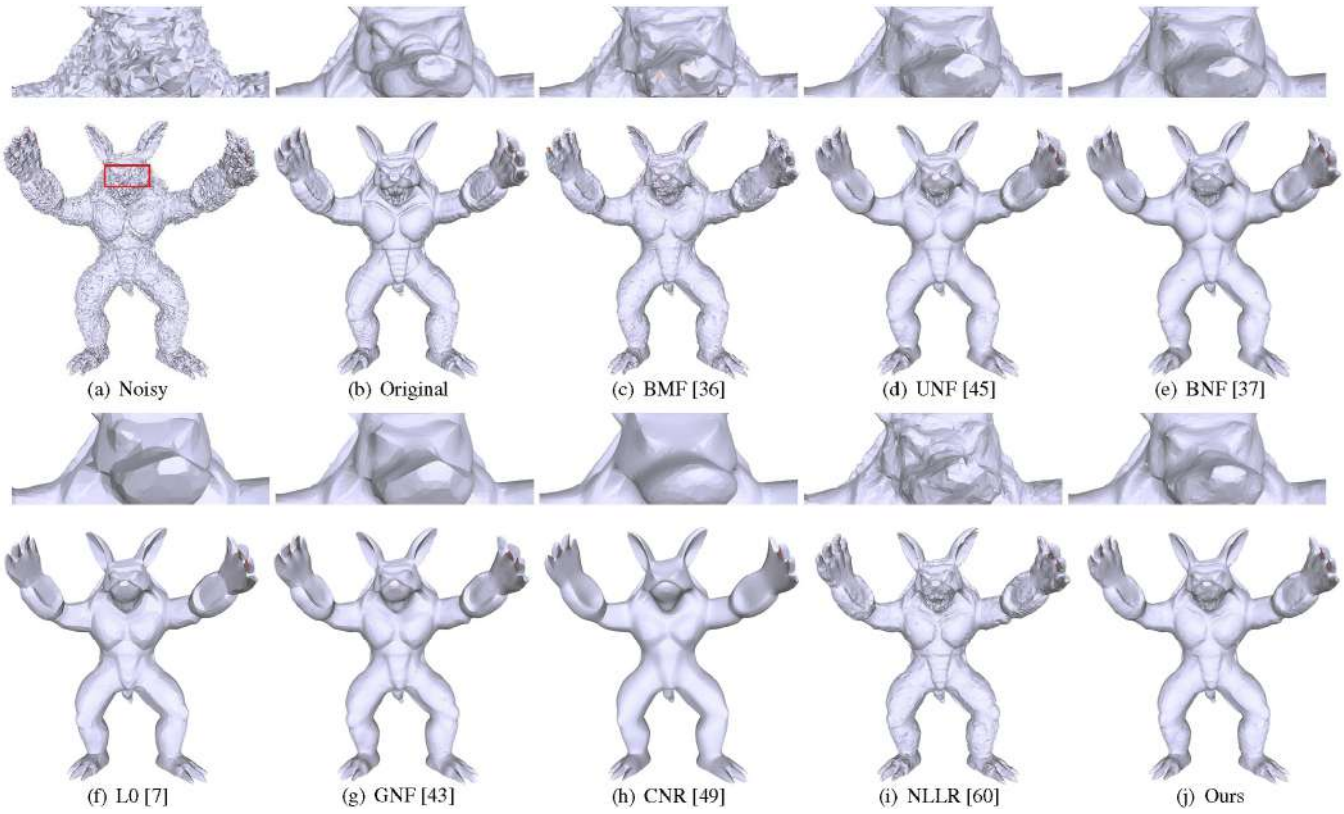


Fig. 8. Amadillo with Gaussian noise $\sigma_n = 0.5l_e$. The flipped triangles are rendered in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

may oversharpen the ear in Fig. 9 (Bunny: $\sigma_n = 0.5l_e$) and nose in Fig. 9 (Nicolo: $\sigma_n = 0.5l_e$). In Fig. 9 (Vaselion: $\sigma_n = 0.8l_e$), there are many curved features which are particularly difficult to recover by most existing methods when removing high-level noise. The method [7] can robustly remove the noise while it results in oversharpening in certain areas and loses some fine details. By contrast, our method outputs better results, in terms of features and details preservation. In Fig. 10, our method well recovers the sphere shape which has fewer folded triangles than other methods. *Notice that CNR [49] may produce better results if the model is trained on surface meshes with large noise.*

Undoubtedly, our method can also well handle small or medium noise. Fig. 9 (Nicolo: $\sigma_n = 0.2l_e$ and Vaselion: $\sigma_n = 0.2l_e$) shows two such examples. We can observe from Fig. 9 (Nicolo: $\sigma_n = 0.2l_e$) that our method and NLLR [60] generates similar results which are the best among all results. NLLR [60] uses non-local information for mesh smoothing while our method uses local information only. Fig. 9 (Vaselion: $\sigma_n = 0.2l_e$) demonstrates that the result by our HLO is better than the results by other techniques, in terms of noise removal and features preservation.

Raw scanned models. In addition to the synthetic models, we also tested the proposed HLO on scanned models corrupted with raw noise. Fig. 11 and Fig. 12 show the smoothing results of all methods on some real scanned surface meshes. Our method produces very competitive results, in terms of preserving features. Taking the second row in Fig. 11 as example, our HLO and NLLR [60] produce the best results while other methods may oversharpen or oversmooth certain regions. Note that NLLR [60] uses non-local information while ours utilizes local information only.

There are several reasons for why our method outperforms or is comparable to state-of-the-art techniques. (1) Our method is position based which largely mitigates the issue of flipped triangles. By contrast, most normal-based approaches (e.g., [37,45,60]) cannot effectively overcome this issue, without using any pre-processing strategy like [8,14]. [7] is also position based; however it may over-sharpen some regions unexpectedly, due to the edge-based operator on the whole shape. CNR [49] is a normal regression method that relies on immediate position update in each iteration; that is, it cannot directly predict the final normals. It thus easily smooths out features without a good mapping. (2) Multiple steps and parameters make those methods more complicated to search the solution space, thus leading to less automation and less robustness in handling different levels of noise, especially heavy noise. On the contrary, our method involves a single parameter, and is more simpler, automatic and robust in dealing with various noise.

4.4. Quantitative evaluations

Besides the above visual comparisons, we also compare the state-of-the-art techniques with our approach from a quantitative perspective. Specifically, we employ E_v and MSAE (mean square angular error) to respectively evaluate the positional error and normal error, as suggested by previous works [8,37,45]. These two metrics are calculated between the smoothing results and their corresponding ground truth.

According to [45], E_v is the L^2 vertex-based mesh-to-mesh error metric, and MSAE measures the mean square angular error between the face normals of the denoised mesh and those of the ground truth.

$$E_v = \sqrt{\frac{1}{3\sum_{k\in F} A_k} \sum_{i\in V} \sum_{j\in F_V(i)} A_j dist(x_i', T)^2}, \tag{7}$$

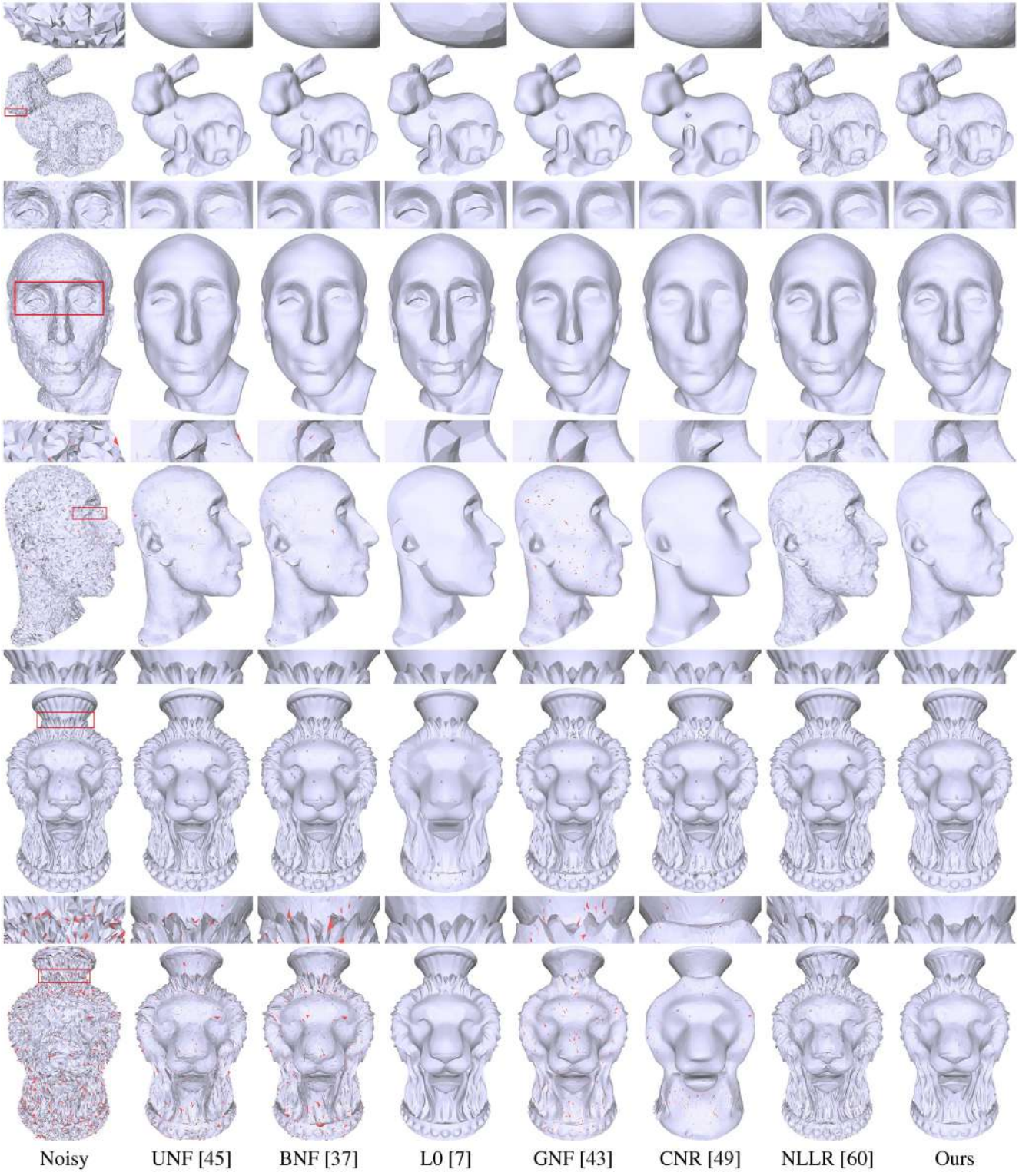


Fig. 9. Models corrupted with Gaussian noise: Bunny $\sigma_n = 0.5l_e$, Nicolo $\sigma_n = 0.2l_e$ and $\sigma_n = 0.5l_e$, Vaselion $\sigma_n = 0.2l_e$ and $\sigma_n = 0.8l_e$. The flipped triangles are rendered in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where A_k is the area of face k, and $dist(x'_i, T)$ is the L^2 distance between the updated vertex x'_i and a triangle of the reference mesh T which is closest to x'_i.

$$\text{MSAE} = \frac{\sum_{k \in F} \theta_k^2}{N_F} \tag{8}$$

where θ_k is the angle between the kth face normal of the denoised model and its corresponding normal in the ground-truth model, and N_F is the number of faces in the 3D shape.

Table 2 summarizes the statistical numbers of E_v and MSAE over most models for all the compared methods. As with previous works [8,45], we also found that the visual comparison results

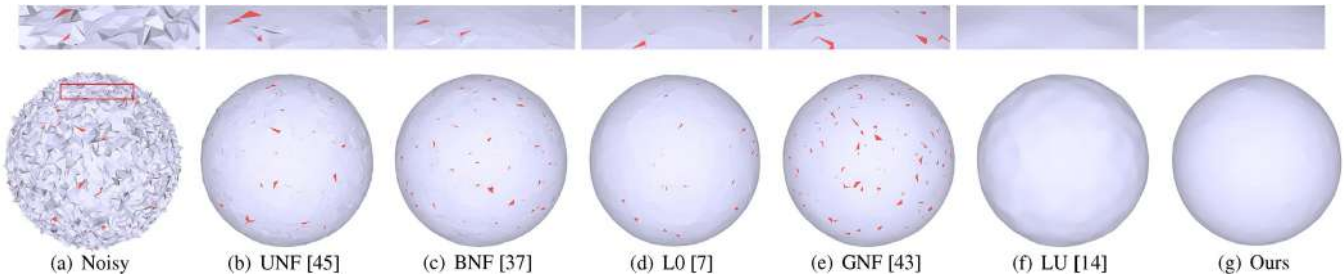


Fig. 10. Sphere with heavy Gaussian noise ($\sigma_n = 0.7l_e$). The flipped triangles are rendered in red. See the close-up views for details. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

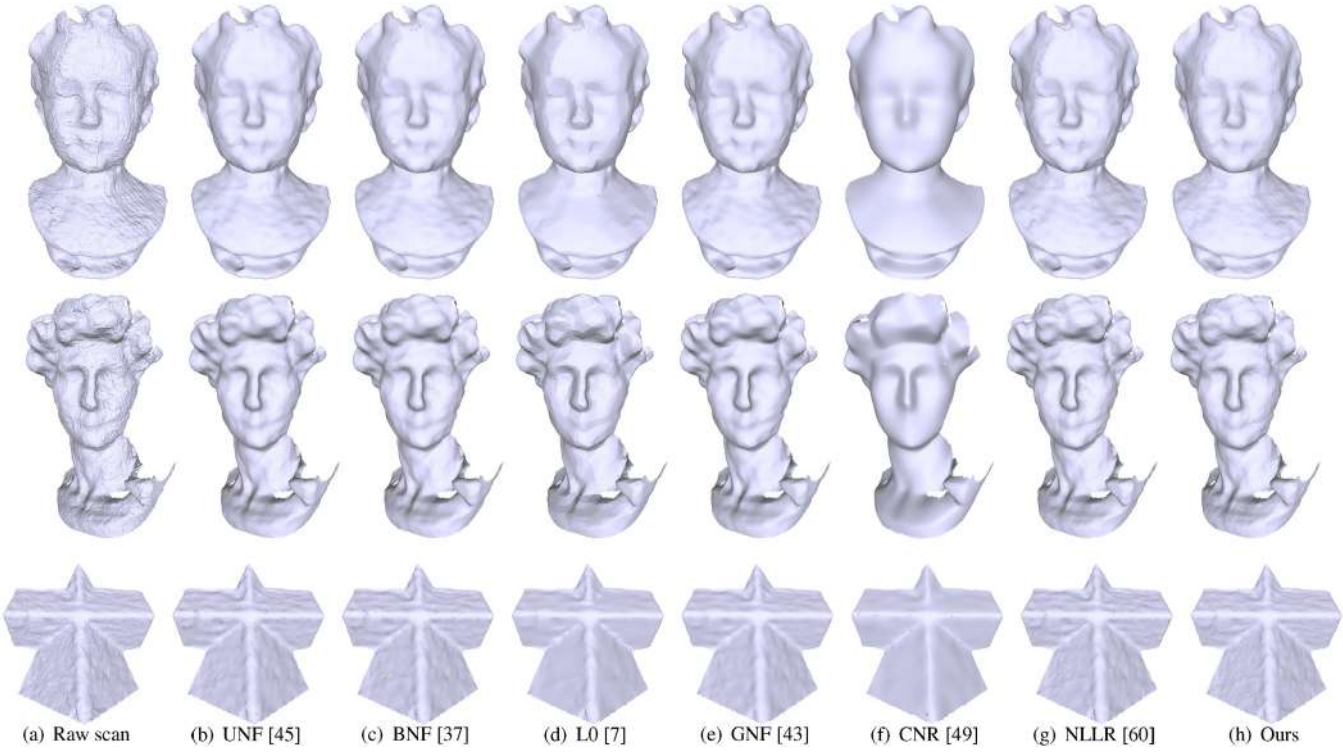


Fig. 11. Results of models scanned by Microsoft Kinect.

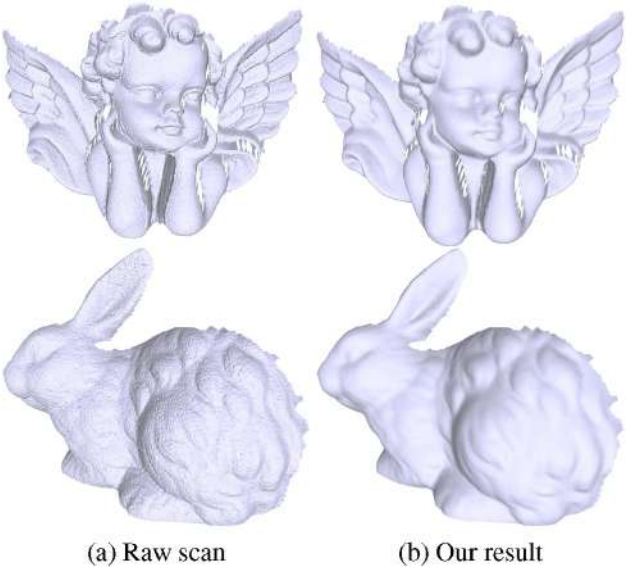


Fig. 12. Results of scanned models: Angel and Rabbit.

might not necessarily agree with the E_v. This is because our method smooths surfaces via unfolding the flipped triangles in the noisy input, which is similar to the behavior of [7]. As a consequence, vertices may walk far from their ground-truth locations. Regarding MSAE, the L_0 minimization [7] and our method are usually the best two among all techniques. We also compare the runtime of our method with those methods and list them in Table 3. Our approach outperforms those methods in terms of speed. It has excellent speed performance in handling models with large sizes (e.g., Amadillo and Vaselion in Fig. 10). It should be noted that our code has not been optimized or speeded up via parallelization.

5. Conclusion

In this work, we have introduced a novel method for feature-preserving surface smoothing. Motivated by the shrinking and feature-wiping issues of the uniform Laplacian operator, we analyzed the differential property at feature points. We developed a Half-kernel Laplacian Operator (HLO) which can preserve surface features and resist shrinkage. Various experiments show that our approach is better or comparable to state-of-the-art surface

Table 2
Quantitative comparisons with the representative mesh smoothing methods.

Models	Methods	MSAE ($\times 10^{-2}$)	E_v ($\times 10^{-3}$)	Parameters
Amadillo	BMF	47.54	3.34	(30)
($\sigma_n = 0.5l_e$)	BNF	53.58	3.11	(10, 0.35, 10)
(Fig. 8)	CNR	34.28	3.39	(*synthetic*, 10)
$\|V\|$: 43243	GNF	44.90	3.10	(10, 0.35, 10)
$\|F\|$: 86482	UNF	49.20	3.07	(0.5, 10, 10)
	L0	9.59	3.52	(*Default*)
	NLLR	32.18	**3.03**	(*Default*)
	OURS	**9.33**	3.57	(5)
iHbunny	BMF	5.07	4.99	(30)
($\sigma_n = 0.5l_e$)	BNF	3.57	4.39	(10, 0.35, 20)
(Fig. 9 row 1)	CNR	29.7	3.96	(*synthetic*, 10)
$\|V\|$: 34834	GNF	**3.39**	4.25	(10, 0.35, 20)
$\|F\|$: 69451	UNF	3.87	4.38	(0.4, 20, 10)
	L0	4.32	4.70	(*Default*)
	NLLR	25.24	**3.58**	(*Default*)
	OURS	6.25	4.32	(5)
Nicolo	BMF	16.20	0.96	(30)
($\sigma_n = 0.2l_e$)	BNF	8.97	0.92	(10, 0.35, 20)
(Fig. 9 row 2)	CNR	8.56	0.96	(*synthetic*, 3)
$\|V\|$: 14846	GNF	9.42	0.94	(10, 0.35, 20)
$\|F\|$: 29437	UNF	9.77	0.93	(0.5, 20, 10)
	L0	4.44	0.93	(*Default*)
	NLLR	7.13	0.93	(*Default*)
	OURS	**3.35**	**0.89**	(3)
Nicolo	BMF	56.33	1.13	(50)
($\sigma_n = 0.5l_e$)	BNF	65.33	1.00	(10, 0.45, 20)
(Fig. 9 row 3)	CNR	41.35	1.22	(*synthetic*, 10)
$\|V\|$: 14846	GNF	58.15	0.93	(10, 0.45, 20)
$\|F\|$: 29437	UNF	61.66	0.94	(0.45, 20, 10)
	L0	**5.67**	1.02	(*Default*)
	NLLR	38.44	**0.92**	(*Default*)
	OURS	10.33	1.08	(5)
Vaselion	BMF	38.33	34.47	(30)
($\sigma_n = 0.2l_e$)	BNF	33.76	14.20	(10, 0.35, 10)
(Fig. 9 row 4)	CNR	45.25	14.49	(*synthetic*, 1)
$\|V\|$: 38728	GNF	44.75	14.40	(10, 0.35, 10)
$\|F\|$: 77452	UNF	39.07	13.84	(0.5, 10, 10)
	L0	35.84	15.30	(*Default*)
	NLLR	35.00	**12.51**	(*Default*)
	OURS	**14.49**	14.80	(3)
Vaselion	BMF	167.9	30.06	(50)
($\sigma_n = 0.8l_e$)	BNF	171.9	12.58	(10, 0.45, 30)
(Fig. 9 row 5)	CNR	126.0	14.82	(*synthetic*, 10)
$\|V\|$: 38728	GNF	166.6	13.98	(10, 0.45, 30)
$\|F\|$: 77452	UNF	158.7	13.26	(0.5, 20, 10)
	L0	**14.72**	16.49	(*Default*)
	NLLR	82.25	**11.29**	(*Default*)
	OURS	32.35	15.87	(10)
Boy	BMF	10.03	5.00	(30)
(kinect fusion)	BNF	9.67	4.89	(10, 0.35, 20)
(Fig. 11 row 1)	CNR	9.70	4.95	(*kinectfusion*, 10)
$\|V\|$: 76866	GNF	8.50	5.06	
$\|F\|$: 152198	UNF	9.42	4.90	(10, 0.35, 20)
	L0	11.21	4.90	(0.5, 10, 10)
	NLLR	9.61	4.91	(*Default*)
	OURS	**8.06**	**4.86**	(*Default*)
				(10)
David	BMF	10.74	1.60	(30)
(kinect fusion)	BNF	10.70	1.572	(10, 0.45, 20)
(Fig. 11 row 2)	CNR	15.85	2.09	(*kinectfusion*, 5)
$\|V\|$: 51789	GNF	**9.14**	1.65	(10, 0.45, 20)
$\|F\|$: 101937	UNF	10.28	**1.566**	(0.45, 10, 10)
	L0	12.59	1.60	(*Default*)
	NLLR	10.39	1.58	(*Default*)
	OURS	9.32	2.12	(10)

(*continued on next page*)

Table 2 (*continued*).

Models	Methods	MSAE ($\times 10^{-2}$)	E_v ($\times 10^{-3}$)	Parameters
Pyramid	BMF	6.70	3.68	(30)
(kinect fusion)	BNF	6.98	3.45	(10, 0.45, 20)
(Fig. 11 row 3)	CNR	5.41	3.45	(*kinectfusion*, 3)
$\|V\|$: 35261	GNF	**5.40**	**3.28**	(10, 0.45, 20)
$\|F\|$: 69611	UNF	7.43	3.43	(0.6, 20, 10)
	L0	5.59	3.29	($\sqrt{2}$, 0.1, 0.1)
	NLLR	6.79	3.29	(*Default*)
	OURS	6.62	3.58	(5)

Table 3
Runtime comparisons of several state-of-the-art methods with our method (in s). The vertex iterations of all the methods are set to 10.

Models	UNF	BNF	GNF	L0	NLLR	OURS
Amadillo (Fig. 8) $\|V\|$: 43243 $\|F\|$: 86482	5.68	7.12	117.5	55.8	75.9	**1.81**
iHbunny (Fig. 9 Row 1) $\|V\|$: 34834 $\|F\|$: 69451	3.80	3.15	35.6	102.4	21.4	**1.39**
Nicolo (Fig. 9 Row 2) $\|V\|$: 14846 $\|F\|$: 29437	0.89	0.83	28.0	23.8	19.0	**0.59**
Vaselion (Fig. 9 Row 5) $\|V\|$: 38728 $\|F\|$: 77452	6.52	7.23	84.6	54.2	27.8	**1.54**
Sphere (Fig. 10) $\|V\|$: 10242 $\|F\|$: 20480	0.39	0.54	8.97	16.1	15.1	**0.38**

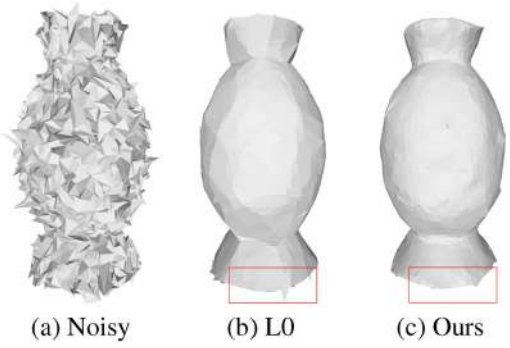


Fig. 13. (a) The vase model with noise ($\sigma_n = 1.5l_e$). (b) The smoothing result of L0 [7]. (c) The smoothing result of our HLO after 10 iterations.

smoothing techniques, in terms of visual quality and quantitative evaluations. Our method is robust to high noise and it has a single parameter (the number of vertex update iterations) which is easy and intuitive to tune.

Our method still suffers from a few limitations. First, it has difficulty to recover smooth sharp edges for CAD-like models with a small number of vertices (Fig. 13). This is because HLO is not an edge-based operator [7]. For real scanned CAD-like models with large number of vertices, our method can produce results with comparable quality to the state-of-the-art methods (Fig. 11 row 3). As with previous works [8,37], we also fixed the open

boundaries of the input noisy mesh, which might sometimes lead to undesired results.

As the future work, we would like to incorporate new schemes to overcome the issue of recovering smooth sharp edges. We would also like to improve the performance via parallelization.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61907031. Xuequan Lu is supported by Deakin, Australia CY01-251301-F003-PJ03906-PG00447. Ying He is supported by MOE, Singapore RG26/17.

References

[1] TaubinÝ G. Geometric signal processing on polygonal meshes. In *Proceedings of EUROGRAPHICS 2000: state of the art report*; 2000.

[2] Zhang H. Discrete combinatorial laplacian operators for digital geometry processing. In: Proceedings of SIAM conference on geometric design and computing. Nashboro Press; 2004, p. 575–92.

[3] Floater MS, Hormann K. Surface parameterization: a tutorial and survey. In: Dodgson NA, Floater MS, Sabin MA, editors. Advances in multiresolution for geometric modelling. Berlin, Heidelberg: Springer Berlin Heidelberg; 2005, p. 157–86.

[4] Sorkine O. Differential representations for mesh processing. In: Computer graphics forum, vol. 25. Wiley Online Library; 2006, p. 789–807.

[5] Váša L, Marras S, Hormann K, Brunnett G. Compressing dynamic meshes with geometric Laplacians. Comput Graph Forum 2014;33(2):145–54, Proceedings of Eurographics.

[6] Hu K, Zhang YJ, Li X, Xu G. Feature-aligned surface parameterization using secondary Laplace operator and loop subdivision. In: 25th international meshing roundtable. Procedia Eng 2016;163:186–98. http://dx.doi.org/10.1016/j.proeng.2016.11.047, http://www.sciencedirect.com/science/article/pii/S1877705816333525.

[7] He L, Schaefer S. Mesh denoising via l0 minimization. ACM Trans Graph 2013;32(4):64:1–8. http://dx.doi.org/10.1145/2461912.2461965.

[8] Lu X, Deng Z, Chen W. A robust scheme for feature-preserving mesh denoising. IEEE Trans Vis Comput Graphics 2016;22(3):1181–94. http://dx.doi.org/10.1109/TVCG.2015.2500222.

[9] Wei M, Yu J, Pang W, Wang J, Qin J, Liu L, Heng P. Bi-normal filtering for mesh denoising. Visual Comput Graph IEEE Trans 2015;21(1):43–55.

[10] Fan H, Yu Y, Peng Q. Robust feature-preserving mesh denoising based on consistent subneighborhoods. IEEE Trans Vis Comput Graphics 2010;16(2):312–24.

[11] Bian Z, Tong R. Feature-preserving mesh denoising based on vertices classification. Comput Aided Geom Design 2011;28(1):50–64. http://dx.doi.org/10.1016/j.cagd.2010.10.001, http://www.sciencedirect.com/science/article/pii/S0167839610001111.

[12] Wang J, Zhang X, Yu Z. A cascaded approach for feature-preserving surface mesh denoising. Comput Aided Des 2012;44(7):597–610. http://dx.doi.org/10.1016/j.cad.2012.03.001, http://www.sciencedirect.com/science/article/pii/S0010448512000516.

[13] Zhu L, Wei M, Yu J, Wang W, Qin J, Heng P-A. Coarse-to-fine normal filtering for feature-preserving mesh denoising based on isotropic subneighborhoods. Comput Graph Forum 2013;32(7):371–80. http://dx.doi.org/10.1111/cgf.12245.

[14] Lu X, Chen W, Schaefer S. Robust mesh denoising via vertex pre-filtering and l1-median normal filtering. Comput Aided Geom Design 2017;54:49–60. http://dx.doi.org/10.1016/j.cagd.2017.02.011, http://www.sciencedirect.com/science/article/pii/S0167839617300638.

[15] Lu X, Liu X, Deng Z, Chen W. An efficient approach for feature-preserving mesh denoising. Optics Lasers Eng 2017;90:186–95. http://dx.doi.org/10.1016/j.optlaseng.2016.09.003, http://www.sciencedirect.com/science/article/pii/S0143816616301981.

[16] Botsch M, Pauly M, Kobbelt L, Alliez P, Lévy B. Geometric modeling based on polygonal meshes. In: Eurographics tutorial. 2008.

[17] Botsch M, Kobbelt L, Pauly M, Alliez P, Lévy B. Polygon mesh processing. CRC press; 2010.

[18] Vollmer J, Mencl R, Müller H. Improved Laplacian smoothing of noisy surface meshes. Comput Graph Forum 1999;131–8.

[19] Field DA. Laplacian smoothing and Delaunay triangulations. Commun Appl Numer Methods 1988;4(6):709–12. http://dx.doi.org/10.1002/cnm.1630040603.

[20] Taubin G. A signal processing approach to fair surface design. In: Proceedings of the 22nd annual conference on computer graphics and interactive techniques. ACM; 1995, p. 351–8.

[21] Desbrun M, Meyer M, Schröder P, Barr AH. Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proc. of SIGGRAPH'99. 1999, p. 317–24.

[22] Liu X, Bao H, Shum H-Y, Peng Q. A novel volume constrained smoothing method for meshes. Graph Models 2002;64(3–4):169–82. http://dx.doi.org/10.1006/gmod.2002.0576.

[23] Kim B, Rossignac J. GeoFilter: geometric selection of mesh filter parameters. Comput Graph Forum 2005;295–302.

[24] Nehab D, Rusinkiewicz S, Davis J, Ramamoorthi R. Efficiently combining positions and normals for precise 3D geometry. ACM Trans Graph 2005;24(3):536–43. http://dx.doi.org/10.1145/1073204.1073226.

[25] Nealen A, Igarashi T, Sorkine O, Alexa M. Laplacian mesh optimization. In *Proceedings of GRAPHITE'06*; 2006. p. 381–9.

[26] Su Z-X, Wang H, Cao J-J. Mesh denoising based on differential coordinates. In *Proc. of IEEE Int'L conf. on shape modeling and applications 2009*; 2009. p. 1–6.

[27] Tasdizen T, Whitaker R, Burchard P, Osher S. Geometric surface smoothing via anisotropic diffusion of normals. In *Proceedings of IEEE conference on visualization '02*; 2002. p. 125–32.

[28] Clarenz U, Diewald U, Rumpf M. Anisotropic geometric diffusion in surface processing. In *Proc. of IEEE conference on visualization '00*; 2000. p. 397–405.

[29] Desbrun M, Meyer M, Schröder P, Barr AH. Anisotropic feature-preserving denoising of height fields and bivariate data. In: Graphics interface. 2000, p. 145–52.

[30] Bajaj CL, Xu G. Anisotropic diffusion of surfaces and functions on surfaces. ACM Trans Graph 2003;22(1):4–32. http://dx.doi.org/10.1145/588272.588276.

[31] Hildebrandt K, Polthier K. Anisotropic filtering of non-linear surface features. Comput Graph Forum 2004;23(3):391–400. http://dx.doi.org/10.1111/j.1467-8659.2004.00770.x.

[32] Ohtake Y, Belyaev AG, Bogaevski IA. Polyhedral surface smoothing with simultaneous mesh regularization. In *Proc. of the geometric modeling and processing 2000*; 2000. p. 229–37.

[33] El Ouafdi A, Ziou D. A global physical method for manifold smoothing. In *Proc. of IEEE international conf. on shape modeling and applications 2008*; 2008. p. 11–7.

[34] El Ouafdi A, Ziou D, Krim H. A smart stochastic approach for manifolds smoothing. Comput Graph Forum 2008;27(5):1357–64. http://dx.doi.org/10.1111/j.1467-8659.2008.01275.x.

[35] Jones TR, Durand F, Desbrun M. Non-iterative, feature-preserving mesh smoothing. ACM Trans Graph 2003;22(3):943–9. http://dx.doi.org/10.1145/882262.882367.

[36] Fleishman S, Drori I, Cohen-Or D. Bilateral mesh denoising. ACM Trans Graph 2003;22(3):950–3. http://dx.doi.org/10.1145/882262.882368.

[37] 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. http://dx.doi.org/10.1109/TVCG.2010.264.

[38] Lee K-W, Wang W-P. Feature-preserving mesh denoising via bilateral normal filtering. In *Proc. of int'l conf. on computer aided design and computer graphics 2005*; 2005.

[39] Wang CC. Bilateral recovering of sharp edges on feature-insensitive sampled meshes. Visual Comput Graph IEEE Trans 2006;12(4):629–39. http://dx.doi.org/10.1109/TVCG.2006.60.

[40] Taubin G. Linear anisotropic mesh filtering. IBM research report RC22213(W0110-051), IBM T.J. Watson Research; 2001.

[41] Ohtake Y, Belyaev A, Bogaevski I. Mesh regularization and adaptive smoothing. Comput Aided Des 2001;33(11):789–800. http://dx.doi.org/10.1016/S0010-4485(01)00095-1, http://www.sciencedirect.com/science/article/pii/S0010448501000951.

[42] Chen C-Y, Cheng K-Y. A sharpness dependent filter for mesh smoothing. Comput Aided Geom Design 2005;22(5):376–91.

[43] Zhang W, Deng B, Zhang J, Bouaziz S, Liu L. Guided mesh normal filtering. Comput Graph Forum 2015;34(Special Issue of Pacific Graphics 2015):23–34.

[44] Sun X, Rosin PL, Martin RR, Langbein FC. Random walks for feature-preserving mesh denoising. Comput Aided Geom Design 2008;25(7):437–56. http://dx.doi.org/10.1016/j.cagd.2007.12.008, http://www.sciencedirect.com/science/article/pii/S0167839608000307, Solid and Physical Modeling Selected papers from the Solid and Physical Modeling and Applications Symposium 2007 (SPM 2007) Solid and Physical Modeling and Applications Symposium 2007.

[45] 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.

[46] Shen Y, Barner K. Fuzzy vector median-based surface smoothing. IEEE Trans Vis Comput Graphics 2004;10(3):252–65.

[47] Yagou H, Ohtake Y, Belyaev A. Mesh smoothing via mean and median filtering applied to face normals. In *Geometric modeling and processing, 2002. Proceedings*; 2002. p. 124–31. http://dx.doi.org/10.1109/GMAP.2002.1027503.

[48] Yagou H, Ohtake Y, Belyaev A. Mesh denoising via iterative alpha-trimming and nonlinear diffusion of normals with automatic thresholding. In *Computer graphics international, 2003. Proceedings*; 2003. p. 28–33. http://dx.doi.org/10.1109/CGI.2003.1214444.

[49] Wang P-S, Liu Y, Tong X. Mesh denoising via Cascaded normal regression. ACM Trans Graph (SIGGRAPH Asia) 2016;35(6).

[50] Tomasi C, Manduchi R. Bilateral filtering for gray and color images. In *ICCV'98*; 1998. p. 839–46.

[51] Yadav SK, Reitebuch U, Polthier K. Mesh denoising based on normal voting tensor and binary optimization. IEEE Trans Vis Comput Graphics 2018;24(8):2366–79. http://dx.doi.org/10.1109/TVCG.2017.2740384.

[52] Yadav SK, Reitebuch U, Polthier K. Robust and high fidelity mesh denoising. IEEE Trans Vis Comput Graphics 2018;1. http://dx.doi.org/10.1109/TVCG.2018.2828818.

[53] Wang J, Yu Z. A novel method for surface mesh smoothing: applications in biomedical modeling. In *Proceedings of the 18th international meshing roundtable*; 2009. p. 195–210.

[54] Wei M, Liang L, Pang W, Wang J, Li W, Wu H. Tensor voting guided mesh denoising. IEEE Trans Autom Sci Eng 2017;14(2):931–45.

[55] 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. http://dx.doi.org/10.1109/TVCG.2018.2802926.

[56] Avron H, Sharf A, Greif C, Cohen-Or D. L1-sparse reconstruction of sharp point set surfaces. ACM Trans Graph 2010;29(5):135:1–135:12. http://dx.doi.org/10.1145/1857907.1857911.

[57] Wang R, Yang Z, Liu L, Deng J, Chen F. Decoupling noise and features via weighted L1-analysis compressed sensing. ACM Trans Graph 2014;33(2):18:1–18:12. http://dx.doi.org/10.1145/2557449.

[58] Robust and effective mesh denoising using L0 sparse regularization. Comput Aided Des 2018;101:82–97. http://dx.doi.org/10.1016/j.cad.2018.04.001.

[59] Lu X, Schaefer S, Luo J, Ma L, He Y. Low rank matrix approximation for geometry filtering. CoRR 2018;abs/1803.06783. http://arxiv.org/abs/1803.06783.

[60] 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. http://dx.doi.org/10.1111/cgf.13556, https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13556.

[61] 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 2018;1. http://dx.doi.org/10.1109/TVCG.2018.2865363.

[62] Desbrun M, Meyer M, Schröder P, Barr AH. Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proceedings of the 26th annual conference on computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co.; 1999, p. 317–24.

[63] Gong Y, Sbalzarini IF. Curvature filters efficiently reduce certain variational energies. IEEE Trans Image Process 2017;26(4):1786–98. http://dx.doi.org/10.1109/TIP.2017.2658954.