EUROGRAPHICS 2017 / L. Barthe and B. Benes
(Guest Editors)

Volume 36 (2017), Number 2

Mesh denoising via generalized intrinsic filter on 2d manifold

Abstract

Weighted average is one of the most common strategy used in various digital signal filters, and their performances depend on the weight design. When computing the weight between the current sample and one of its neighbours, existing methods consider only properties of the two samples, such as positions and values. However, if the two samples belong to different regions which are separated by a narrow feature edge, even when their properties are close, it is improper to assign a large weight which will damage the feature.
In this paper, we present an intrinsic filter on 2D manifold. It estimates the weight between the current point and its neighbour based on the integral of some properties along the geodesic path connecting them. Therefore, features are better preserved when removing noises. It is also a very generalized model and many classic filters are just special cases of it. Furthermore, by projections on the tangent plane, we introduce a simple pattern to estimate the weight, which is faster and more effective than using geodesic path. Finally, the proposed filter is applied to mesh denoising, and experiments illustrate the efficacy of our method comparing with state-of-the-art methods.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

Due to the popularity of 3D scanners, meshes are becoming more and more accessible. However, noises from various sources are still unavoidable for many generated meshes. Mesh filtering becomes a vital tool for denoising. It can also be used for smoothing and texture removing. Although many mesh filters yield promising results [JDD03, ZFAT11, HS13, ZDZ*15, WFL*15], preserving various features when denoising remains a challenge.

Many traditional signal or image filters replace the signal value at each sample by a weighted average of signal values from nearby samples, and only Euclidean distance between samples is considered when computing the weight between the current sample and one of its neighbours. Such filters fail to preserve features. The bilateral filter, introduced in [TM98], is a famous edge-preserving image filter. Its weights depend not only on the spatial closeness, but also on the intensity difference. Recently, the guided filters [PSA*04, HST10] further set the filtering weights using the intensity differences from another image and yield better edge-preservation. Due to their success in image processing and computational photography, researchers make many attempts expand them to mesh denoising and smoothing [JDD03, ZFAT11, SCBW14, ZDZ*15].

However, the above methods have two defects. First, choosing a large neighborhood will result in smoothed edges, while selecting a small one would limit the ability of removing noise. Second, they can not handle close-by edges well. When there are close-by edges, samples with both close Euclidean distance and range value may come from different regions separated by another narrow region. Hence, for such sample pairs, both of them should contribute nothing to the filtering of the other one, shown in the figure 1. The reason behind is that these methods compute the filtering weight between a pair of samples only using the properties of the two samples. The samples between them are ignored. Hence we wish to take the geodesic path between them into consideration to solve above problems effectively. Although the geodesic filter [GS09] and propagated image filter [CW15] also use the samples on the geodesic path when computing weights for superior edge-preservation. They are not expanded to mesh filtering and they do not provide a generalized model for the problem.

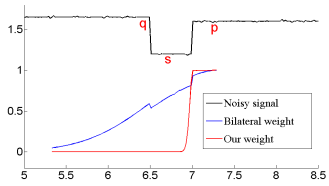

Figure 1: *When filtering the signals on p and q, if the p and q are close and their signal values are equal, classical filters offer a large weight. It is not reasonable. Our method solves this problem by the middle signal value on s.*

In our paper, we introduce an intrinsic filtering model for geometry signal (position, normal, curvature and so on) on 2D manifold. It is an universal model and many state-of-the-art filters are special cases of it, such as bilateral filter, geodesic filter and propagated filter. We estimate the weight between the current point and its neighbor based on the integral of the p norm of some properties' difference along the geodesic path between them, which yields to better edge-preservation when denoising and smoothing. Based on the generalized filtering model, we present a novel mesh denoising method following the common meshing denoising framework. The position of vertices are updated after filtering the face normals, and the two steps are iterated until achieving a satisfactory results. However, the generation of geodesic path for each face and its neighbors during all the iterations are extremely time consuming. We design a particular pattern on each face's tangent plane to approximate the integral along the geodesic path. It is very efficient and yields even better experimental results. Besides, considering that even geometrically complex shapes can be characterized by a rather small number of features, which means large normal differences are sparse, L1 norm is employed in the integral. Benefiting from the intrinsic filtering model and the above two enhancements for meshes, extensive experiments illustrate the efficacy of our mesh denoising approach and it achieves more desirable results comparing with the state-of-the-art mesh denoising methods.

2. Relate work

Mesh denoising, as a basic tool for geometry processing, has been under the attention of researchers all the time [WLDB08]. Due to the large amount of research work in this field, we only introduce some methods highly related to ours. Most of these methods are based on the weighted average on the local neighborhood of a mesh.

The purpose of mesh denoising is to remove the noise without damaging the true geometry structure, like edges and corner. After the bilateral filter is introduced in [TM98], it is extended in mesh denoising [FDCO03, JDD03, ZFAT11, SCBW14] because its outstanding ability. Fleishman et al [FDCO03] and [JDD03] apply bilateral filter to the mesh vertex positions and perform the similar neighborhood weighted average strategy. The difference is that [JDD03] estimates vertex positions through predicting the vertex tangent plane.

Due to the facet normal can better display the local feature of mesh, other researchers apply filtering methods to the normal field, then recover the surface by the relation of face vertices and face normal. [ZFAT11] employ the bilateral filter to the mesh face normals, then the process of vertex updated is implemented according the filtered normals. [SCBW14] also perform mesh denoising by filtering the face normals, but they use another filter rule, a generalized cross-bilateral filter. The effectiveness of bilateral filter mainly relies on the range function which reflects the local information of signal, and then applies the averaging weights as the filtered output. However, the difference between the input signal can not provides reliable prediction for that between the desired signal, especially in the region that separated by edge structure. To solve this problem, inspired by the image joint bilateral filter [**?**, PSA*04], [ZDZ*15] constructs a mesh joint filter method by estimating the normal of facets in a local region and achieves big success. However, the construction method of [ZDZ*15] may appear ambiguity in the sharp corners. These methods [ZFAT11,SCBW14,ZDZ*15] only consider the spatial and range distances between a pair of face barycenters. They may damage the weak edges because of the small range distance and smooth the sharp corners.

The two-stage process including filtering face normals and updating vertex positions has also been adapted by many famous work [YOB02, CC05, SRML08, WFL*15]. The mainly difference among these methods is their normal filtering strategies. Mean and median filtering and rolling guidance filter [ZSXJ14] are applied in [YOB02] and [WFL*15], respectively. [CC05] automatically selects filters according the mesh local sharpness. In [SRML08], face normal filtering is performed by weighted averaging of normals based on the concept of random walks. Our denoising algorithm also applies the similar approach which iterates face normal filtering and vertex updates. But, we build the relations of face normals between a pair of face barycenters connected by a geodesic path, obtaining the more accurate filtering weight for denoising algorithm.

Another type of denoising methods employ different strategies to filter the type of vertices which belonging to corner, edge or flat areas on a mesh. [BT11] classifies the vertex based on the volume integral invariant, while [FYP10] uses a local quadric model to fit the vertex and curvature tensor for obtaining the types of vertices. Wang et al [WZY12] use projection techniques to update each vertex of the mesh. Wei et al [WYP*15] apply the normal tensor voting strategy to classify the vertex.

Recently, sparsity optimization technique is used in mesh denoising, and obtains satisfactory result. The reason behind is that mesh structures are usually sparse. [HS13] applies L_0 minimization to an edge-based Laplacian operator and effectively preserve the sharp structure of mesh. However, the optimization algorithm prefers flat shapes and may not be suitable for complex meshes. Wang et al [WYL*14] use L_1 optimization to recover sharp structures from noisy meshes and also has above-mentioned problems.

3. Intrinsic filtering for 2D manifold surface

In this section, the detail of the intrinsic filtering model will be introduced for 2D manifold surfaces. Furthermore, we explain that almost all classical filtering methods are special case of ours. And, we give an application to demonstrate the efficacy of our filter algorithm.

3.1. Filtering for 2D manifold surface

Suppose the signals we are interested in are defined on a 2D manifold surface Ω and with values in domain Γ. Then the filtered signal $\bar{J}_p$ at point p can be calculated by the following general form:

$$\bar{J}_p = \frac{1}{W_p} \int_{\mathcal{N}(p)} \omega_{p,q} J_q d_q \,, \tag{1}$$

where, J_q denotes the input signal at point q, $\mathcal{N}(p)$ is the neighborhood of point p, $\omega_{p,q}$ indicates the weight between p and q, and $W_p = \int_{\mathcal{N}(p)} \omega_{p,q} d_q$ is the normalization factor.For signal filter,

the most important thing is how to construct $\omega_{p,q}$, since it directly relates to the performance of edge-preserving.

In the following, we define a weight which includes the cumulative difference of a path between two points p and q:

$$\omega_{p,q} = g(d^s(\varphi_p,\varphi_q);\sigma_s)g(d^r(\psi_p,\psi_q);\sigma_r)\,, \tag{2}$$

where, $g(\cdot;\cdot)$ is the kernel function. We choose Gaussian function because it decreases quickly. σ_s and σ_r are variance parameters. d^s and d^r are defined as:

$$\begin{cases} d^s(\varphi_p,\varphi_q) = \lim_{\Delta s\to 0}(\int_L ||\varphi_{q_{s+\Delta s}} - \varphi_{q_s}||^n ds\,)^{1/n} \\ d^r(\psi_p,\psi_q) = (\int_L ||\psi_s - \psi_p||^m ds\,)^{1/m} \end{cases}. \tag{3}$$

Note that L is the geodesic path between p and q. φ and ψ are two different signals, such as position, Gaussian curvature, and normal of a 2D manifold surface. $q_{s+\Delta s}$, q_s and s are points on path L.

The cumulative differences d^s and d^r are able to reflect the local structure of desired signal. The main reason is that the two cumulative differences are calculated on geodesic path and uses the signal difference. According to different problems, we can choose suitable parameters φ, ψ, n and m.

Equation 3 can be approximated as:

$$\begin{cases} d^s(\varphi_p,\varphi_q) = (\sum_{q_{i+1},q_i\in L} \|\varphi_{q_{i+1}} - \varphi_{q_i}\|^n \|q_{i+1} - q_i\|\,)^{1/n} \\ d^r(\psi_p,\psi_q) = (\sum_{q_{i+1},q_i\in L} \|\psi_{q_i} - \psi_p\|^m \|q_{i+1} - q_i\|\,)^{1/m} \end{cases}. \tag{4}$$

In the above formula, $\|q_{i+1} - q_i\|$ is step length. The orderly coordination of image can be regard as a discretization of 2D manifold. And the step length is equal to 1 in image, so according to our formulation, Bilateral filter[TM98], Geodesic filter[GS09], and image Propagation filter[CW15] are special cases of our model:

Case 1. Bilateral filter [TM98], as a classical nonlinear filter, uses two Gaussian functions as the spatial and range weights, respectively. In our generalized model 4, we use starting and end point p and q to replace the geodesic path L. At the same time, signal functions φ and ψ are defined as $\varphi_x = x$ and $\psi_x = I_x$, where x is the pixel position and I_x represents the pixel intensity. When $m = 2$ and $n = 2$, our model is simplified as:

$$\begin{cases} d^s(p,q) = ||p - q|| \\ d^r(I_p,I_q) = ||I_p - I_q|| \end{cases}. \tag{5}$$

We can see that the above formulation gets the distance differences, which are used to calculate the weights in bilateral filtering.

Case 2. Geodesic filter [GS09] only uses d^s, one kind of accumulative difference in adjacent pixels during the process of image denoising. This way gives a high response in image edges, so has a better edge-preserving power than bilateral filtering. Setting $n = 2$, the image intensity value I as a signal value φ, and $d^r(\psi_p,\psi_q) = 1$, our intrinsic filtering can be transformed into the following form:

$$\begin{cases} d^s(I_p,I_q) = (\sum_{q_{i+1},q_i\in L} ||I_{q_{i+1}} - I_{q_i}||^2\,)^{1/2} \\ d^r(\psi_p,\psi_q) = 1 \end{cases}. \tag{6}$$

Case 3. In a similar way, our intrinsic filtering is simplified as a propagation image filtering [CW15] by setting $n = 2$, $m = 2$, φ and ψ both represent the pixel intensity. Therefore,

$$\begin{cases} d^s(I_p,I_q) = (\sum_{q_{i+1},q_i\in L} ||I_{q_{i+1}} - I_{q_i}||^2\,)^{1/2} \\ d^r(I_p,I_q) = (\sum_{q_i\in L}) ||I_{q_i} - I_p||^2\,)^{1/2} \end{cases}. \tag{7}$$

Propagation filter [CW15] is a successful application of our method in image filtering. [CW15] also demonstrates an outstanding power in preserving the image edge feature.

Figure 2: *Comparison between geodesic paths and square-pattern paths. Left: the noisy mesh selected from [WYL*14]. Middle: the result of applying geodesic paths. Right: the result of applying square-pattern paths. The parameters of these two strategies are the same ($k_{iter} = 30$, $v_{iter} = 2$, $r = 4$, $\sigma_r = 0.3$). The time consumption is about 22.76s and 2.55s respectively. The former is not good in dealing with straight edges.*

Our intrinsic filtering builds connections between p and q, which providing more reliable information about the structure of the desired output. And deciding the filtering weights is more reasonable and reliable. Figure 1 also shows this fact. Further, we find our method inherits the advantage of these filtering algorithms [TM98, GS09, CW15], while compensating their deficiencies.

3.2. Filtering mesh geometry

Given a noisy triangle mesh, our goal is to filter the noise while keeping the mesh structures. In order to achieve this purpose, we also adapt a two-stage process which be widely used in mesh filtering. Firstly, noisy face normals are filtered iteratively by our intrinsic filtering model. Secondly, according to the filtered face normals, vertex positions are updated.

Filtering face normals. When considering normals as a signal defined over the triangle mesh, it is easy to use the intrinsic filtering algorithm for mesh denoising. First, the triangle face f_i outward normal $\mathbf{n_i}$ can be easily calculated. Then we consider $\mathbf{n_i}$ as a signal associated with the face barycenter c_i. Next, we need find a path L to connect $\mathbf{n_i}$ and its neighbor $\mathbf{n_j}$. In order to filter the face normals $\mathbf{n_i}$, we need find a path L to connect $\mathbf{n_i}$ and its neighbor $\mathbf{n_j}$. Finally, a filtered face normal $\bar{\mathbf{n}}_\mathbf{i}$ is computed through our intrinsic mesh filtering model:

$$\bar{\mathbf{n}}_\mathbf{i} = \frac{1}{W_i}\sum_{f_j\in\mathcal{N}_i} A_j g(d^s(\mathbf{n_i},\mathbf{n_j});\sigma_s)g(d^r(\mathbf{n_i},\mathbf{n_j});\sigma_r)\mathbf{n_j}\,. \tag{8}$$

here,

$$\begin{cases} d^s(\mathbf{n_i},\mathbf{n_j}) = (\sum_{x,x+1\in L} ||\mathbf{n_{x+1}} - \mathbf{n_x}||^n\,)^{1/n} \\ d^r(\mathbf{n_i},\mathbf{n_j}) = (\sum_{x\in L} ||\mathbf{n_x} - \mathbf{n_i}||^m\,)^{1/m} \end{cases}, \tag{9}$$

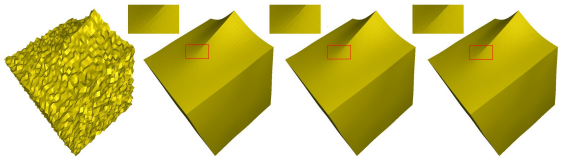

Figure 3: *From the left to right: Noised mesh (Gaussian noise with $0.3\sigma_E$), the results of $m = n = 2$ with step length, $m = n = 2$ without step length, $m = n = 1$ without step length. Because of the sparseness of mesh feature, L_1 norm receives the best results in preserving the mesh structure.*

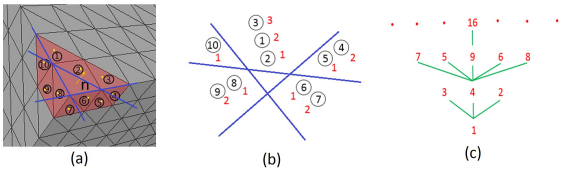

Figure 4: *The process of generation path. (a) the 10 neighbors of current face with its normal* $\mathbf{n}$*. (b) The schematic diagram after projection, dividing and sorting. (c) the pattern. The numbers with circle are the neighbor of current triangle face, the yellow points are the projection points of neighbor face centroid, the blue lines are used to decide the face label, the red numbers are their serial number after sorting, the green lines are connected path.*

where, W_i is the normalization factor, calculated by $||\sum_{f_j \in \mathcal{N}_i} A_j g(d^a(\mathbf{n_i}, \mathbf{n_j}); \sigma_s) g(d^r(\mathbf{n_i}, \mathbf{n_j}); \sigma_r)||$ which ensures that $\bar{\mathbf{n}}_\mathbf{i}$ is a unit normal; A_j is the area of f_j; $\mathcal{N}_i$ is the neighborhood of f_i. In our paper, we use the geometrical neighborhood, defined in [ZDZ*15].

Note that, equation 8 can be derived from equation 1. A_j is the accumulation of infinitesimal area because the domain of integration is triangle face. Furthermore, we give up the step length which measures the distance between adjacent barycenter of triangle faces. We simply set it to one for facilitating the calculation.

Although the geodesic algorithm gives the shortest distance between two points, its calculation is expensive. To improve computational efficiency, we introduce a simple, fast and effective strategy to chose filtering paths, which will be in the next section. In Figure 2, we compare the calculation efficiency and results obtained by these two strategies. The proposed strategy is much faster than geodesic distance. Furthermore, the result obtained by our strategy is much better in dealing with details, like edges and corners.

To chose appropriate m and n, we conduct several experiments. Figure 3 shows the experimental results. From this figure we can see that both m and n are set to 1 can obtain the best result, since features on 3D meshes are always sparse.

Updating vertices. After all face normals are filtered, the vertex positions need to be updated to cater to these new normals. We adopt the iterative scheme in the paper [SRML07] to update the vertex positions. Namely, for a face f_i, its vertex positions are updated via the following iteration form

$$\bar{v}_i^{(t+1)} = \bar{v}_i^{(t)} + \frac{1}{|\mathcal{F}_i|} \sum_{j \in \mathcal{F}_i} \bar{\mathbf{n}}_\mathbf{j} [\bar{\mathbf{n}}_\mathbf{j} \cdot (\bar{c}_j^{(t)} - \bar{v}_j^{(t)})], \qquad (10)$$

where, $\bar{v}_j^{(t)}$ is the value of $\bar{v}_i$ in the $t-th$ iteration, $\mathcal{F}_i$ is the index set of the incident faces for $\bar{v}_i$, $|\cdot|$ denotes the cardinality of a set, and $\bar{c}_j^{(t)} = \sum_{i=1}^{3} \bar{v}_{j_i}^{(t)}/3$ is the barycenter of the triangle f_j. Gradient descend process can be used to solve this formulation.

In our experiments, almost all models need less than 75 iterations to achieve satisfactory results for filtering face normals, and approximately up to above conditions about 5 iteration in the process of vertex update. Pseudocode of our mesh filter algorithm is shown in Algorithm 1. In the next section, we will solve the problem in selecting filter path.

Algorithm 1 Intrinsic mesh filtering framework

Input: Initial mesh M_{in}, number of iterations k_{iter}.
Output: Filtered mesh M_{out}.
1: $M^{(0)} = M_{in}$
2: Compute the path sets $\mathcal{P}$ according to section 4.
3: **for** $s = 1$ to k_{iter} **do**
4: Compute face normals n_i of mesh $M^{(s-1)}$;
5: Get the paths $\mathcal{P}_i$ of n_i;
6: Compute filtered normals $\bar{\mathbf{n}}_\mathbf{i}$ according to 8;
7: Compute updated mesh $M^{(s)}$ according to $\bar{\mathbf{n}}_\mathbf{i}$;
8: **end for**
9: $M_{out} = M^{k_{iter}}$.

4. Path chosen for mesh filtering

For an image, many particular patterns can be chosen because of the regular coordination system. These patterns are simple and easy to be thought about. But for a triangle mesh, it is very difficult to obtain a good pattern which can preserve the mesh feature. The easiest way is to use geodesic distance defined on triangle mesh.However, as seen in figure 2, this method is time consuming.

To solve this problem, we propose a simple strategy to chose path. Our method is based on the following theory: the areal coordinates of triangles segment the plane into seven regions. Areal coordinates are extremely useful in engineering applications involving triangular subdomains. These make analytic integrals often easier to evaluate, and Gaussian quadrature tables are often presented in terms of area coordinates. In the context of triangle, the areal coordinates of a point P are the ratios of the areas of PBC, PCA and PAB to the area of the reference triangle ABC. Because the ratios are a plus or a minus, it divide the plane where the triangle is to seven parts. That gives us an enlightenment to obtain a particular pattern for solving the problem of paths.

Path generation. For each triangle f_i, we project its neighborhood $\mathcal{N}_i$ onto a tangent plane based on its normal $\bar{\mathbf{n}}_\mathbf{i}$. In practice, we only project their barycenter onto that plane. Then basing the three points of f_i, we use areal coordinations to divide these barycenters to seven regions. Note that, if the projection point falls into the face f_i, we throw it away from $\mathcal{N}_i$. In this way, we give each neighbor a label that belong to a specified region. Afterwards, we sort

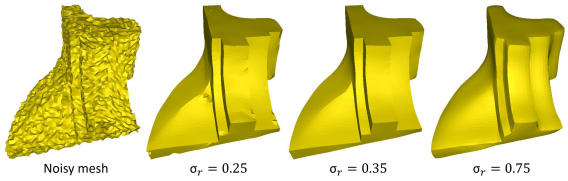

Figure 5: *Denoising results using different values of σ_r with other parameters fixed ($k_{iter} = 30$, $v_{iter} = 2$, $r = 4$).*

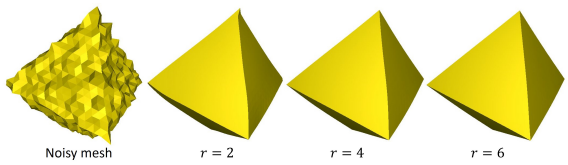

Figure 6: *Denoising results using different values of r with other parameters fixed ($k_{iter} = 75$, $v_{iter} = 2$, $\sigma_r = 0.28$). The corresponding error E_v is 0.0049, 0.0038 and 0.0030, E_n is 2.35, 1.67 and 1.00 respectively.*

these neighbors belong to the same region according to their face barycenter distance to c_i in original noisy mesh. Finally, we provide a particular pattern to assign the path for matching the ordered neighbors. This pattern is defined square-pattern, which is used for generating paths. The entire process is depicted in the figure 4.

Advantages. The above process has the power to protect local mesh features, such as edges and corners. To a certain extend, the six regions surrounding f_i depict the local structure of a mesh. Projecting along the normal of f_i makes the faces having similar normal flock together. Furthermore, the division further restricts the normal difference according to areal coordinates. The combination of these two aspects insures that one of six regions has the similar normal to f_i. The part including faces ①, ② and ③ has the similar normal to f_i in the figure 4(b). These normals play a important role in applying our intrinsic filtering algorithm.

Figure 4(c) shows the square-pattern for generating path. We use $n^2 (n = 1, 2, ...)$ as connection points, the sequence numbers between $(m-1)^2$ and m^2 including m^2 directly connect the number $(m-1)^2$. In this way, paths are generated. For example, the path between number 8 and f_i is $8-4-1-f_i$, number 16 and f_i is $16-9-4-1-f_i$ and so on. Once generating the paths, they will not change in the subsequent iterations.

In the next section, we will introduce the experimental results basing our method.

5. Implementation and results

5.1. Parameters chosen

Our method has four parameters: the number of normal filtering iterations k_{iter}, the number of iterations v_{iter} for vertex update, choosing a radius parameter r determined the geometrical neighborhood, and the standard variance parameters σ_s and σ_r, for the two difference between normals, respectively. In our experiments, we find that $k_{iter} \leq 75$ and $v_{iter} \leq 5$ are enough for achieving satisfactory results. For all results in this paper, we set the same value to σ_s and σ_r for employing the filtering unless stated otherwise. As the σ_s and σ_r control the integral of two kinds of normal difference, small value can retain details, for example weak edge, but some noise can not be removed; big value can remove noise but weaken some features (Fig.5). So we need tune their values to achieve a good experimental result. For most meshes, we choose their values between 0.1 and 0.8. In addition, we choose $r \in [2d, 6d]$ where d is the average distance between neighboring face centroid across the whole mesh. For CAD model, the two error metrics introduced in next section will decrease with increase of r seen in figure 6, so bigger r obtains more satisfactory experimental results. The reason is that some one of six regions provides more reliable normal estimation.

5.2. Results and comparisons

In the following, we compare our mesh denoising strategy with other methods basing the vertex filtering [FDCO03, JDD03], normal filtering [SRML07,ZFAT11,ZDZ*15] and L_0 minimization [HS13] on synthetic and real noisy meshes. [ZFAT11] proposes two denoising scheme: a local scheme and a global scheme. Its local method uses bilateral normal filtering strategy and its global method solves the denoising problem by minimizing a energy function. In this paper, we only compare with their local scheme.

We generate the synthetic meshes through adding noise to the vertices of a ground truth mesh along the vertex normals. The intensity of the noise is defined by a relative standard deviation $\sigma_E = \sigma / E_{mean}$, where σ is the standard deviation of the Gaussian function and E_{mean} is the average edge length of the ground truth mesh. For a fair comparison, for each method we fine tune the parameters to produce the best results in their parameter space. And furthermore, for synthetic meshes, we also evaluate two quantitative metrics about vertex-based and normal-based mesh-to-mesh error metric respectively introduced in papers [BO03] and [NH00] to analyze the difference between our results and others. These two metrics measure mean deviations about vertex and face between denoised mesh and true mesh.

The vertex-based error metric is

$$E_v = \sqrt{\frac{1}{3A(M')} \sum_{P' \in M'} A(P') dist(P', M)^2}, \tag{11}$$

where M and $M^{'}$ reference original noiseless mesh and denoised mesh respectively, $P^{'}$ is a vertex of $M^{'}$ and $dist(P^{'}, M)$ defines the distance between $P^{'}$ and a triangle of M which is closest to $P^{'}$.

The normal-based error metric is used to reveal the average angle offset which is defined as follows

$$E_n = \sum_{f' \in M', f \in M} angle(\mathbf{n}^{'}, \mathbf{n}) / N, \tag{12}$$

where $f^{'}$ and f are triangle faces of $M^{'}$ and M respectively, $\mathbf{n}^{'}$ and $\mathbf{n}$ reference the normal of $f^{'}$ and f, $angle(\mathbf{n}^{'}, \mathbf{n})$ defines the angle between $\mathbf{n}^{'}$ and $\mathbf{n}$ and the last N is the number of triangle faces of a mesh.

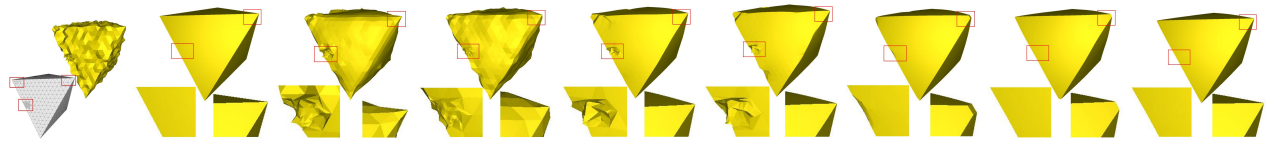

Figure 7: *Denoising a mesh with non-uniform sampling with the Gaussian noise intensity $\sigma_E = 0.3$. The results are from left to right noisy mesh, original mesh , [FDCO03], [JDD03], [SRML07], [ZFAT11], [HS13], [ZDZ*15] and ours respectively.*

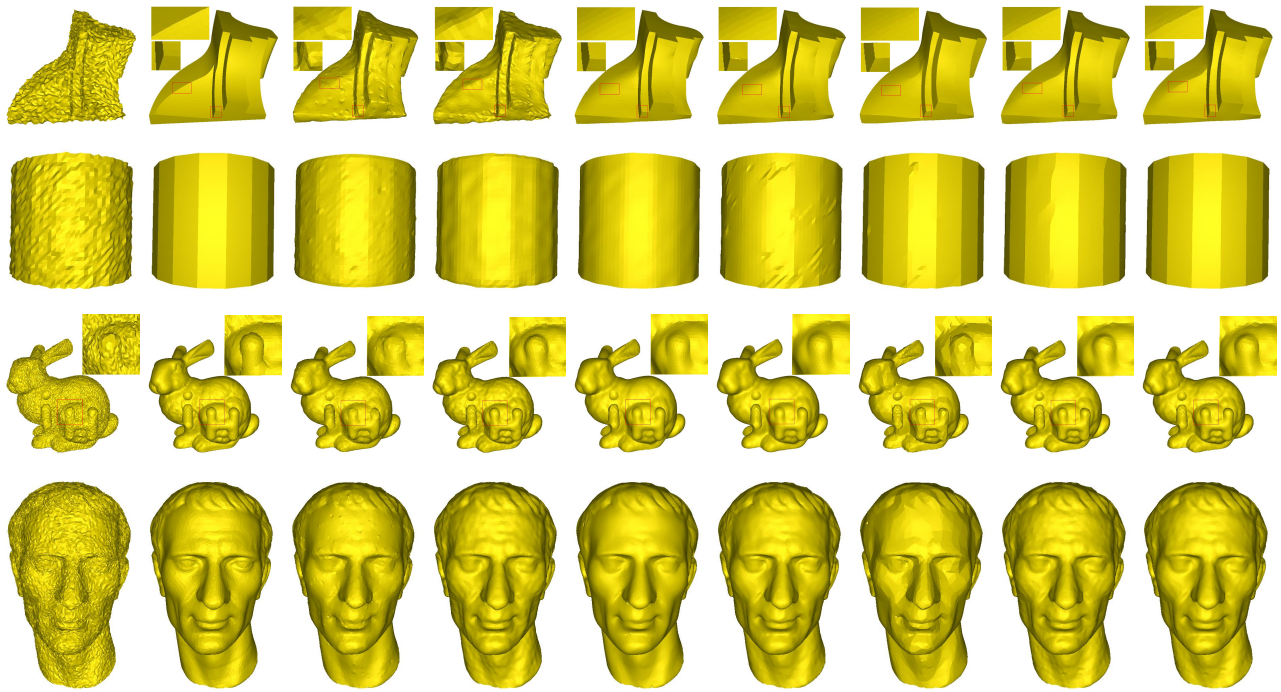

Figure 8: *Comparisons with other methods on synthetic meshes with additive Gaussian noise. The results are from left to right noisy mesh, original mesh, [FDCO03], [JDD03], [SRML07], [ZFAT11], [HS13], [ZDZ*15] and ours respectively. The intensity σ_E of the noise is from the top to bottom 0.3, 0.1, 0.2 and 0.2.*

Comparison with other methods on the synthetic meshes.

Figure 7 witnesses the effectiveness of our model on non-uniform sampling mesh. Note that we make subdivision on a strong edge and a sharp corner of the octahedron model. These regions are shown in the red rectangle boxes and one that not do subdivision is used for comparison. From the results, the approaches of [FDCO03, JDD03] can not well deal with the strong edges, because the spatial weight weaken the strength of filtering. Although [SRML07, ZFAT11, HS13] have the ability to maintain the mesh strong edge structure, they lose the effectiveness in the non-uniform sampling regions. The reason is that these region have larger normal difference than others, so these algorithms are hard to weight the two aspects. While [ZDZ*15] can well maintain the strong edge structure of mesh and also deal with non-uniform sampling problem. However, as the local structure of sharp corner is special, the guided normal of [ZDZ*15] may be prone to ambiguity. Hence, [ZDZ*15] do not well deal with the sharp corners. Our model do well in these conditions not only our weight design by two kinds of accumulative normal differences but also the function of six regions which give a better estimation in the structure of local mesh.

Our algorithm obtains satisfactory results in mesh denoising shown in the figure 8. This figure contains four models, respectively is fandisk, prism, bunny and Julius. Our method and [ZDZ*15] both have the power on preserving the mesh weak feature than others [FDCO03, JDD03, SRML07, ZFAT11, HS13] from the fandisk model, but [ZDZ*15] still hardly maintains the sharp corners in fandisk model. The prism with 18 edges is used for explain the ability in dealing with the same strength weak feature among these methods. After removing most noises, [FDCO03, JDD03, SRML07, ZFAT11] can not restore the edge structure of prism. [HS13, ZDZ*15] can recover the original structure, but they damage the straight line feature of prism. However, our method can well retains the mesh structures from the above two models. Bunny and Julius models also show us that our method can preserve the mesh detail while smoothing the noise.

Table 1 shows the quantitative errors according to the two equations 11 and 12. For most of the models, our method achieves better results than others. In general, we demonstrate that our model can compare with the state-of-the-art methods. The detail of our and their parameters can be found in the supplementary material.

Comparison with other methods on the real noisy meshes.

We also test our algorithm on the real-world 3d models with the above methods and further demonstrate the effectiveness of our model. Figure 9 illustrates the three real scanned models, angel, rabbit and iron respectively. From the eye of angel, our model depicts the eyelid well. And the iron model also reveals the ability in maintaining the edge structure.

Table 2 provides the timing of our approach for the shown examples, on a PC with an Intel Core $i7-4790K$. Even with those bigger mesh, the time of our algorithm is acceptable.

Table 2: *Time consumption for different results.*

Model	Vertices	Faces	Time(s)	k_{iter}
Octahedron	1221	2438	2.75	75
Fandisk	6475	12946	2.77	30
Prism	3914	7824	2.55	50
Bunny	34834	69451	4.33	4
Julius	36201	71912	3.54	4
Angel	24566	48090	2.86	4
Rabbit	37394	73679	4.18	4
Iron	85574	168285	14.67	20

5.3. Limitation and discussion

Although our model is effective for denoising most meshes, it still has some limitations. First, we control the normal difference by σ_r and σ_s, but our parameters can not adaptively change according the local characteristic of mesh, inevitably lost some details. Second, our algorithm easily generates folded triangles in some cases. The process of vertex update may results in the error, because it only depends on the orthogonality between the filtered normals and the new edges. Third, our method can not guarantee algorithm convergence from the figure 10. The main reason is that we apply local strategy to filter face normals, which not ensures the global structure of mesh. For solving this problem, we need use a global filtering strategy like [ZFAT11].

6. Conclusion

In this paper, we introduce a intrinsic signal filtering framework for denoising 2D manifold surface. Intrinsically, our method builds the relation between desired filtering signals and its neighbors, making calculate weights more reasonably. Other famous filtering algorithm, such as bilateral, geodesic and propagation filters, can be simplified by our algorithm. Furthermore, we apply our filtering framework to triangular meshes and obtain state-of-the-art performance. We also propose a simple, fast and effective method for

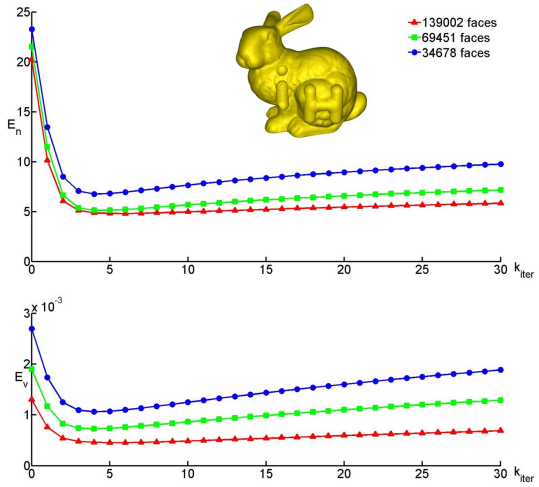

Figure 10: *Convergence plots of error metrics E_n and E_v on bunny models.(put the free noisy model)*

choosing path in the process of intrinsic filtering algorithm instead of geodesic path. And, the approach for dividing region protects the local mesh structure in a certain extent, further increasing the effectiveness of mesh filter. Finally, a large number of experiments prove the effectiveness of our method.

Acknowledgements

References

[BO03] BELYAEV A., OHTAKE Y.: A comparison of mesh smoothing methods. In *Israel-Korea Bi-national conference on geometric modeling and computer graphics* (2003), vol. 2, Citeseer. 5

[BT11] BIAN Z., TONG R.: Feature-preserving mesh denoising based on vertices classification. *Computer Aided Geometric Design 28*, 1 (2011), 50–64. 2

[CC05] CHEN C.-Y., CHENG K.-Y.: A sharpness dependent filter for mesh smoothing. *Computer Aided Geometric Design 22*, 5 (2005), 376–391. 2

[CW15] CHANG J.-H. R., WANG Y.-C. F.: Propagated image filtering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), IEEE, pp. 10–18. 1, 3

[FDCO03] FLEISHMAN S., DRORI I., COHEN-OR D.: Bilateral mesh denoising. In *ACM transactions on graphics (TOG)* (2003), vol. 22, ACM, pp. 950–953. 2, 5, 6, 8

[FYP10] FAN H., YU Y., PENG Q.: Robust feature-preserving mesh denoising based on consistent subneighborhoods. *IEEE Transactions on Visualization and Computer Graphics 16*, 2 (2010), 312–324. 2

[GS09] GRAZZINI J., SOILLE P.: Edge-preserving smoothing using a similarity measure in adaptive geodesic neighbourhoods. *Pattern Recognition 42*, 10 (2009), 2306–2316. 1, 3

[HS13] HE L., SCHAEFER S.: Mesh denoising via l_0 minimization. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 64. 1, 2, 5, 6, 8

[HST10] HE K., SUN J., TANG X.: Guided image filtering. In *European conference on computer vision* (2010), Springer, pp. 1–14. 1

[JDD03] JONES T. R., DURAND F., DESBRUN M.: Non-iterative, feature-preserving mesh smoothing. In *ACM Transactions on Graphics (TOG)* (2003), vol. 22, ACM, pp. 943–949. 1, 2, 5, 6, 8

Table 1: *Quantitative comparisons using two error metrics. For each model, the best error metric value is highlighted.*

Model	Error	[FDCO03]	[JDD03]	[SRML07]	[ZFAT11]	[HS13]	[ZDZ*15]	ours
Octahedron	E_n	12.81	10.65	6.85	5.27	3.33	3.36	**2.03**
	$E_v(\times 10^{-3})$	28.02	14.70	10.71	6.28	10.81	4.90	**2.91**
Fandisk	E_n	9.73	9.82	4.04	3.35	5.53	2.62	**2.27**
	$E_v(\times 10^{-3})$	18.14	15.10	11.02	8.72	18.79	**6.29**	6.35
Prism	E_n	4.98	4.48	3.29	3.87	0.71	0.87	**0.46**
	$E_v(\times 10^{-2})$	11.89	10.46	7.45	7.9	3.42	4.09	**2.77**
Bunny	E_n	6.93	5.81	5.89	5.68	7.21	5.35	**5.11**
	$E_v(\times 10^{-4})$	18.36	9.24	9.98	9.35	10.65	7.61	**7.26**
Julius	E_n	7.70	7.63	7.01	6.21	7.98	6.38	**6.11**
	$E_v(\times 10^{-4})$	10.58	7.71	6.49	**5.53**	8.60	6.02	5.78

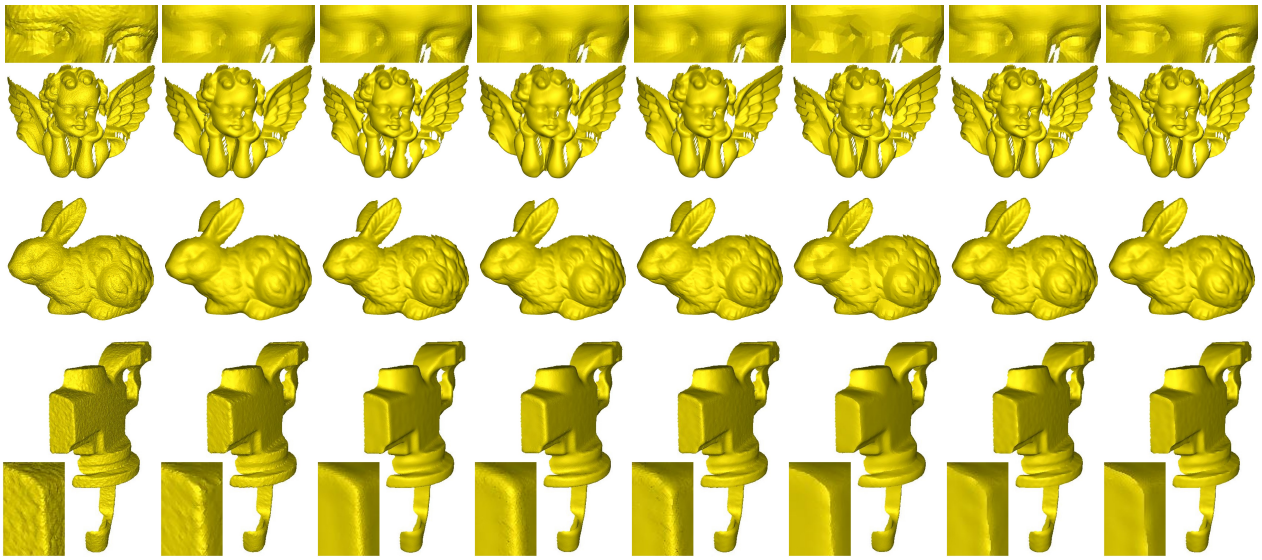

Figure 9: *The real noisy mesh. The results are from left to right noisy mesh, original mesh , [FDCO03], [JDD03], [SRML07], [ZFAT11](l), [HS13], [ZDZ*15] and ours respectively.*

[NH00] NEHORAI A., HAWKES M.: Performance bounds for estimating vector systems. *IEEE Transactions on Signal Processing 48*, 6 (2000), 1737–1749. 5

[PSA*04] PETSCHNIGG G., SZELISKI R., AGRAWALA M., COHEN M., HOPPE H., TOYAMA K.: Digital photography with flash and no-flash image pairs. *ACM transactions on graphics (TOG) 23*, 3 (2004), 664–672. 1, 2

[SCBW14] SOLOMON J., CRANE K., BUTSCHER A., WOJTAN C.: A general framework for bilateral and mean shift filtering. *CoRR abs/1405.4734 8*, 9 (2014). 1, 2

[SRML07] SUN X., ROSIN P., MARTIN R., LANGBEIN F.: Fast and effective feature-preserving mesh denoising. *IEEE transactions on visualization and computer graphics 13*, 5 (2007), 925–938. 4, 5, 6, 8

[SRML08] SUN X., ROSIN P. L., MARTIN R. R., LANGBEIN F. C.: Random walks for feature-preserving mesh denoising. *Computer Aided Geometric Design 25*, 7 (2008), 437–456. 2

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on* (1998), IEEE, pp. 839–846. 1, 2, 3

[WFL*15] WANG P.-S., FU X.-M., LIU Y., TONG X., LIU S.-L., GUO B.: Rolling guidance normal filter for geometric processing. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 173. 1, 2

[WLDB08] WANG K., LAVOUÉ G., DENIS F., BASKURT A.: A comprehensive survey on three-dimensional mesh watermarking. *IEEE Transactions on Multimedia 10*, 8 (2008), 1513–1527. 2

[WYL*14] WANG R., YANG Z., LIU L., DENG J., CHEN F.: Decoupling noise and features via weighted l_1-analysis compressed sensing. *ACM Transactions on Graphics (TOG) 33*, 2 (2014), 18. 2, 3

[WYP*15] WEI M., YU J., PANG W.-M., WANG J., QIN J., LIU L., HENG P.-A.: Bi-normal filtering for mesh denoising. *IEEE transactions on visualization and computer graphics 21*, 1 (2015), 43–55. 2

[WZY12] WANG J., ZHANG X., YU Z.: A cascaded approach for feature-preserving surface mesh denoising. *Computer-Aided Design 44*, 7 (2012), 597–610. 2

[YOB02] 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), IEEE, pp. 124–131. 2

[ZDZ*15] ZHANG W., DENG B., ZHANG J., BOUAZIZ S., LIU L.: Guided mesh normal filtering. *Computer Graphics Forum (Special Issue of Pacific Graphics 2015) 34* (2015), 23–34. 1, 2, 4, 5, 6, 8

[ZFAT11] ZHENG Y., FU H., AU O. K.-C., TAI C.-L.: Bilateral normal filtering for mesh denoising. *IEEE Transactions on Visualization and Computer Graphics 17*, 10 (2011), 1521–1530. 1, 2, 5, 6, 7, 8

[ZSXJ14] ZHANG Q., SHEN X., XU L., JIA J.: Rolling guidance filter. In *European Conference on Computer Vision* (2014), Springer, pp. 815–830. 2