Computer Aided Geometric Design 28 (2011) 50–64

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Feature-preserving mesh denoising based on vertices classification

Zhe Bian [a,b,*], Ruofeng Tong [c]

[a] Tsinghua National Laboratory for Information Science and Technology, China
[b] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[c] Institute of Artificial Intelligence, Zhejiang University, Hangzhou 310027, China

ARTICLE INFO

Article history:
Received 14 January 2009
Received in revised form 15 June 2010
Accepted 10 October 2010
Available online 16 October 2010

Keywords:
Digital process
Surface denoising
Vertex classification
Integral invariant

ABSTRACT

In this paper, we present an effective surface denoising method for noisy surfaces. The two key steps in this method involve feature vertex classification and an *iterative, two-step* denoising method depending on two feature weighting functions. The classification for feature vertices is based on volume integral invariant. With the super nature of this integral invariant, the features of vertices can be fixed with less influence of noise, and different denoising degrees can be applied to different parts of the pending surface. Compared with other methods, our approach produces better results in feature-preserving.

1. Introduction

Nowadays, 3D models are widely used in many domains, such as industry manufacturing, architecture, animation, movies and so on. The techniques for the acquisition and transmission of 3D meshes have been developed rapidly, e.g., 3D scanning (Levoy et al., 2000; Rusinkiewicz et al., 2002) and 3D simplification (Garland and Heckbert, 1997). However, the emergence of noise and uneven faces is unavoidable in those processing courses, and therefore an effective surface denoising algorithm must be achieved.

The goal of a surface denoising algorithm is to eliminate noise or spurious information on the 3D mesh while preserving original features. It is different from the fairing and smoothing algorithms: the main goal of mesh fairing is related to aesthetics, and mesh smoothing generally attempts to remove small scale details (Sun et al., 2007), while the goal of mesh denoising emphasizes more on fidelity. Hence a good denoising algorithm should be able to remove the noise on surfaces while retaining as many original features as possible.

The most common techniques are based on Laplacian (Vollmer et al., 1999; Hubeli and Gross, 2000) and Gaussian (Ohtake et al., 1997) operators, which can be regarded as band-pass filter designs. However, the common drawback of those frequency-based filters is that global sharp features are often blurred if no special care is taken. When being applied iteratively, these kinds of algorithms will not only cause the pending model to be deformed, but also shrunk. Therefore, accompanied by increasing requirements for model fidelity, considering how to preserve surface features while preventing deformity and shrinkage is crucial in surface denoising.

The key idea of our approach is to use different denoising algorithms automatically on the vertices with different features. Our approach is as such: first we classify the vertices into feature and non-feature clusters; then we adjust the face normals of the surface to update the vertex positions; and finally, we provide an available drag-back updating algorithm to eliminate the noise on the mesh. In our paper, we only consider the mesh models without holes. However, the universality of our

* Corresponding author at: Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.
E-mail addresses: bz05@mails.tsinghua.edu.cn (Z. Bian), trf@zju.edu.cn (R. Tong).

0167-8396/$ – see front matter
doi:10.1016/j.cagd.2010.10.001

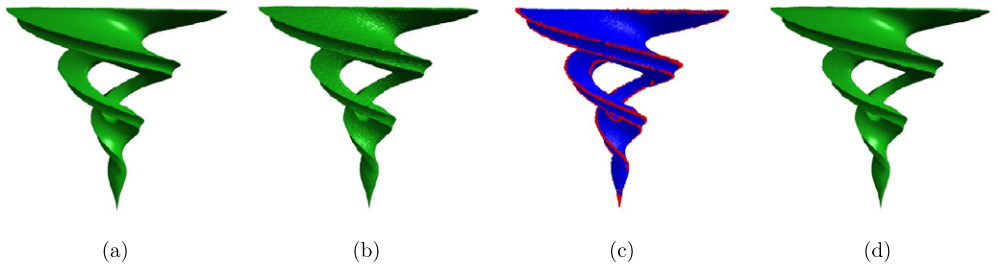


Fig. 1. Our denoising algorithm for Twirl noisy model. (a) The original model. (b) The noisy model. (c) The classification of feature and non-feature clusters on the noisy model. (d) The filtered model.

method is not lost since many repairing algorithms (Chen and Cheng, 2008) can be used to preprocess the defective models. Referred to Sun et al. (2007), our denoising approach is an *iterative, two-step* algorithm. One of our results is shown in Fig. 1.

An available way to classify mesh vertices is to do so according to their curvatures and maybe even higher order differential invariants. And differential geometry can provide powerful methods which by their very nature perform very well on smooth data. However, our approach is to denoise the surface of a noisy model, where the domain differential invariants become much less useful (Pottmann et al., 2009). Pottmann et al. (2009) introduced integral invariants to find the curvatures of vertices on meshes, which are proved to be effective in classification by Mortara et al. (2003) and Lai et al. (2007). Based on the robust characteristic of the volume integral invariant, we can classify the vertices as feature and non-feature clusters with less influence of noise. With the algorithm we proposed, obvious denoising effects can be produced on noisy meshes.

This paper gives an approach to denoising surfaces based on vertices classification. The rest of this paper is organized as follows: in the following section (Section 2) we briefly summarize our idea and various related techniques; an overview of our algorithm is given in Section 3; detailed implements are provided in Sections 4 and 5; experimental results are presented in Section 6; and conclusions and discussions of future work are given in Section 7.

2. Related work

Various surface denoising and smoothing algorithms have been proposed in recent years for noise removal, and to a large degree their different aims have been conflated: smoothing algorithms have also been suggested for noise removal. Therefore, both smoothing algorithms and mesh denoising algorithms are taken into discussion in this section. The most common techniques applied are based on the Laplacian operator (Kobbelt et al., 1998; Taubin, 1995; Desbrun et al., 1999; Vollmer et al., 1999) since this operator is simpler and faster. The method mentioned by Taubin (1995) is a signal processing on meshes, which is a simple, linear and isotropic one that does not resort to energy minimization. Desbrun et al. (1999) improved this approach to irregular meshes by using an implicit integration scheme that allows for larger time-steps and stabilizes the flow. Vollmer et al. (1999) proposed another way to improve the Laplacian denoising, which is to apply a HC-algorithm that pushes vertices back along the directions of the original locations. However, these techniques are all isotropic, and therefore apt at smoothening the noise and salient features indistinguishably for the insufficient natural instincts of the Laplacian operator, e.g., CAD models with sharp edges will become rounded after being smoothed. Many of these earlier surface fairing algorithms are based on the minimization of certain energy functionals, such as membrane energy and thin plate energy (Kobbelt et al., 1998), but to minimize these continuous functionals is computationally complicated.

A great number of feature-preserving denoising algorithms have been proposed earlier in the literature (Ohtake et al., 1997; Desbrun et al., 2000; Meyer et al., 2002; Bajaj and Xu, 2003; Hildebrandt and Polthier, 2004), which modify the diffusion equation to make it nonlinear or anisotropic. Most of these are inspired by image processing work on scalespace and anisotropic diffusion (like Desbrun et al., 2000). Although the results have much higher quality, Jones et al. (2003) pointed out that these methods rely on shock formation to preserve details, which affects the numerical conditions of the diffusion equations. The method mentioned by Jones et al. (2003) is a robust estimation smoothing algorithm that is non-iterative and effective. Nevertheless, it is still slow since it treats normal and vertex updating as a global problem of meshes.

Some special denoising algorithms have been introduced in recent years, which are applied to classify the feature and non-feature vertices on noisy meshes (Chen and Cheng, 2005; Huang and Ascher, 2008). Chen and Cheng (2005) used Bayesian discriminant analysis to determine filter methodologies for separating potential feature and non-feature vertices in curvature space, and then combined normal filtering and vertex position updating to achieve denoising. Huang and Ascher (2008) developed a specific clustering technique using first order height intensity values to divide vertex clusters for denoising algorithms, which serves as strong reference. Although both methods mentioned above are effective to a certain degree, the classification is only based on 1-*ring* differential curvatures which are not robust enough to overcome the influence of noise on meshes. The method of Shen et al. (2005) implemented a feature-preserving pre-smoothing to eliminate the influence of noise, then partitioned feature and non-feature regions and adopted different approaches for feature and

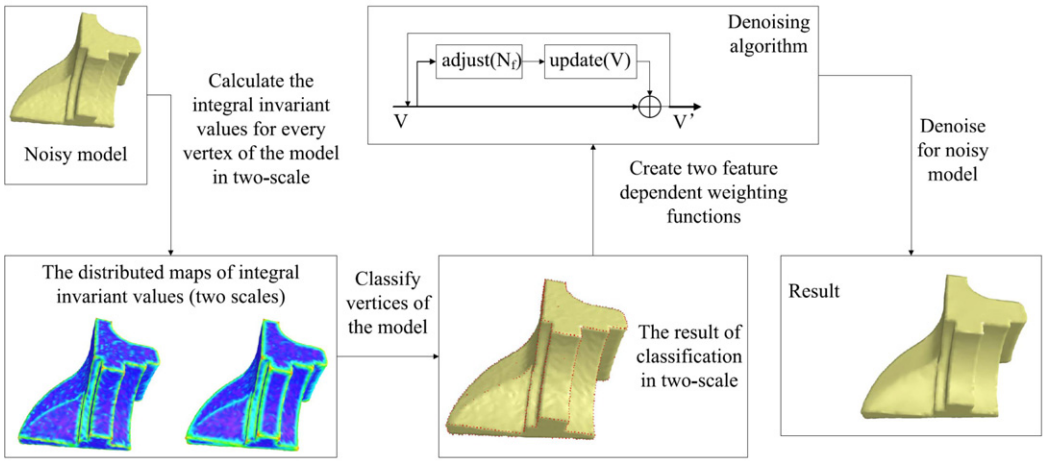


Fig. 2. Algorithm overview.

non-feature regions for denoising. However the pre-smoothing course may eliminate the detailed features without fidelity and the denoising degree is not controlled flexibly.

An *iterative, two-step* method is proposed in our paper, which evolves from methods mentioned by Chen and Cheng (2005). We classify the vertices into feature and non-feature clusters based on the volume integral invariant (Pottmann et al., 2009). This integral invariant is robust enough to find the mean curvature of every vertex with less influence of noise, and can also provide multi-scale analysis on vertices. We then adjust the face normals to update the vertex position, which is based on the classification mentioned above. Finally, a drag-back updating algorithm for vertex position is provided to eliminate noises on pending models. The vertices in the feature cluster are pushed back to their previous positions to a certain degree. The adjusting and updating courses are applied iteratively until it reaches a fine steady state.

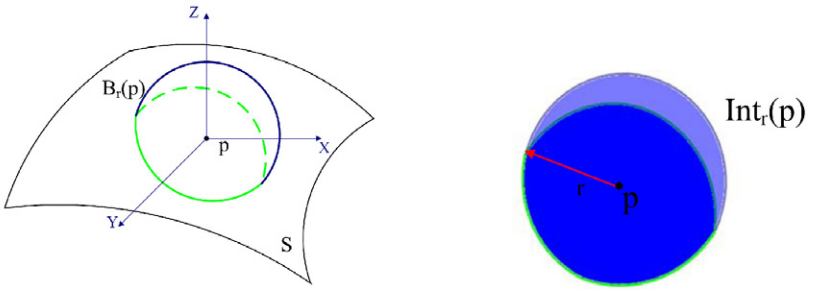
Here we focus on recent feature classifying algorithms that are most related to our work. Although various differential curvatures-based methods for classification have been applied to denoising algorithms (Desbrun et al., 2000; Chen et al., 2004; Chen and Cheng, 2005; Huang and Ascher, 2008), due to their limited 1-*ring* ranges, all of them are sensitive to the degree of noise. Based on the characteristics of intersection curves with blowing bubbles, Mortara et al. (2003) proposed a method to locally classify vertices into a few types. A similar classification method was proposed by Pottmann et al. (2009). These types of approaches are based on integral invariants obtained by integration over the local neighborhood which is constructed with the help of a ball whose radius r defines its scale. The use of integration rather than differentiation has a smoothing effect and thus indicates that this approach is robust to noises in a very effective way (Pottmann et al., 2009; Wang and Hu, 2009).

3. Overview

At the top level, our algorithm for feature-preserving denoising performs the following steps, as shown in Fig. 2.

1. *Calculation of two-scale volume integral invariant*. The vertices on the noisy model are arranged to calculate their integral invariants. The computing method is the *Fast Fourier Transform*, which is similar to the one mentioned by Pottmann et al. (2009). The integral invariants of multi-scale are more useful to understanding the features of a pending model than single-scale (Yang et al., 2006). Here two scales are enough: the small scale is used to find detailed features, while the large scale is used to find global features.
2. *k-Means clustering for vertices on the pending model.* In this step, we classify the vertices into the feature and non-feature clusters by their volume integral invariants. The classification algorithm is *k-means clustering*, which is fast when fixing the initial centers (Duda et al., 2001). The boundary of feature and non-feature clusters depending on the small-scale integral invariants is used to adjust the face normals, and the vertices in the feature cluster depending on both scales will be partly pushed back into their original positions in the course of updating.
3. *Adjustment of face normals*. In this step, we adjust face normals to update vertex positions using a method similar to that adopted by Chen and Cheng (2005). However, we use a more effective weighting function.
4. *Updating for vertex positions*. Here, we update each vertex position to complete one of the iteration steps. If a fine steady state is not reached, the algorithm will automatically go back to Step 3 in the next iteration step. The number of iteration steps is controlled by the *maximal iteration time* or the condition of the steady state, which is mentioned in Section 5.

In the following parts, we discuss the four steps in detail: Steps 1 and 2 in Section 4, Steps 3 and 4 in Section 5.



(a) The neighborhood ball of point p with radius r (b) The volume integral invariant

Fig. 3. The sketch maps of the volume integral invariant.

4. Vertex classification

As far as we know, the most crucial task of the denoising methods is to eliminate noises while retaining the fidelity of the model. Therefore, how to keep the features without the influence of noise is crucial to denoising algorithms. In this section, we introduce the *k-means clustering* based on the volume integral invariant to find the feature vertices. Although the *k-means clustering* is a global classifying method, the *k-means clustering* is good enough with its robust feature capturing ability and computational efficiency, since we only want to find the global feature vertices to retain the sharp features in the whole models. Furthermore, since the integral invariant is irrelevant to the meshes' topology, our denoising algorithm is available for uniformly and nonuniformly distributed mesh models, while many methods (like Taubin, 1995; Chen and Cheng, 2005; Huang and Ascher, 2008) are less effective to nonuniform ones.

Before giving the detailed description, we first introduce the notations. A manifold is discretized by a triangular surface mesh S. The set of its vertices is $V(S) = v_i;\ i = 1, \ldots, M$, where M is the number of vertices. The $Int_r(v_i)$ indicates the volume integral invariant of the vertex v_i used in our paper. The resolution of the model here is the average length l_{ave} of its edges.

4.1. The volume integral invariant

The concept of the integral invariant was first introduced by Manay et al. (2004) in 2D plane. They found that the integral invariants were closely related to the curvatures of curves. Furthermore, Manay et al. (2004) demonstrated the superior performance of the integral invariants on noisy data, especially for the reliable retrieval of shapes from geometric databases. Pottmann et al. (2009) extended the concept of integral invariant from R^2 to R^3. They introduced integral invariants for surfaces with the integral of a local neighborhood in 3D space (see Fig. 3(a)). In this paper we only use the volume integral invariant, which is sensitive enough to understand the shape features and is more efficiently computed than other integral invariants (Pottmann et al., 2009).

Considering the local neighborhood as a ball with radius r, centered at p, called neighborhood ball $B_r(p)$, the volume integral invariant $Int_r(p)$ is the volume of $B_r(p)$ intersected with the exterior part D of the mesh model (see Fig. 3): $Int_r(p) = \int_{D \cap B_r(p)} dx$. Clearly, the volume integral invariant will be changed along with the alteration of the concave and convex degree of a surface. Furthermore, a linear relative equation between volume integral invariant $Int_r(p)$ and the mean curvature H_p at p is presented by Pottmann et al. (2009):

$$Int_r(p) = \frac{2\pi}{3} r^3 - \frac{\pi H_p}{4} r^4 + O\left(r^5\right). \tag{1}$$

We know that a general way to classify mesh vertices is to do so according to their curvatures: the magnitude of the mean curvature is small for non-feature vertices and large for feature vertices. From Eq. (1), we can see that the volume integral invariant is sensitive enough to represent the feature distribution of the model.

We use the computational method mentioned by Pottmann et al. (2009), whose main idea is to calculate the volume integral invariant discretely. A triangle-mesh model of a sphere with radius r is used as a discretization of the spherical local neighborhood. The algorithm moves the center point of the sphere model to the vertex p. A 3D-bitmaps, where 1 means inside the $D \cap B_r(p)$ and 0 means outside the $D \cap B_r(p)$, is used to represent the space of the model. The volume integral invariant is then computed by the *fast Fourier transform* (FFT).

The volume integral invariant is preprocessed here for the classification. The values of the volume integral invariant are normalized with the volume of the neighborhood ball in a range of $[0, 1]$. A value of 0.5 means that there is no sharp feature. For expedient expression, we adjust the values of the volume integral invariant with the formula $Int'_r(v_i) = 2 * |Int_r(v_i) - 0.5|$, whose new value from 0 to 1 corresponds to the sharp feature from *non* to *most*. We still use $Int_r(v_i)$ to denote the new volume integral invariant below for convenience.

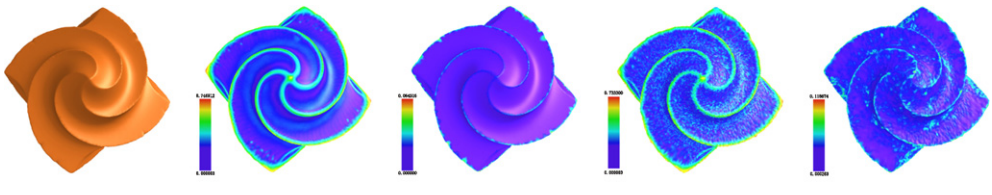


Fig. 4. The description maps of the volume integral invariant and the sharpness exponent after Chen and Cheng (2005). (a) Original octa-flower model. (b) The description map of volume integral invariant for original model. (c) The description map of sharpness exponent after Chen and Cheng (2005) for original model. (d) The description map of volume integral invariant for noisy model. (e) The description map of sharpness exponent after Chen and Cheng (2005) for noisy model. (The intensity of noisy is 40% of the mesh's resolution.)

Based on the analysis by Pottmann et al. (2009), the volume integral invariant exhibits a better behavior in shape capturing, since it can pick up more information from the neighborhood ball, and it is more robust against noises and perturbations, which are small compared to the scale r, than differential invariants used by other classification algorithms, e.g., Chen and Cheng (2005) and Huang and Ascher (2008). Since the exterior volume of the neighborhood ball, which has a large scale r compared to the noises, is approximately identical before and after noise interference, the feature degree denoted by the volume integral invariant is robust. We can observe this effect significantly in Fig. 4: the integral invariant computed via the volume neighborhood (Fig. 4(b)) picks up a little more features than the differential invariants mentioned by Chen and Cheng (2005) (Fig. 4(c)) on the original model. This effect is much more significant in the noisy model, see Figs. 4(d) and 4(e).

4.2. k-Means clustering

From the above section, we understand that the volume integral invariant can indicate the feature rising superior to noise, and thus we use this invariant as the categorical data for classification. A classic unsupervised fast clustering algorithm *k-means clustering* (Duda et al., 2001) is introduced in this paper, which allocates each data point to one of the k clusters to minimize the within-cluster sum of squares. Since the volume integral invariant can effectively detect the local features, we only need to find the global feature vertices in the pending models. And the *k-means clustering* is good enough with its computational efficiency.

Taking the categorical data set $Int_r(V(S))$ as an example, we give the minimization with a distance norm:

$$\min\left(\sum_{j=1}^{k}\sum_{v_i\in C_r^{Int}(j)}\left\|Int_r(v_i)-\mu_{C_r^{Int}(j)}\right\|\right), \tag{2}$$

where $C_r^{Int}(j)$ is a set of data points in the jth cluster and $\mu_{C_r^{Int}(j)}$ is the center of $C_r^{Int}(j)$.

The number of k in our paper is specified to 2: feature cluster $C_r^{Int}(F)$ and non-feature cluster $C_r^{Int}(NF)$. The boundary of *k-means clustering* is apt for the clusters with more elements, which leads to some non-feature vertices to be divided into the feature cluster here. In order to overcome this natural weakness, we simply renew $\mu_{C_r^{Int}(j)}$ as the median data and the clustering time is reduced correspondingly. Although it is more sensitive to noise, our robust categorical data prevents this influence.

Here, a special issue must be raised: since the classification process is automatically performed on every given 3D model, the number of clusters for an isotropic model should be set to 1, e.g., a sphere model. We only discuss the anisotropic models that are more widespread.

In a general *k-means clustering*, the random initial centers of clusters are given at the beginning of the algorithm, which needs much time for clustering. We now give a better starting point to speed up the processing time:

$$\begin{aligned}&\mu_{init}\left(C_r^{Int}(F)\right)=\max\left(Int_r(v_i),\ i=1,\ldots,M\right),\\&\mu_{init}\left(C_r^{Int}(NF)\right)=\min\left(Int_r(v_i),\ i=1,\ldots,M\right).\end{aligned} \tag{3}$$

The categorical data via the integral invariant with its scale r_1 as 3–6 times as l_{ave} are classified for the algorithm of adjusting face normals, and a two-scale classification, with scale r_1 mentioned above and scale r_2 as 7–10 times as l_{ave}, is employed to find both the local and global feature vertices for the algorithm of updating vertex positions. The feature cluster in the two-scale classification is the intersection between $C_{r_1}^{Int}(F)$ and $C_{r_2}^{Int}(F)$. The selection of scale is not absolute in the ranges above. It is also related to the sharpness of the features and the intensity of noises. Although a more precise clustering result can be obtained with more scales, it needs more computing time for integral invariant and classification, and our numerical results show that two-scale classification performs well enough. One of the clustering results is shown in Fig. 5. The boundary of the non-feature and feature clusters (denoted by *Bou*) in Fig. 5 is the mean value of $\mu_{C_r^{Int}(F)}$ and $\mu_{C_r^{Int}(NF)}$.

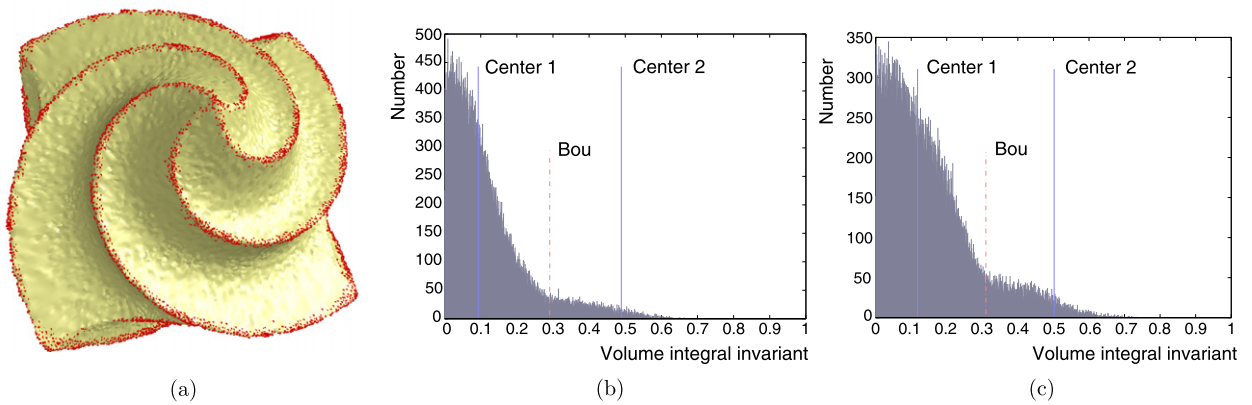


Fig. 5. The k-means clustering for octa-flower model. (a) The vertex classification (red points denote the feature cluster). (b) The statistics for the volume integral invariant with r_1 as about 4 times the model's resolution. (c) The statistics for the volume integral invariant with r_2 as about 7 times the model's resolution. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Feature-preserving mesh denoising algorithm

Our denoising algorithm is an *iterative, two-step* method which is similar to algorithms proposed by Ohtake et al. (2001) and Chen and Cheng (2005) in general structure. The procedure for this kind of algorithms is as follows:

1. Adjust the normal n_{f_i} for each polygon face f_i.
2. Update each vertex v_i according to a diffusion equation related to the adjusted face normals.

In Step 1 the mean-filter is suggested by Ohtake et al. (2001) while Chen and Cheng (2005) propose a more available method: a sharpness-dependent weighting function is employed to determine a polygon face normal that lies between the mean normal and the closest normal of its neighboring faces, which is similar to ours. In Step 2, a similar updating method is suggested by both Ohtake et al. (2001) and Chen and Cheng (2005), which use the adjusted face normals to compute the new position of every vertex, while a drag-back item is added in our method to keep global sharp features.

However, there is a problem existing in these combined normal filtering and vertex position updating methods. When the noise level is high, the model may be distorted very severely and the positions of the vertices will be chaotic, but updating vertices positions via normal filtering does not work well enough. Therefore a preprocessing course is suggested, using a one time iterative Laplacian filter to adjust the vertex positions if the noise level is very high.

In the following section we only discuss the part that concerns combining normal filtering and vertex position updating, which is the core of our method.

5.1. The denoising algorithm

Chen and Cheng (2005) presents a verdict that the mean-filter design using surface normal blurs the features in the same way that the isotropic filter design using the Laplacian operator does, and the median filter does enhance edges but fails to preserve corners. Gonzalez and Woods (2002) show that the min-filter is the only ranking filter that can preserve corner images. Taking the above analysis into account, we employ a feature dependent weighting function to determine a ranking diffusion for the face normal: the face normal is adjusted to lie between the mean normal and the closest normal of its neighboring faces.

The feature degree of the face is the volume integral invariant of its centroid, which can be interpolated by the volume integral invariants of its vertices. The feature dependent weighting function is defined based on the distribution of feature degrees. The algorithm of the face normals' diffusion is stated as follows:

1. Apply the following equation to compute the mean normal $\alpha(f_i)$ for each face f_i:

$$\alpha(f_i) = normalize\left(\sum_{f_j \in N_i^F} w_{ij}^F n_{f_j}\right), \qquad w_{ij}^F = \frac{1}{\|N_i^F\|}, \tag{4}$$

   where n_{f_j} is the normal of face f_j, N_i^F is the set of neighbor faces of f_i (defined by Chen and Cheng, 2005), and $\|N_i^F\|$ is the number of N_i^F.

2. Find the closest normalized face normal $\beta(f_i)$ in the set N_i^F for f_i:

$$\theta\big(n_{f_i}, \beta(f_i)\big) = \min\big(\theta(n_{f_i}, n_{f_j}) \mid f_j \in N_i^F\big), \tag{5}$$

   where $\theta(n_{f_i}, n_{f_j})$ is the angle between n_{f_i} and n_{f_j}.

3. Adjust the new face normal n'_{f_i} between $\alpha(f_i)$ and $\beta(f_i)$, in which the Gaussian feature dependant weighting function $G(Int_{r_1}(f_i))$ is applied:

$$n'_{f_i} = normalize\big(G\big(Int_{r_1}(f_i)\big)\alpha(f_i) + \big(1 - G\big(Int_{r_1}(f_i)\big)\big)\beta(f_i)\big), \tag{6}$$

where $Int_{r_1}(f_i)$ is the volume integral invariant of the centroid in face f_i, which is obtained by interpolating the volume integral invariants of its vertices. The $Int_{r_1}(f_i)$ is used to detect the local features in its range for the inherent quality of anti-noise.

At each iterative step, every vertex is moved to a new position and its adjacent face normals are adjusted to the new face normals at the same time. We formulate the energy required to move a vertex v_i to make the $\overrightarrow{v_i c_j}$ be perpendicular to the face normal n'_{f_j} as follows (Chen and Cheng, 2008):

$$eng(v_i) = \sum_{f_j \in N_i^v} \big(\overrightarrow{v_i c_j} \cdot n'_{f_j}\big)^2, \tag{7}$$

in which the c_j is the centroid of the adjacent face f_j, N_i^v is the set of neighbor faces of v_i (defined by Chen and Cheng, 2005) and $\overrightarrow{v_i c_j}$ is a vector from v_i to c_j. The adjustment of a vertex to a new position should achieve a steady state that is related to the new face normals above. In other words, the converging point of this sequence for updating vertex positions should be fixed. Considering the anisotropic geometric diffusion (Desbrun et al., 2000; Clarenz et al., 2000), we do not add any weight related to the adjacent faces. Furthermore, if a adjacent triangle has a large area or is a large distance away from v_i, it should have a smaller influence on the updating position and a lower weight should be given (Sun et al., 2007). For these reasons, we simply give the degree of the vertex as its weight. According to the Newton–Raphson method (Nakamura, 1992) the generation of a convergent sequence can be expressed as follows:

$$v'_i = v_i - \frac{eng(v_i)}{eng'(v_i)} = v_i + \frac{\gamma}{2\|N_i^v\|} \sum_{f_j \in N_i^v} \big(\overrightarrow{v_i c_j} \cdot n'_{f_j}\big) n'_{f_j} \tag{8}$$

where γ is the step length that is used to determine the speed for moving the vertex to a new position and to avoid flipping the polygon faces, and $\|N_i^v\|$ is the number of the neighbor faces. The default value of γ is set to 2 and if the flipping problem exists the value should be reduced until no flipping occurs.

Considering the Int_{r_1} is not precise enough to understand the features, and to perfect the effects even more, the two-scale clustering mentioned above is used to locate the feature vertices that are necessary to be dragged back into their original positions in certain degrees. The updating equation is as follows:

$$v'_i = \begin{cases} v_i + \frac{1}{\|N_i^v\|} \sum_{f_j \in N_i^v} (\overrightarrow{v_i c_j} \cdot n'_{f_j}) n'_{f_j} + (v_i^l - v_i) L(Int_{r_1}(v_i)), & v_i \text{ is feature vertex}, \\ v_i + \frac{1}{\|N_i^v\|} \sum_{f_j \in N_i^v} (\overrightarrow{v_i c_j} \cdot n'_{f_j}) n'_{f_j}, & \text{otherwise} \end{cases}$$

where v_i^l is the position before updating to v_i, and $L(Int_{r_1}(v_i))$ is the feature dependent weighting function for dragging back the feature vertices.

The denoising algorithm is executed by iterating the two steps. At the beginning the v_i^l is set to the original position. The execution is stopped if the number of iteration times reaches a preset *maximal iteration times*, or if all face normals satisfy the condition: $1 - n'_{f_i} \cdot n_{f_i} < \varepsilon$, where ε is a preset tolerance for the steady state.

5.2. Two feature dependent weighting functions

In this section, we discuss the two feature dependent weighting functions for face normal adjustment (called *FA*) and vertex position updating (called *VU*). Clearly, proper weighting functions are very important to preserve the sharp features of pending models. According to the analysis for the feature distributions of most noisy models in the literature by Chen and Cheng (2005) (e.g., Figs. 5(b) and 5(c)), we select two weighting functions: a monotonic decreasing part of Gaussian function is chosen for *FA*, which adjusts the weight to keep the noisy model's feature (see Fig. 6(a)); and a monotonic increasing function with a Laplacian function kernel is chosen for *VU*, which gives corresponding weights to feature degrees for dragging back feature vertices to their original positions (see Fig. 6(b)).

The function for *FA* is defined as: $G(x) = e^{-x^2/(2\sigma_1^2)}$. We can use σ_1 to adjust the feature sensitivity of our method. As the value of σ_1 leans toward 0, our method can detect more features on the surfaces, and at the same time the noise is kept inevitably; while the value of σ_1 leans toward $+\infty$, our method can detect global sharp features and our algorithm is apt to the mean filter.

The function for *VU* has a Laplacian function (Chen and Cheng, 2005) core, which is used to capture global features on the noisy surface: $L(x) = \lambda e^{-\sqrt{2}(x_{\max} - x)/\sigma_2}$. $x_{\max}$ is the maximum value of input variables, and λ is provided for users to adjust the entire degree of drag-back.

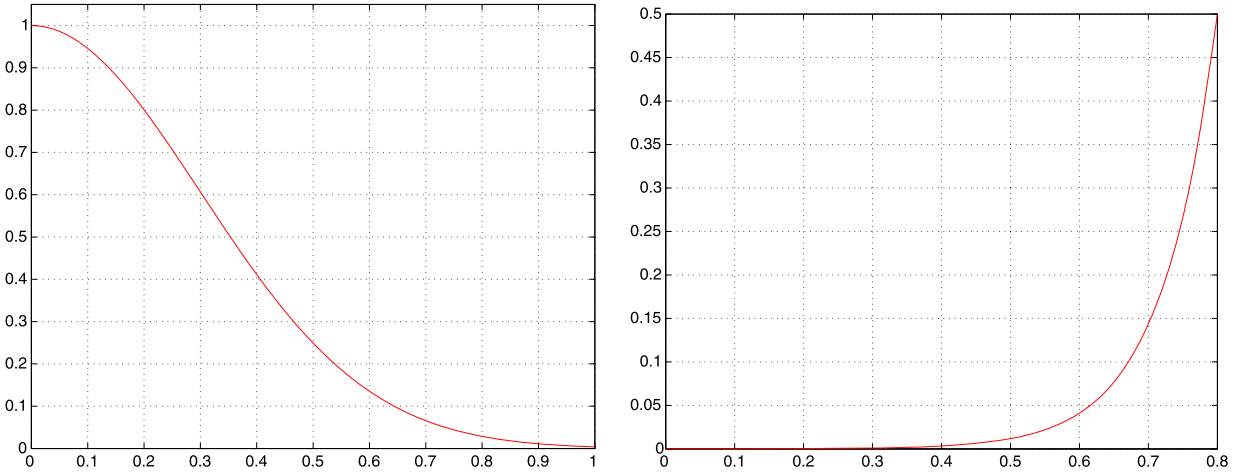


Fig. 6. The charts for two feature dependent weighting functions. Left: Gaussian feature dependent weighting function for *FA*. Right: feature dependent weighting function with Laplacian kernel for *VU*.

5.3. The choice of parameters

In this section, we briefly discuss the influence of the main parameters chosen by users in our method. These are the radiuses, r_1 and r_2, of the two-scale neighbor balls in calculating the volume integral invariant, the σ_1 in the Gaussian function for *FA*, the σ_2 and the λ in the Laplacian function for *VU*.

From Pottmann et al. (2009), we know that the influence of noise decreases with the increasing radius of the neighbor ball. So if the intensity of noise is high, we should increase the r_1 and r_2 against the noise; otherwise, we should decrease the r_1 and r_2 to find more meaningful features.

The Gaussian function is used to give different weights for face normal adjustment. If the noise is high, we have to lose some high frequency feature to eliminate the noise by increasing the σ_1; and the σ_1 should be decreased in reverse. Here, we give the reference values for σ_1. The value of σ_1 is chosen such that the weights lean toward 1 for non-feature areas, and to 0 for feature areas, so the *Bou* (see Fig. 5(b)) is suggested to fix the σ_1: when $x > Bou$, the weight $G(x)$ should be small, e.g., $G(x) = e^{-12}$ is a good choice, so $\sigma_1 = \frac{Bou}{\sqrt{24}}$.

In our method, the Laplacian function is used to control the degree of drag-back for the feature cluster. If the noise on the pending model is high, we should decrease the drag-back degree according to the decrease of the σ_2 or the λ; and if the noise is low, we should do the opposite. Since setting λ is intuitive and easy, we only give a reference value for σ_2. Following a similar ideology for fixing σ_2 as with σ_1, we set it up to make the weight approach 0 when $x = x_{\min}$, where $x_{\min}$ is the minimum value of input variables, e.g., $\sigma_2 = \frac{\sqrt{2}(x_{\max} - x_{\min})}{5}$.

Although reference values are given here, they are not absolute and since the Gaussian dependent function is related to all face normals, adjusting the σ_1 is more usual for different denoising degrees.

6. Experiment results

In this section we provide some experiments to show the characteristics of our algorithm. The noise we use in the experiments is random noise, which adds a random variation uniformly drawn from $-a$ to $+a$ to each vertex coordinate, where a is the noise amplitude. For convenience, the intensity of noise here is measured by the proportion of a to the average length l_{ave}, e.g., 0.1 means $a = 0.1 l_{ave}$, and the scale of the volume integrant invariant is defined in the same way as the noise intensity, e.g., 0.1 means $r = 0.1 l_{ave}$.

The algorithm described in this paper has been implemented in Microsoft Visual Studio 2005. Several experiments are performed on a PC with Intel Core (TM) 2 1.86 GHz CPU and 3.25 GB of RAM. Considering the aim of our method is to keep the features on mesh models, we use smooth shading to render the models.

We first visually compare our algorithm with others to observe the different filtering effects on the different intensities of noises, then employ two numerical measures to further compare the results. The model we use is the Cube with 24 578 vertices and 49 152 faces, in which it is easy to observe the shrinking and deformation of features. Its resolution is about 0.018. The intensity of noises added to this model is 0.1, 0.2, and in the extreme case, 0.4. The filters we imitate for comparison are: the Laplacian umbrella filter with a damping factor 0.6 by Taubin (1995) (shortening as L), the mean-filter proposed by Yagou et al. (2002) (shortening as M) and the sharpness dependent filter proposed by Chen and Cheng (2005) (shortening as S). In Figs. 7(h) and 7(l) the parameters of our method are set as $\sigma_1 = \frac{Bou}{\sqrt{48}}$, $\sigma_2 = \frac{\sqrt{2}(x_{\max} - x_{\min})}{5}$ and $\lambda = 0.3$, and in Fig. 7(n) are set as $\sigma_1 = \frac{Bou}{\sqrt{12}}$, $\sigma_2 = \frac{\sqrt{2}(x_{\max} - x_{\min})}{5}$ and $\lambda = 0.3$.

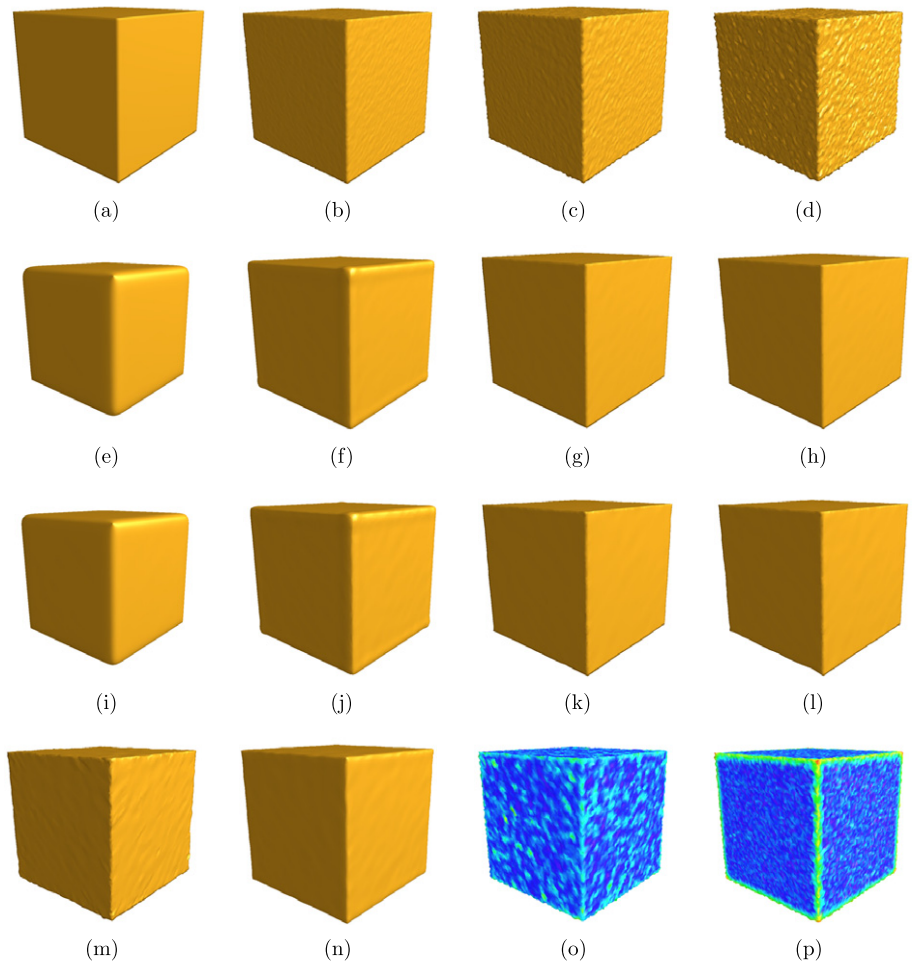


Fig. 7. The comparison for four filters (the L, M, S and ours). (a) Original cube model. (b)–(d) The models with noise 0.1, 0.2 and 0.4. (e)–(h) (b) Denoised by the four filters. (i)–(l) (c) Processed by the four filters. (m)–(n) (d) Denoised by the S and ours. (o)–(p) The description maps of the sharpness exponent after Chen and Cheng (2005) and the volume integral invariant for (d).

From Fig. 7, we can draw the following conclusions: the Laplacian filter considers the features as noise in the model, thus large features like arris of Cube shrink significantly (see Figs. 7(e) and 7(i)); the mean-filter by Yagou et al. (2002) uses surface normal diffusion that blurs the features in the same way as the isotropic filter design using the Laplacian operator (see Figs. 7(f) and 7(j)); when the intensity of noise is not very large, e.g., 0.1 and 0.2, both methods in our paper and after Chen and Cheng (2005) can yield better results in preserving the features and filtering the noises (see Figs. 7(g), 7(h), 7(k) and 7(l)); while when the intensity of noise is large, e.g., 0.4, the method after Chen and Cheng (2005) is less effective than ours since the sharpness it depends on is less robust to noise than the integral invariant (see Figs. 7(o) and 7(p)).

The above visual comparisons show that the results from our approach are better than others, and the sharpness dependent filter also yields good results when the noise level is not high. Besides the essential visual comparison, we also use the numerical measures to compare our approach with other approaches, especially the sharpness dependent filter (Chen and Cheng, 2005). Many metrics have been proposed in the literature to compare the similarity of two mesh surfaces. These include the distance metrics (Yagou et al., 2002; Kim et al., 2002; Belyaev and Ohtake, 2003), vertex-based mesh-to-mesh distance error metrics (Belyaev and Ohtake, 2003), normal error metrics (Yagou et al., 2002; Shen and Barner, 2004; Belyaev and Ohtake, 2003) and so on.

In our experiments the vertex-based distance error is introduced to denote the shrinkage degree of these algorithms (Belyaev and Ohtake, 2003). The equation of vertex-based distance error is as follows:

$$E_d = \sqrt{\frac{1}{3A'(S')} \sum_{v'_i \in S'} \sum_{f'_k \in N_i^v} A'_k dist\left(v'_i, S\right)^2}, \tag{9}$$

in which the $A'(S')$ is the sum of the areas of all triangles of smoothed surface, S', S are the original reference surfaces, A'_k is the area of triangle f'_k and the $dist(v'_i, S)$ is the L^2 distance between the vertex v'_i and a triangle of the reference

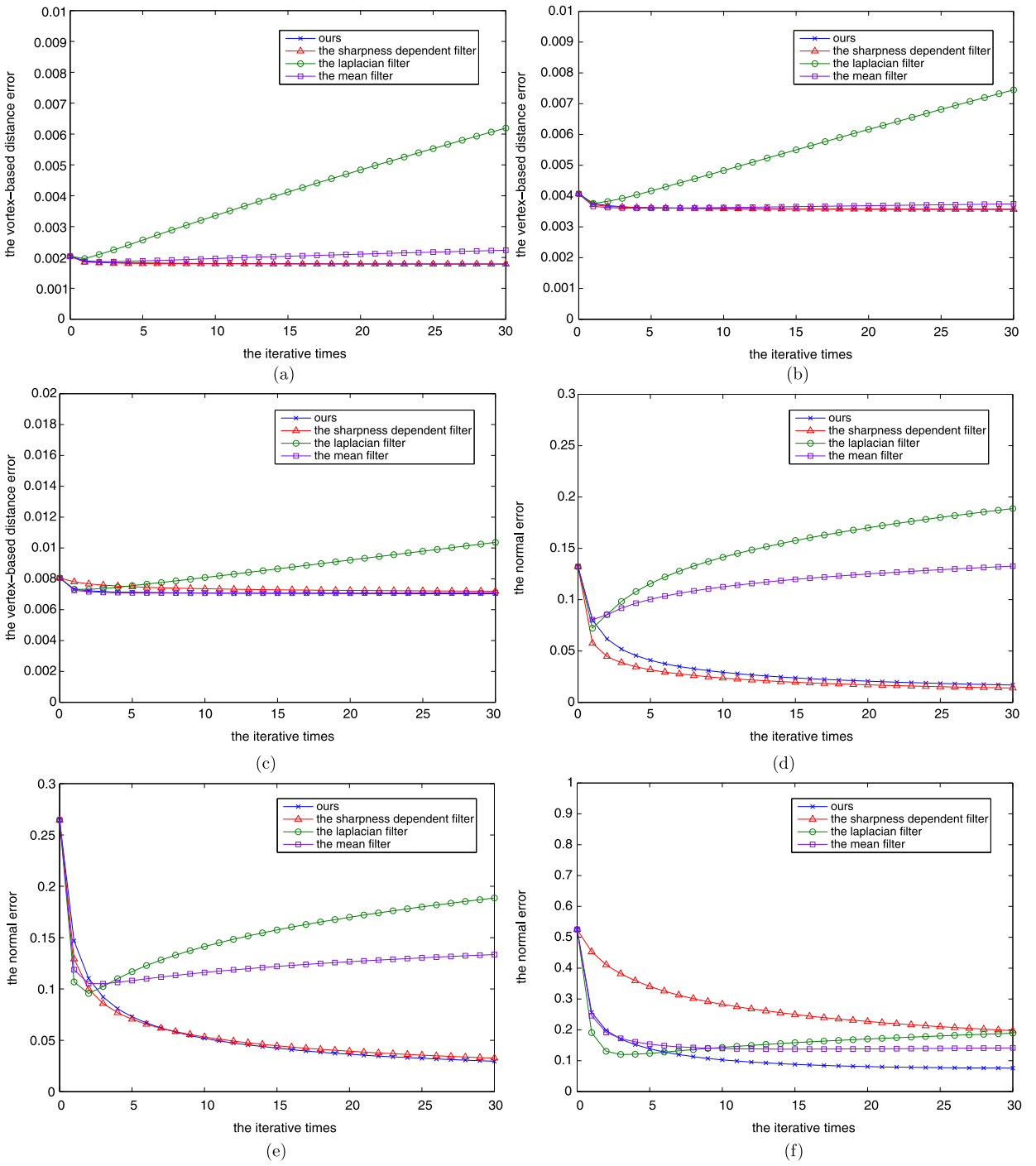


Fig. 8. The numerical comparison corresponding to Fig. 7. (a)–(c) E_d for the Cube model with noise 0.1, 0.2 and 0.4. (d)–(f) E_n for the Cube model with noise 0.1, 0.2 and 0.4.

mesh S that is closest to v_i'. And because the difference of the vertex-based distance error in the results of mean filter, the sharpness dependent filter and our approach are small (see Figs. 8(a), 8(b) and 8(c)). The normal error is also suggested in the comparison (Belyaev and Ohtake, 2003):

$$E_n = \sqrt{\frac{1}{A'(S')} \sum_{f_i' \in S'} A_i' |n_{f_i'} - n_{f_i}|^2}, \tag{10}$$

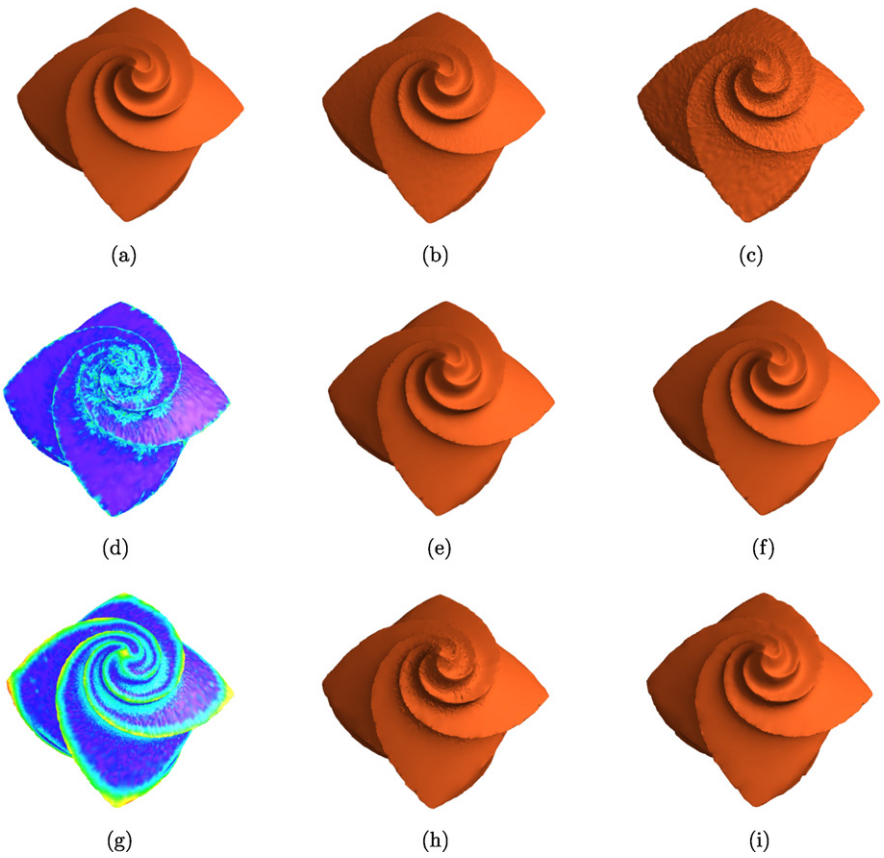


Fig. 9. The comparison for the S and ours. (a) Original octa-flower model. (b)–(c) The models with noise 0.1 and 0.4. (d) The description maps of the sharpness exponent after Chen and Cheng (2005) for (c). (e)–(f) (b) Denoised by the S and ours. (g) The description maps of the volume integral invariant for (c). (h)–(i) (c) denoised by the S and ours.

in which the f_i' and f_i are the corresponding triangle faces in surfaces S' and S. This error can denote the feature-keeping degree availably.

With the numerical comparison (see Fig. 8), we can obtain the same conclusions: the Laplacian filter shrinks considerably, see Figs. 8(a), 8(b) and 8(c); the mean filter blurs the features of the pending model, which is reflected by the growth of the normal error with an increase in the iteration times, see Figs. 8(d), 8(e) and 8(f); the sharpness dependent filter and ours both give better effects when the degree of noise is low, see Figs. 8(d) and 8(e); however, when the degree of noise is high (e.g. 0.4) the method after Chen and Cheng (2005) is worse than ours, even worse than the mean filter, see Fig. 8(f).

For further comparison between the sharpness dependent filter and ours, the subdivided octa-flower model is selected, which is a large model with 71 255 vertices and 142 506 faces. This model has both flat and curved surfaces, and sharp edges. The resolution is about 0.005132. In Fig. 9(f) the parameters of our method are set as $\sigma_1 = \frac{Bou}{\sqrt{48}}$, $\sigma_2 = \frac{\sqrt{2}(x_{\max}-x_{\min})}{5}$ and $\lambda = 0.3$, and in Fig. 9(i) are set as $\sigma_1 = \frac{Bou}{\sqrt{12}}$, $\sigma_2 = \frac{\sqrt{2}(x_{\max}-x_{\min})}{5}$ and $\lambda = 0.3$. From Figs. 9 and 10, a similar conclusion is obtained: the feature dependent filter cannot catch the features when the degree of noise is high, but ours can.

After that, we provide some results for the different parameters in our feature dependent functions. Considering the λ and σ_2 in the function of *VU* merely act on the feature vertices whose influence on these two errors is tiny, we only focus on the discussion of parameter σ_1 in the feature dependent weighting function of *FA*. Here, we use the model Fandisk with noise 0.2 for this discussion, which has 6551 vertices and 13 098 faces. The other parameters of our method are set as $\lambda = 0.3$ and $\sigma_2 = \frac{\sqrt{2}(x_{\max}-x_{\min})}{5}$. From Section 5.3, we know that if the σ_1 increases, our algorithm will be apt to the mean filter, the denoising degree will increase and inevitably the feature-keeping effects will be worse, see Fig. 11.

Following, we conduct an experiment to discuss the different denoising effects of the volume integral invariants at various scales. To accentuate the different denoising effects with various scales, we add noise 0.6 to the Balljoint model with 91 191 vertices and 181 733 faces. The resolution of Balljoint is about 0.00477. From Fig. 12, we can see that the result in Fig. 12(g) is better than the others. Although it has some differences from the original model, it is difficult to improve the denoising effect further, taking into account the large intensity of noise added. Comparing Fig. 12(c) with Fig. 12(h), we can see that there are more noises in Fig. 12(c) for the reason that the small-scale volume integral invariant cannot more effectively distinguish between the features and noises. While comparing Fig. 12(f) with Fig. 12(i), the noises are not filtered

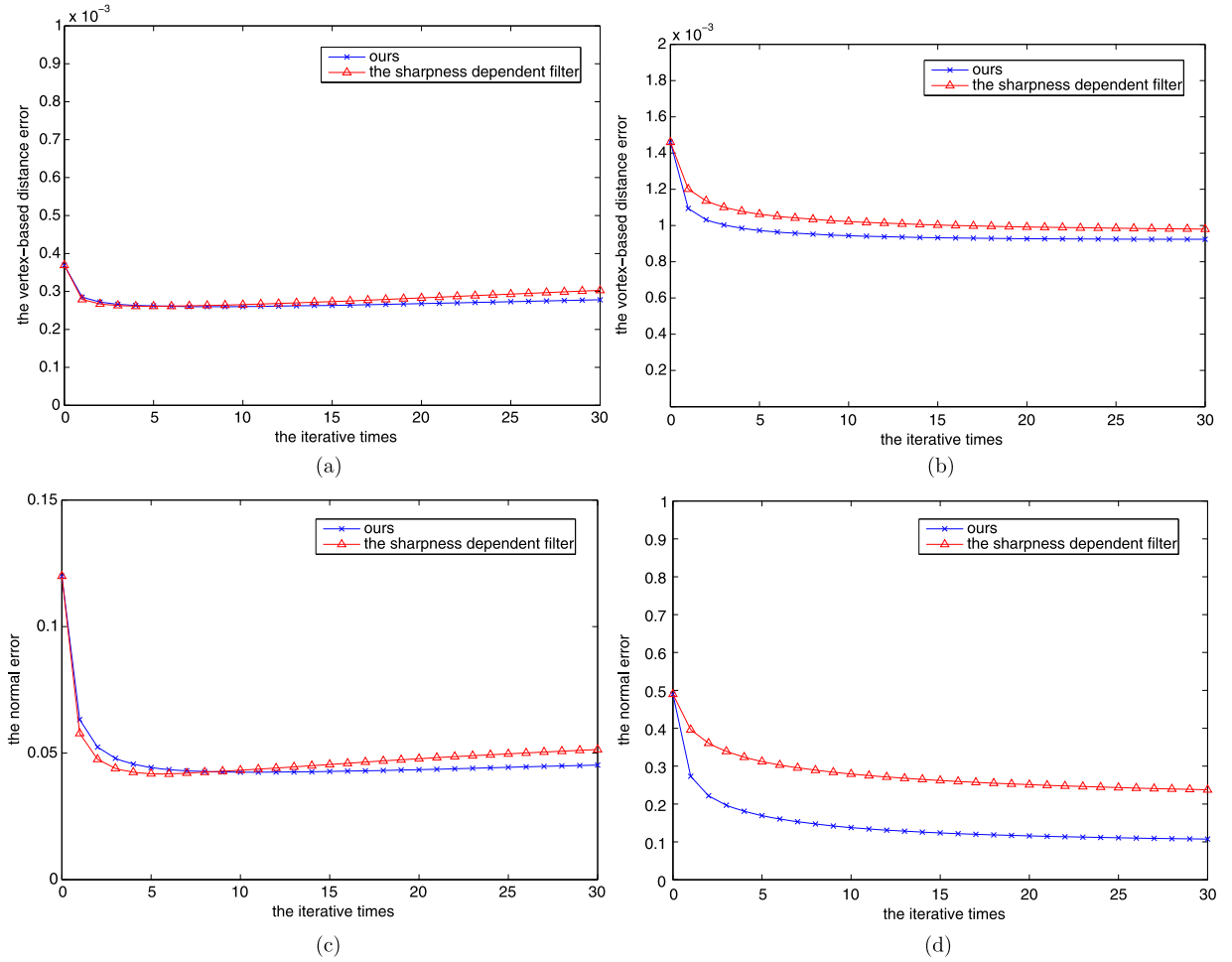


Fig. 10. The numerical comparison corresponding to Fig. 9. (a)–(b) E_d for the octa-flower model with noise 0.1 and 0.4. (c)–(d) E_n for the octa-flower model with noise 0.1 and 0.4.

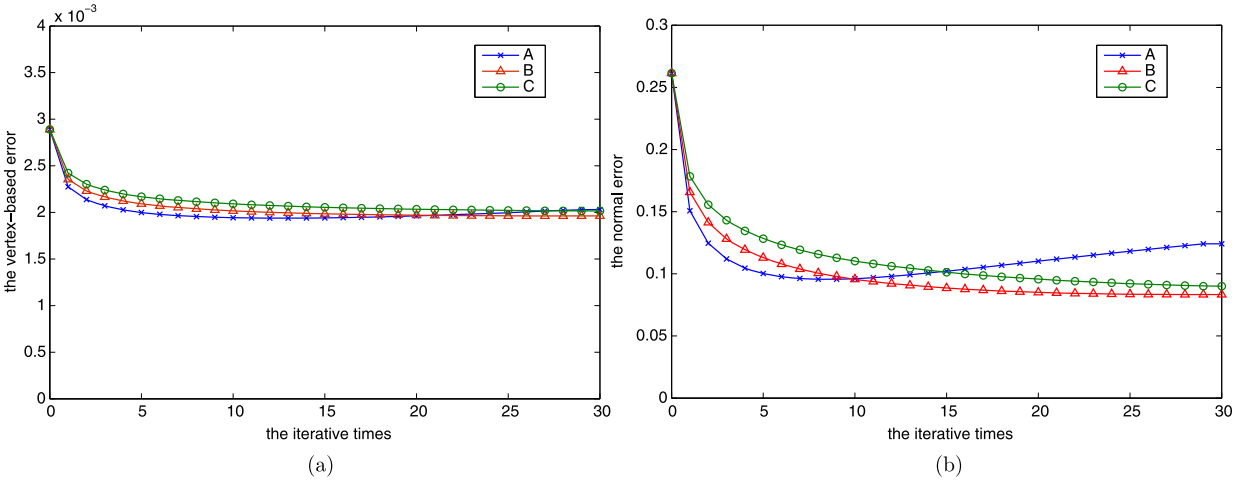
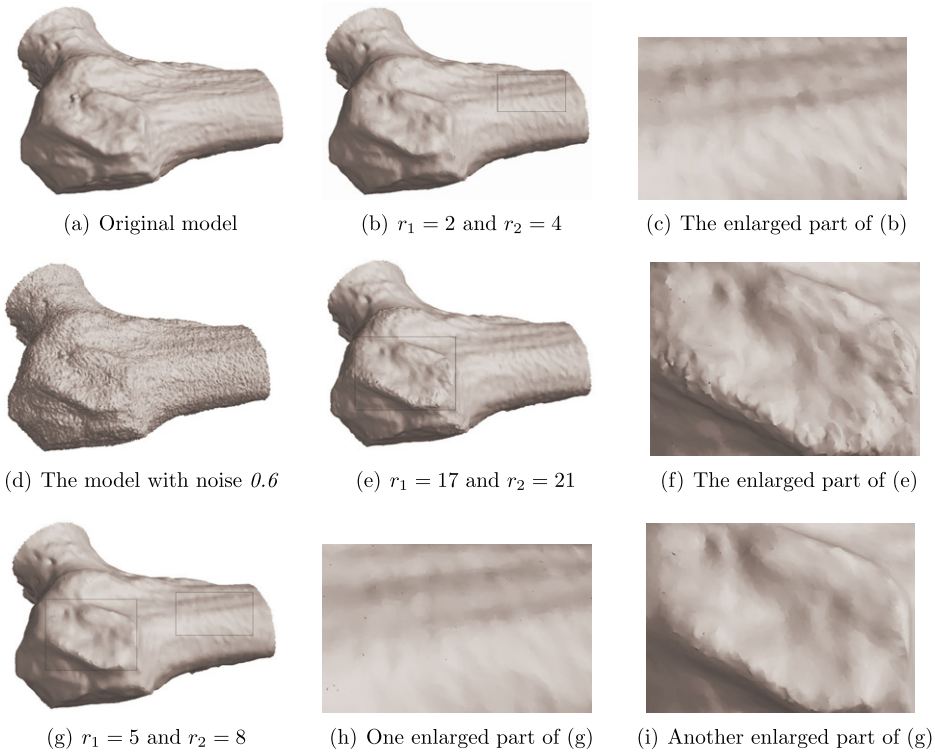


Fig. 11. The E_d and E_n of different σ_1 (the σ_1 of lines *A*, *B* and *C* are $\frac{Bou}{\sqrt{12}}$, $\frac{Bou}{\sqrt{24}}$ and $\frac{Bou}{\sqrt{48}}$) in the feature dependent function of *FA*.



(a) Original model (b) $r_1 = 2$ and $r_2 = 4$ (c) The enlarged part of (b)

(d) The model with noise *0.6* (e) $r_1 = 17$ and $r_2 = 21$ (f) The enlarged part of (e)

(g) $r_1 = 5$ and $r_2 = 8$ (h) One enlarged part of (g) (i) Another enlarged part of (g)

Fig. 12. The comparison of different scales with the Balljoint model (small, large and suitable).

well in the former since the large-scale volume integral invariant captures the oversized features, which are not the desired features on the model.

Finally we provide more demonstrations of our algorithm, see Fig. 13. The Moai and Fandisk models are frequently used to check the results of the denoising algorithms, and the Rock-arm and Asia-dragon models are the large models.

7. Conclusion and future work

In this paper, we present an available *iterative, two-step* denoising algorithm based on vertex classification. The volume integral invariant is introduced to understand the feature information on models and an improved *k*-means clustering with two-scale categorical data is used to sort out feature and non-feature vertices. Then, an *iterative, two-step* denoising algorithm based on the classification is applied to eliminate the noise on pending models. The two steps are: adjusting the face normal and updating vertex position. They both are dependent on feature weighting functions, which are used to keep models' features. Experiments show that our approach achieves better effects on uniform and nonuniform models with different intensities of noises than many other algorithms (Taubin, 1995; Yagou et al., 2002; Chen and Cheng, 2005; Huang and Ascher, 2008).

Although our algorithm is effective for feature-preserving mesh denoising, it also has some disadvantages: the calculation time of integral invariant is slow compared to the filter time; there are more parameters in our method; the suitable scales of the volume integral invariant is not very easily found for different models; the drag-back part of denoising algorithm is a little difficult to control; and if there are fewer feature vertices, the mesh shrinkage may occur in no features regions, although it is unusual. We are striving to overcome these disadvantages and propose better classification to distinguish between different kinds of features on models, such as ridges, valleys and prongs. In the future, we still need to compare more anisotropic denoising methods to examine and improve our work.

Acknowledgements

We wish to thank Susan Hu from Wellesley College for helping to improve the English. And thanks to Yong-Liang Yang for providing the code to calculate the volume integral invariant. This work was supported by the National Basic Research Project of China (Project No. 2011CB302203) and the Natural Science Foundation of China (Project Nos. U0735001, 60873126). The models we used are courtesy of the Stanford 3D Scanning Repository, the websites of Alexander G. Belyaev and the INRIA Gamma team research database.

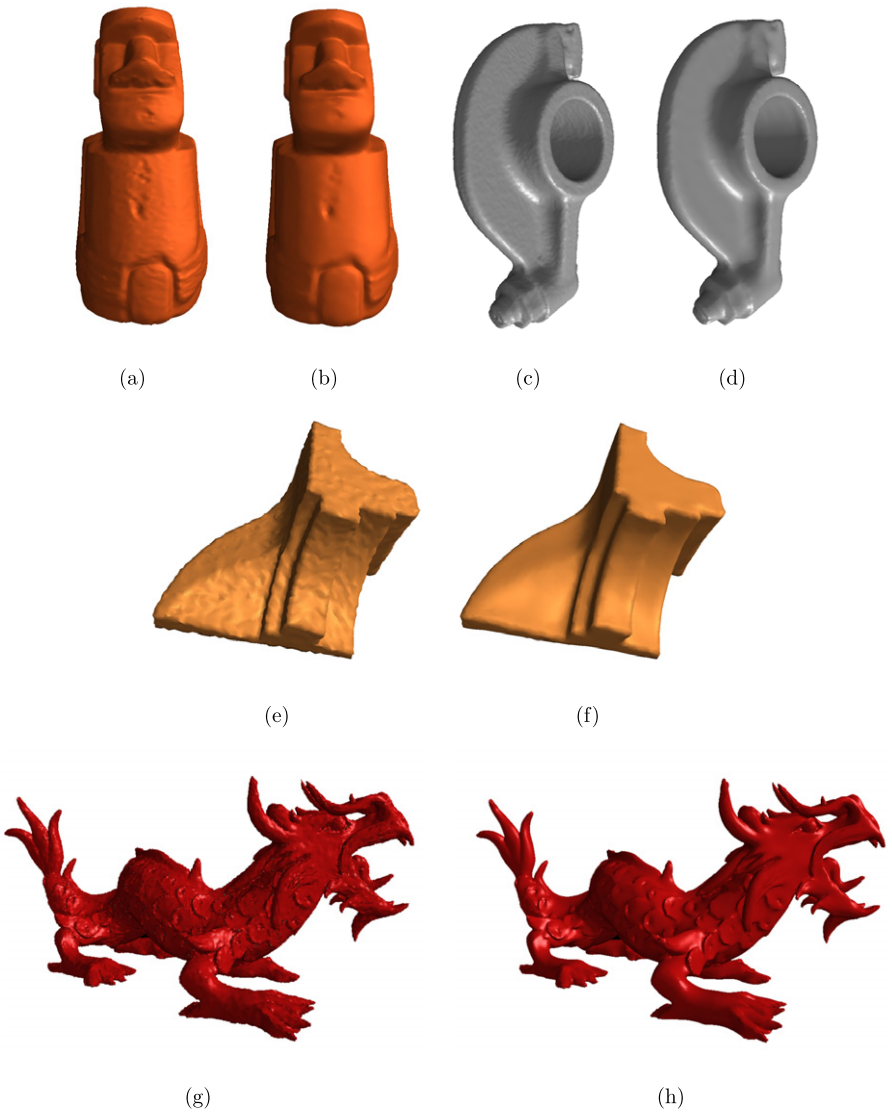


Fig. 13. The demonstrations for our algorithm. (a) Moai model with unknown noise. (b) The denoised Moai model. (c) Rock-arm model with noise 0.1. (d) The denoised Rock-arm model. (e) Fandisk model with noise 0.2. (f) The denoised Fandisk model. (g) Asia-dragon model with noise 0.4. (h) The denoised Asia-dragon model.

References

Bajaj, C.L., Xu, G., 2003. Anisotropic diffusion of surfaces and functions on surfaces. ACM Transactions on Graphics 22, 4–32.

Belyaev, A., Ohtake, Y., 2003. A comparison of mesh smoothing methods. In: Proceedings of Israel–Korea Bi-National Conference on Geometric Modeling and Computer Graphics, pp. 83–87.

Chen, C.-Y., Cheng, K.-Y., 2005. A sharpness dependent filter for mesh smoothing. Computer Aided Geometric Design 22, 376–391.

Chen, C.-Y., Cheng, K.-Y., 2008. A sharpness-dependent filter for recovering sharp features in repaired 3d mesh models. IEEE Transactions on Visualization and Computer Graphics 14, 200–212.

Chen, C.-Y., Cheng, K.-Y., Liao, H.-Y.M., 2004. Fairing of polygon meshes via Bayesian discriminant analysis. Journal of WSCG 12, 175–182.

Clarenz, U., Diewald, U., Rumpf, M., 2000. Anisotropic geometric diffusion in surface processing. In: Proceedings of IEEE Visualization, pp. 397–405.

Desbrun, M., Meyer, M., Schröder, P., Barr, A.H., 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, pp. 317–324.

Desbrun, M., Meyer, M., Schröder, P., Barr, A.H., 2000. Anisotropic feature-preserving denoising of height fields and bivariate data. In: Proceedings of Graphics Interface, pp. 145–152.

Duda, R.O., Hart, P.E., Stork, D.G., 2001. Pattern Classification, second ed. Wiley & Sons, Inc.

Garland, M., Heckbert, P.S., 1997. Surface simplification using quadricerror metrics. In: Proceedings of SIGGRAPH 1997, pp. 209–216.

Gonzalez, R.C., Woods, R.E., 2002. Digital Image Processing, second ed. Prentice-Hall, Englewood Cliffs, NJ.

Hildebrandt, K., Polthier, K., 2004. Anisotropic filtering of non-linear surface features. In: Proceedings of Eurographics 2004, pp. 391–400.

Huang, H., Ascher, U., 2008. Surface Mesh Smoothing, Regularization and Feature Detection, vol. 31. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 74–93.

Hubeli, A., Gross, M., 2000. Fairing of non-manifolds for visualization. In: Proceedings of the IEEE Visualization Conference 2000, pp. 407–414.
Jones, T.R., Durand, F., Desbrun, M., 2003. Non-iterative, feature-preserving mesh smoothing. ACM Transactions on Graphics 22, 943–949.
Kim, S.-J., Kim, S.-K., Kim, C.-H., 2002. Discrete differential error metric for surface simplification. In: Proceedings of the 10th Pacific Conference on Computer Graphics and Application, pp. 276–283.
Kobbelt, L., Campagna, S., Vorsatz, J. Seidel, H.-P., 1998. Interactive multi-resolution modeling on arbitrary meshes. In: Proceedings of SIGGRAPH 1998, vol. 32, pp. 105–114.
Lai, Y.-K., Zhou, Q.-Y., Hu, S.-M., Pottmann, H., Wallner, J., 2007. Robust feature classification and editing. IEEE Transactions on Visualization and Computer Graphics 13, 34–45.
Levoy, M., Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D., Pereira, L., Ginzton, M., Anderson, S., Davis, J., Ginsberg, J., Shade, J., Fulk, D., 2000. The digital Michelangelo project: 3d scanning of large statues. In: Proceedings of SIGGRAPH 2000, pp. 131–144.
Manay, S., Yezzi, A.J., Hong, B.W., Soatto, S., 2004. Integral invariant signatures. In: Proceedings of the European Conference on Computer Vision, pp. 87–99.
Meyer, M., Desbrun, M., Schröder, P., Barr, A.H., 2002. Discrete differential-geometry operators for triangulated 2-manifolds. In: Proceedings of Visualization and Mathematics 2002.
Mortara, M., Patané, G., Spagnuolo, M., Falcidieno, B., Rossignac, J., 2003. Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. Algorithmica 38, 227–248.
Nakamura, S., 1992. Applied Numerical Methods in C. Prentice-Hall.
Ohtake, Y., Belyaev, A., Bogaevski, I., 2001. Mesh regularization and adaptive smoothing. Computer-Aided Design 33, 789–800.
Ohtake, Y., Belyaev, A., Seidel, H.-P., 1997. Mesh smoothing by adaptive and anisotropic Gaussian filter applied to mesh normal. In: Vision Modeling and Visualization 2002, pp. 203–210.
Pottmann, H., Wallner, J., Huang, Q.-X., Yang, Y.-L., 2009. Integral invariants for robust geometry processing. Computer Aided Geometric Design 26, 37–60.
Rusinkiewicz, S., Hall-Holt, O., Levoy, M., 2002. Real-time 3d model acquisition. In: Proceedings of SIGGRAPH 2002, pp. 438–446.
Shen, J., Maxim, B., Akingbehin, K., 2005. Accurate correction of surface noises of polygonal meshes. International Journal for Numerical Methods in Engineering 64, 1678–1698.
Shen, Y., Barner, K.E., 2004. Fuzzy vector median-based surface smoothing. IEEE Transactions on Visualization and Computer Graphics 10, 252–265.
Sun, X., Rosin, P.L., Martin, R.R., Langbein, F.C., 2007. Fast and effective feature-preserving mesh denoising. IEEE Transactions on Visualization and Computer Graphics 13, 925–938.
Taubin, G., 1995. A signal processing approach to fair surface design. In: Proceedings of SIGGRAPH 1995, pp. 351–358.
Vollmer, J., Mencl, R., Müller, H., 1999. Improved Laplacian smoothing of noisy surface meshes. In: Proceedings of Eurographics 1999, pp. 131–138.
Wang, Y.-P., Hu, S.-M., 2009. A new watermarking method for 3D models based on integral invariants. IEEE Transactions on Visualization and Computer Graphics.
Yagou, H., Ohtake, Y., Belyaev, A., 2002. Mesh smoothing via mean and median filtering applied to face normals. In: Proceedings of Geometric Modeling and Processing, pp. 124–131.
Yang, Y.-L., Lai, Y.-K., Hu, S.-M., Pottmann, H., 2006. Robust principal curvatures on multiple scales. In: Proceedings of the Fourth Eurographics Symposium on Geometry Processing, pp. 223–226.