INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2005; **64**:1678–1698
Published online 23 August 2005 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme.1441

Accurate correction of surface noises of polygonal meshes

Jie Shen*,†, Bruce Maxim and Kiumi Akingbehin

Department of Computer and Information Science, University of Michigan, Dearborn, MI 48128, U.S.A.

SUMMARY

In this paper we propose a new algorithm for accurate correction of surface noises of polygonal meshes. It consists of three basic components: (a) feature-preserving pre-smoothing; (b) partitioning of feature and non-feature regions; (c) second-order predictor for non-feature regions and median filter for feature regions. The unique contributions of our approach include (a) an idea of partitioning an input surface into feature and non-feature regions so that different smoothing algorithms, which are best suited for either feature or non-feature regions can be, respectively, applied; (b) a second-order predictor that provides higher smoothing accuracy and better convergence on smoothly curved surfaces. In comparison with several existing algorithms, our algorithm is evaluated quantitatively in terms of surface normal and vertex distance error metrics. Numerical experiments indicate the effectiveness of our approach in the aspects of convergence and accuracy. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: noise reduction; polygonal mesh; structure; topology optimization; reverse engineering

1. INTRODUCTION

Accurate correction of surface noises of polygonal meshes is an important issue in engineering applications. The noises may come from non-contact optical sensors in reverse engineering [1, 2] or from isosurface extraction of finite-element-based topology optimization [3, 4] in structural optimization. In general, if 3D objects are represented by polygonal meshes, noise at each vertex may cause arbitrary perturbation along the surface normal or a movement on the underlying surface of the model. The former destroys the surface smoothness, while the latter deteriorates

*Correspondence to: Jie Shen, Department of Computer and Information Science, University of Michigan, Dearborn, MI 48128, U.S.A.
†E-mail: shen@umich.edu

Contract/grant sponsor: State of Michigan
Contract/grant sponsor: University of Michigan
Contract/grant sponsor: Altair Engineering Inc.

Received 30 October 2004
Revised 30 April 2005
Accepted 28 May 2005

the mesh quality. In this paper, we focus on how to effectively remove noised vertex perturbation in the direction of departing the underlying geometry, and call the removal of such noises as surface denoising or smoothing.

In many cases, surface fairing was used interchangeably with surface smoothing. However, in this paper we stress a subtle difference between surface fairing and denoising. The goal of the former is to achieve an aesthetic surface design of a geometric model, while the latter focuses on removing noises obtained from 3D sensing technologies or finite element simulation. In the past, many researchers [5–11] cast the surface fairing problem as an optimization problem that minimized certain functionals such as membrane energy [12], thin plate energy [12], total curvature [13–15], or sum of distances [16]. The minimization of the continuous functionals is discretized to a finite dimension, and divided difference operators are used to replace derivative operators, leading to a linear or non-linear system with respect to vertex positions.

In the area of surface denoising, one group of studies are characterized by adopting smoothing algorithms in signal processing. One brute-force algorithm is the conventional Laplacian smoothing with well-known problems of oversmoothing and volume shrinkage, which were corrected by Taubin in his volume-preserving Laplacian smoothing [17] and by Vollmer *et al*. [18]. A reweighting of the Laplacian was later introduced by him [19]. A series of studies made by Belyaev, Ohtake and Yagou led to a set of smoothing algorithms of using mean filter, median filter, Gaussian filter and their weighted counterparts [20–23]. One nice feature of their algorithms is that no dampening coefficient λ is needed from users. On the basis of subdivision, a general signal processing framework was introduced, in which denoising was one application [24]. Weiner filter was adopted by Peng *et al*. for smoothing multi-resolution meshes [25], by Pauly and Gross for point-sampled geometry [26] and by Alexa for a general mesh geometry [27].

Another group of studies were based upon the concept of second-order geometric flow with a target of reducing surface area or converging to a minimal surface. Mean-curvature flow was an early approach that was used to reduce noise [28]. It was later improved by considering anisotropic diffusion [29–32]. Ohtake *et al*. [21] used a threshold of mean curvature to terminate the oversmoothing of mean-curvature flow. As a related study, an intrinsic Laplacian of mean curvature was used for mesh fairing by decoupling the fourth-order PDE into a pair of second-order equations [33].

Recently, a number of studies have been conducted to preserve sharp features during a mesh smoothing process. One technique that was commonly used is the so-called anisotropic diffusion [29, 32, 34–37], which was originally proposed in image processing [38]. The basic idea behind this technique is to attenuate the smoothing at the sharp feature. Besides the attenuation, some researchers added feature enhancement into the smoothing process [29]. Another closely related technique is called bilateral filtering [39, 40], which was also originally proposed in image processing [41]. Its basic idea is to combine a standard Gaussian filter and a feature-preserving weighting function, a similarity function. The third technique for feature-preserving smoothing is a median filter that was well-known in image processing. One nice feature about the median filter is that it can be flawlessly implemented for polygonal meshes [23].

Even though a considerable amount of advance has been made in the recent past in the area of surface denoising or surface smoothing, there are still two important issues that have not been sufficiently addressed as listed below.

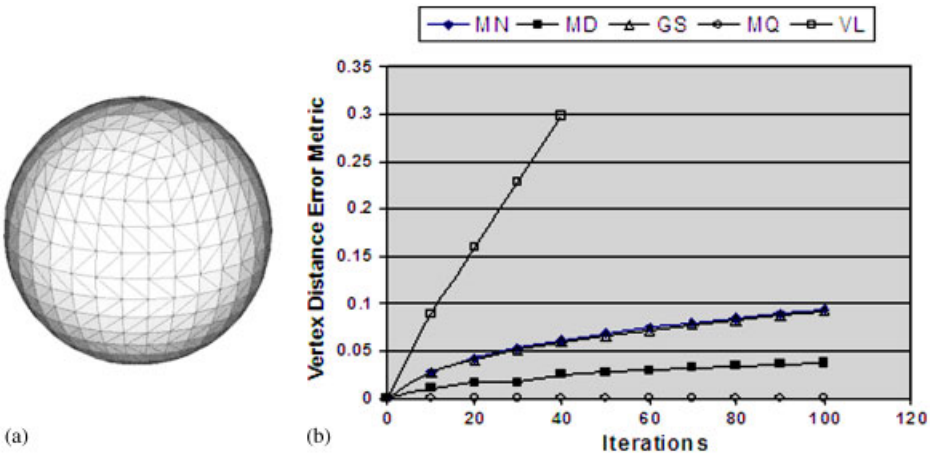

Figure 1. Smoothing of a perfectly smooth sphere: (a) original model; and (b) convergence test.

1.1. Convergence

For any iterative smoothing method, convergence is an important issue to consider. If an input mesh model contains a reasonably uniform level of noises, the smoothing process of all existing algorithms proceeds nicely as expected. However, one ambiguous problem is how to determine the termination point for this iterative process. Normally, a threshold is used as an indicator to terminate the smoothing process, and very little attention was paid to the smoothing behaviour beyond that termination point, which is related to the post-threshold stability of each algorithm. Such stability is an important indicator in classifying different smoothing algorithms. An algorithm without the post-threshold stability cannot be considered as having a true convergence, leading to an unstable smoothing accuracy that may vary from model to model, because it is significantly dependent upon a correct choice of the termination threshold.

When we consider smoothing algorithms, one of the best ways to test the post-threshold stability of different approaches is to apply these algorithms on a 'perfectly' smooth data model. Since we represent a data model in a discrete polygonal mesh, 'perfectly' herein means an almost perfectly smooth mesh. If an algorithm has a true convergence, the smoothing process of a perfectly smooth data model should not deviate from its initial geometric configuration, i.e. no smoothing action should take place. Figure 1 shows the smoothing process of a perfectly smooth sphere by using several existing algorithms (see Section 3). The quantitative error metrics are described in Section 3. We intently choose this simplest model without any sharp feature to demonstrate that there is an issue about convergence even with simple models. From Figure 1(b), it can be seen that there still exists a room for improving the existing algorithms in the aspect of convergence or post-threshold stability. The execution time of the algorithms with this model is given in Table I.

1.2. Accuracy

Very few papers have devoted a quantitative evaluation and comparison between different smoothing algorithms [20, 23]. Only with quantitative error metrics can we compare different

Table I. Execution time of different smoothing algorithms with simple models.

Model	# of vertices	# of triangles	Smoothing algorithms	# of smoothing step	Execution time (s)
Sphere	472	940	MN	100	0.41
			MD	100	2.71
			GS	5	3.05
			BL	100	0.07
			MC	100	0.81
			VL	100	0.03
			MQ	100	6.48
Cone	978	1952	MD	100	6.03
			GS	100	6.68
			BL	5	0.24
			VL	100	0.17
			MN	100	0.95
			MQ	100	22.06
Monkey saddle	225	392	MN	100	0.17
			MD	100	0.97
			GS	100	1.14
			BL	5	0.032
			VL	100	0.015
			MQ	100	4.02
Catenoid	216	384	MD	100	0.94
			GS	100	1.17
			BL	5	0.05
			VL	100	0.016
			MC	100	0.61
			MQ	100	3.95

Note: MN—mean filter; MD—median filter; GS—Gaussian filter; BL—bilateral filter; MC—mean-curvature flow; VL—volume-preserving Laplacian; MQ—median-quadratic filter (our scheme).

algorithms in an unambiguous way in terms of smoothing accuracy, which is important to remove the noises in many engineering applications such as reverse engineering and finite element simulation, but not to surface fairing for aesthetic design. Besides, one interesting phenomenon is that a certain group of algorithms (median filter) have a better accuracy in smoothing noised data models with sharp features (e.g. a box), while another group of algorithms (mean and Gaussian filters) give a higher accuracy on noised data models without sharp features (e.g. a sphere). One question that naturally comes down upon us is 'can we have an algorithm that performs best in both cases?'.

When we evaluate the smoothing error, there are at least two categories of metrics available [20]. The first one is related to the error in the surface normal between corresponding triangles of the denoised surface and the underlying smooth geometry. Here, the underlying smooth geometry means the true target surface after removing noises from the noised surface. Normally, it is difficult to accurately determine the underlying smooth geometry from a noised surface measured from various digital scanning equipments. On the other hand, synthetic smooth surfaces can be easily added by various types of noises and therefore used as test objects in our

evaluations. Surface normal reflects the intrinsic shape of each object, and is not changed when the object is subject to a global and isotropic scaling operation. Therefore, it is an important quantitative index from the viewpoint of geometry.

The second category of metrics is related to the error in the geometric distance between the denoised surface and the underlying smooth surface. Volume shrinking is an important factor that contributes the geometric distance error. Even though, a numerical manipulation, called volume scaling [28], can be used to reduce the geometric distance error for those algorithms that cause volume shrinking, an implicit assumption is that the noise level is uniform over the surface of an object. If this assumption is not satisfied, the volume scaling may not be used effectively to reduce the geometric distance error.

In order to address the above two issues, a new algorithm is designed in this paper. The main contributions include:

(1) our algorithm has a better convergence than existing algorithms such that it is less dependent upon a well-chosen termination threshold.
(2) our algorithm gives an overall better accuracy in smoothing data models with or without sharp features.

The rest of this paper is organized as follows. In Section 2, our new algorithm for surface denoising is introduced. Then, in Section 3, numerical experiments are reported and discussed, followed by some concluding remarks in Section 4.

2. A NEW SURFACE DENOISING ALGORITHM

For a better convergence and denoising accuracy, the following approach is proposed:

(1) Adopt a feature-preserving pre-smoothing (median filter) that does not require any threshold and implicitly retains the sharp features. Use G^1 geometric discontinuity and curvature threshold as an indicator for surface partitioning of feature and non-feature regions. *Feature regions* mean the areas in which either sharp edges or high curvatures exist, while the remaining parts are called the *non-feature regions*.
(2) Adopt a median filter for feature regions. In comparison with anisotropic diffusion algorithms, its main advantage is no need for the directions of principal curvatures, which may become invalid at singular points. It also avoids some pitfalls of bilateral filters at sharp edges.
(3) Design a second-order predictor as an accurate indicator for guiding a surface smoothing process in non-feature regions. The main benefit of the proposed second-order predictor is a better accuracy and convergence with curved surfaces than the first-order predictors, mean-curvature flow and Gaussian predictors in existing algorithms.
(4) Apply our second-order predictor in non-feature regions, while the median filter is used in feature regions. This forms a hybrid approach that performs consistently better than existing algorithms with different types of noised data models in terms of convergence and smoothing accuracy.

In an algorithmic format, our approach is represented by the following procedure, which calls different routines to be introduced in the rest of this paper.

Algorithm: *Hybrid approach of surface denoising*

(1) feature-preserving pre-smoothing of an input noised mesh
(2) surface partitioning of the resulting mesh
(3) loop over all elements in the input noised mesh

(3.1) if it is in a feature region, call median_filter() routine
(3.2) otherwise, call second_order_filter() routine
(3.3) go back to (3), and repeat in an iterative manner

2.1. Feature-preserving smoothing

Sharp features in a data model are a main challenge to all smoothing algorithms, because geometric non-smoothness at these features invalidates many analysis arsenals in calculus and differential geometry. A number of researchers adopted the concept of anisotropic diffusion that was originally proposed by Perona and Malik [38]. The basic idea of this approach is to smooth non-feature regions as usual and to avoid the smoothing of sharp features by using a very small weighting factor that is actually defined as a function. If the noise at sharp features is at a relatively small magnitude, this treatment is reasonable. Otherwise, its effectiveness becomes questionable. Some researchers [29, 31, 37] used additional treatment for feature enhancement in a smoothing process, which may lead to a possibility of overpreserving features. Overpreserving features may be useful in some situations. However, it seems not to be a question about the need for a smoothing algorithm that avoids both underpreserving and overpreserving features in the area of surface denoising.

When we extend the anisotropic diffusion from 2D images to 3D surface meshes, one major shortcoming is that this extension is only valid for simple edges, but not for some singular points. For instance, if discrete curvatures are calculated at the tip of a cone shown in Figure 2(a), the directions of both maximum and minimum principal curvatures are meaningless and misleading. Any edge sharpening treatment along these two directions (k_1, k_2) would result in an erroneous edge enhancement. In Figure 2(b), if three edges meet at one point and are not orthogonal to each other, the anisotropic diffusion will not behave correctly at this tip point, because the directions of two principal curvatures do not coincide with at least two edges.

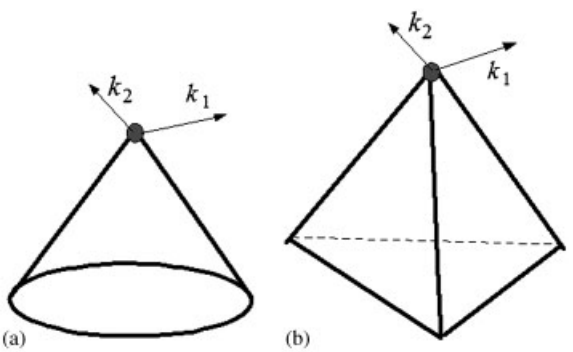

Figure 2. Two counterexamples for anisotropic diffusion at singular points in three dimension.

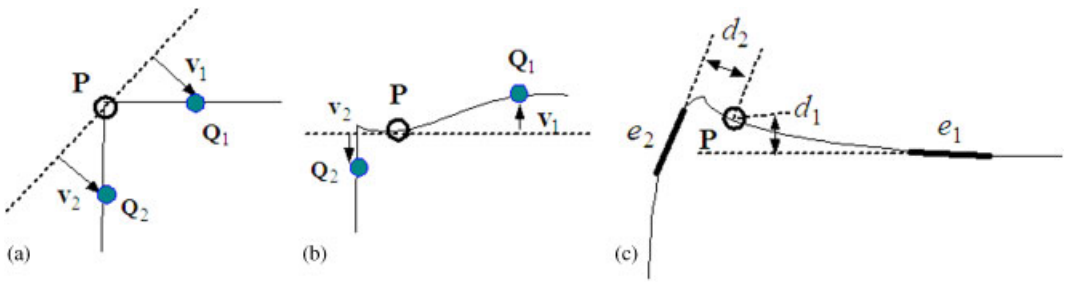

Figure 3. Three counterexamples for bilateral filtering: (a,b) Fleishman's approach; and (c) Jones' approach.

Recently, bilateral filter was used to preserve sharp features in a smoothing process [39, 40]. It is a combination of a standard Gaussian filter and a feature-preserving weighting function, a similarity function, which is another Gaussian function to penalize a large variation in the distance of neighbouring vertices to the target tangent plane [39] or in the distance between the centre vertex and neighbouring elements [40]. Both schemes work in a partially correct way at sharp features. Figure 3(a) shows that if the centre vertex is on a sharp feature, the predictor of Fleishman's approach tends to round the feature. On the other hand, if the centre vertex is at least one element away from the sharp feature, the scheme may work correctly or incorrectly depending upon the variation of neighbouring vertices. Figure 3(b) illustrates an extreme case in which vertex $\mathbf{Q}_2$ contributes a wrong vector $\mathbf{v}_2$ with respect to the vector $\mathbf{v}_1$ of vertex $\mathbf{Q}_1$ when we construct a predictor for the centre vertex $\mathbf{P}$. In contrast, Jones's approach works correctly when the centre vertex is on a feature edge. However, if the centre vertex $\mathbf{P}$ is one or very few elements away from the feature edge, his scheme does not work in an entirely correct way, because the distance d_2 may be in the same range as or even smaller than the distance d_1 in Figure 3(c), leading to a wrong contribution to the predictor of the centre vertex $\mathbf{P}$.

On the contrary, the conventional median filter can avoid the above problems if it is implemented correctly for 3D polygonal meshes. The basic concept and procedure for the median filter were described in Reference [23]. Theoretically, most discrete integrations or averaging operations at sharp features are invalid because of G^1 geometric discontinuity at these locations. The median filter avoids the averaging operations, and in the meantime provides a reasonable quality of smoothing for high-curvature non-sharp feature regions.

Our strategy in handling data models with sharp features includes the following key components:

(1) feature-preserving pre-smoothing
(2) partitioning of feature and non-feature regions
(3) median filter for feature regions

Pre-smoothing is an important step to preprocess a noised mesh model for the purpose of feature recognition or for guiding a smoothing process. If the noise level of a data model is very high, it is difficult to fulfil the feature detection without a pre-smoothing process. On the other hand, if this pre-smoothing process is not conducted carefully, some true feature information

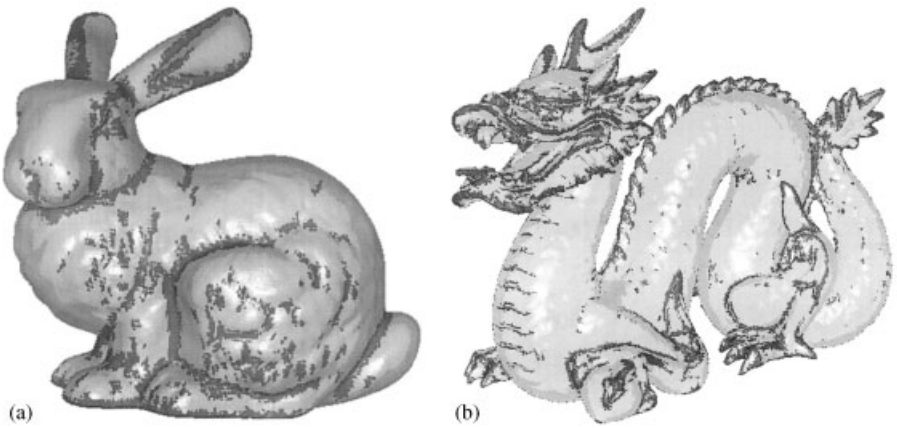

Figure 4. Partitioning of feature and non-feature regions. Data courtesy of Stanford Computer Graphics Laboratory (dark color—feature region).

may be lost before the feature detection procedure is carried out. Some researchers [29, 32] used the pre-smoothing in anisotropic diffusion, and others [40] applied it in bilateral filtering. One remaining problem is that they did not use a feature-preserving smoothing algorithm for pre-smoothing. Instead, either Gaussian filter or mean-curvature flow was used. This casts a doubt about the total validity of their approaches. More recently, Clarenz *et al*. [31] used moment analysis in identifying feature regions. The validity of the moment analysis on an initially noised surface really depends upon the noise level of the model. If a model is highly noised, the authors consider that a pre-smoothing is still a necessary step before an accurate feature detection. In this paper, we propose to use a feature-preserving smoothing algorithm, median filter, for conducting the pre-smoothing. One salient characteristic about the median filter is that it implicitly preserves sharp features, i.e. it does not rely on the feature recognition of feature regions as the anisotropic diffusion does, and it does not depend upon any threshold from users.

The partitioning of feature and non-feature regions is conducted by a breath-first search that traverses all triangles of a mesh model (Figure 4). Sharp features are defined by either a G^1 discontinuity or a high curvature. The G^1 discontinuity is identified by the angle formed by surface normal of two adjacent triangles. In this paper, if it is greater than 45°, a sharp edge is considered to occur between these two triangles. There is a vast amount of literature related to different approaches for estimating discrete curvatures [42–47]. However, no report is available for the comparison among these schemes with respect to accuracy, convergence and computational efficiency. In Reference [48], we also proposed an accurate estimation of discrete nodal curvature with a mathematically proven convergence. The salient features of our scheme include (a) proven convergence to the continuous curvature (see Proposition 2); (b) independence upon surface parameterization (a one-to-one mapping from a parameter domain to a surface). We use an index $k_{12} = 0.5(|k_1| + |k_2|)$, where k_1 and k_2 are maximum and minimum principal curvatures, respectively. When k_{12} is greater than a certain threshold, a high-curvature feature region is located.

We do not consider the surface partitioning mentioned above is a contribution of this paper. However, the application of surface partitioning in the area of surface denoising is an original idea proposed in this paper. Unlike the anisotropic diffusion, we are not concerned with the directions of principal curvatures and their pitfalls. Median filter implicitly preserves the sharp features. Yagou's median filter [23] is used in this paper for feature regions.

2.2. Second-order predictor

Here, the predictor means an indicator for a surface smoothing process, and it is a main driving force for such a process. The Laplacian smoothing is the simplest and oldest scheme, and its predictor is also the worst that frequently leads to a situation of oversmoothing. Figure 5(a) shows that the predictor of the Laplacian smoothing, $\mathbf{V}_L$, is basically a vector from a current vertex, $\mathbf{C}_v$, to a target planar face that contains all neighbouring vertices ($\mathbf{V}_0$–$\mathbf{V}_5$). Even though, a small correction coefficient, $\lambda(\ll 1.0)$, can be used to delay the incident of oversmoothing, it will happen after the iteration number becomes large enough.

Several studies have partially alleviated the problem of Laplacian smoothing, but not completely. Figures 5(b) shows the predictors of mean filter and median filter smoothing algorithms [23]. For the sake of clarity, we show the predictor in 2-dimension. In this figure, $\mathbf{C}_v\mathbf{V}_0$ and $\mathbf{C}_v\mathbf{V}_1$ represent two elements that are adjacent to the current vertex, $\mathbf{C}_v$. $\mathbf{Q}_0$ and $\mathbf{Q}_1$ are the centroid of these two elements, respectively. $\mathbf{M}_0$ and $\mathbf{M}_1$ are the weighted surface normal of these two elements, and calculated by the mean or median filtering algorithm (see Reference [23]). The smoothing predictors contributed by these two elements are, respectively, shown as $\mathbf{V}_L^0$ and $\mathbf{V}_L^1$, the vector sum of which is the overall predictor. It is quite difficult to judge whether the predictors of these two filters oversmooth or undersmooth a curved surface in Figure 5(b), because we do not have an accurate estimation about the underlying smooth geometry in these two approaches. But they are certainly much better than the Laplacian smoothing. The Gaussian smoothing is similar to the mean filter, and the only difference lies in the fact that it is a weighted mean filter with a Gaussian function as its weighting function. Some algorithms like Wiener filter do not have an explicit predictor [25]. The Wiener filtering

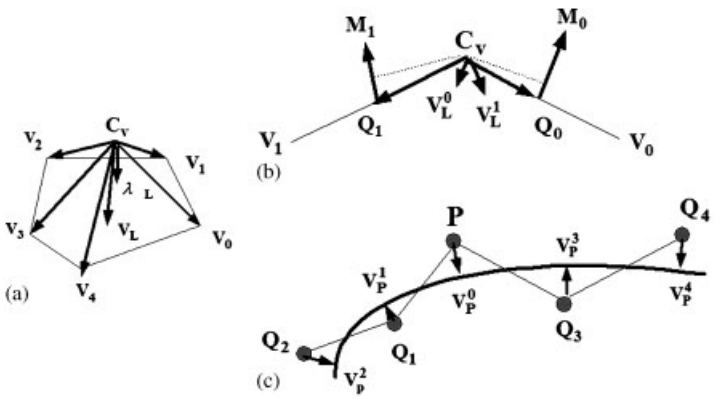

Figure 5. Predictors for different smoothing algorithms: (a) Laplacian smoothing; (b) mean and median filters; and (c) second-order.

relies on a statistical analysis to have an optimal estimate of denoised surface, and is difficult to be compared with other schemes.

A number of recent studies used the mean curvature as a predictor in the so-called mean-curvature flow, surface diffusion flow or its variants. The target surface for a mean-curvature flow should be a minimal surface on which the mean curvature is zero or a minimum surface area per volume. Typical minimal surfaces include plane, catenoid, helicoid, etc. Sphere is the case of minimum surface area per volume. Normally, it is difficult to justify the coincidence between the underlying smooth geometry of a data model and a target minimal surface either globally or locally. The only exception is to approximate a local patch by a target local plane. But this treatment still causes unavoidable smoothing error on a curved surface whose underlying smooth geometry has arbitrary curvatures.

Yet, another group of researchers used a first-order predictor (i.e. a tangent plane or its variants) as an approximation to a local surface vicinity. In other words, a tangent plane is considered as the target local plane. It shares the same advantages and disadvantages as using a linear approximation of Taylor series to a continuous function. It is fast, simple and tolerable in its accuracy as long as the surface itself is close to a planar surface or the local surface vicinity is small enough, which means a dense mesh. Regardless of this minor limitation, this predictor suffers another more fundamental problem: the difference between a true underlying surface and the planar tangent plane or its variants used in existing approaches. This problem can be illustrated by applying a recent scheme [39] onto a perfect sphere as in Figure 1(b), which indicates that the algorithm will make vertices to deviate their original positions that correspond to a perfect sphere.

We propose to use a second-order predictor as an approximation to each local surface vicinity except at the locations of sharp features. It is obvious that a second-order predictor is well suited for a smoothly curved surface patch, is a waste for a planar surface patch, and is not valid at the locations of sharp features. In Section 2.1, we introduce a feature-preserving pre-smoothing procedure to identify all the sharp feature regions. Here, we focus only on the regions without sharp features. The key components of our second-order predictor include:

(1) A 'robust' least-squares fitting procedure to fit each surface neighbourhood with a local quadric patch.
(2) A fast procedure to determine our second-order predictor.

As to the least-squares fitting procedure, 'robust' means that any outlier vertex due to G^1 discontinuity will be excluded in the least-squares fitting. By means of a feature-preserving pre-smoothing procedure, the G^1 discontinuity can be identified. For details, refer to Section 2.1. In addition, when we construct a local surface neighbourhood for the least-squares fitting, any outlier vertex that has a distance far more than a threshold should be excluded from the fitting such that individual abnormally noised points can be removed. Before conducting a least-squares fitting, the tangent plane or surface normal at each vertex needs to be determined in order to construct a local co-ordinate system by using the following procedure, which is suited for arbitrary point sets or polygonal meshes.

Routine: local_coordinate(p, M, $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$)
Pre-condition: $\mathbf{p}$ is a vertex. M contains the information of a mesh.
Post-condition: ($\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$) represents the orientation of a local co-ordinate system

(1) Determine a set of neighbouring vertices for vertex **p**.
The rule for selecting neighbouring vertices is to choose at least six different points. In this paper, we use two-ring vertices as candidates after excluding any outlier vertex due to G^1 discontinuity (see Section 2.1) or extreme distance. We also implement a neighbourhood defined by a radius specified by users.
(2) Determine a tangent plane at vertex **p**.
The principal components analysis [49] is used to determine the tangent plane at vertex **p**. The covariance matrix of the set of neighbouring vertices is

$$\mathbf{CV} = \sum_{\mathbf{q} \in \mathrm{Nbhd}(\mathbf{p})} (\mathbf{q} - \mathbf{p}) \otimes (\mathbf{q} - \mathbf{p}) \tag{1}$$

where Nbhd(**p**) is the set of neighbouring vertices at vertex **p** and $\otimes$ is outer product operator of vectors. A Jacobi transformation [50] can be used to determine eigenvectors $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ and eigenvalues $(\lambda_1 \geqslant \lambda_2 \geqslant \lambda_3)$ of the **CV**. $\mathbf{v}_3$ represents the normal direction of the tangent plane, $\mathbf{v}_1$ and $\mathbf{v}_2$ are the base vectors of orthogonal parameter co-ordinates in the tangent plane.

Note that in step 1 of local_coordinate() routine, if the number of two-ring vertices is still less than 6 as occasionally in cases where **p** is on a boundary edge or at a boundary corner, the centroid and the middle edge points of each neighbouring element are added into the set of neighbouring vertices for vertex **p**. After knowing the local co-ordinate system at vertex **p**, we are ready to conduct a least-squares fitting as follows.

Routine: least_squares_fitting(**p**, M, $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$, **X**)
Pre-condition: **p** is a vertex. M contains the information of a mesh. $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ is the local co-ordinate system.
Post-condition: **X** contains coefficients of a local quadric patch.

(1) Loop over every vertex **p** for a second-order fitting

(1.1) Determine a local quadric co-ordinate patch.
In the local co-ordinate system $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$, a quadric co-ordinate patch:

$$z = f(x, y) = a_1 x^2 + a_2 xy + a_3 y^2 + a_4 x + a_5 y + a_6 \tag{2}$$

is used to approximate the surface in the neighbourhood of vertex **p**. Note that co-ordinates (x, y) are measured along $(\mathbf{v}_1, \mathbf{v}_2)$ directions, and z co-ordinate is measured in the $\mathbf{v}_3$ direction. The linear least-squares estimation of six coefficients a_i is expressed as

$$\mathbf{BX} = \mathbf{Z} \tag{3}$$

in which

$$\mathbf{B} = \begin{bmatrix} x_1^2 & x_1 y_1 & y_1^2 & x_1 & y_1 & 1.0 \\ x_2^2 & x_2 y_2 & y_2^2 & x_2 & y_2 & 1.0 \\ \cdots & & & & & \\ x_n^2 & x_n y_n & y_n^2 & x_n & y_n & 1.0 \end{bmatrix}, \quad \mathbf{Z} = \begin{bmatrix} z_1 \\ z_2 \\ \cdots \\ z_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_6 \end{bmatrix} \tag{4}$$

where n is the number vertices of $\text{Neighbor}_{2\text{-ring}}(\mathbf{p})$. Equation (3) can be transformed to

$$\mathbf{X} = (\mathbf{B}^{\mathrm{T}}\mathbf{B})^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{Z} \tag{5}$$

which can be solved by the Cholesky decomposition of $\mathbf{B}^{\mathrm{T}}\mathbf{B}$.

If $\mathbf{B}^{\mathrm{T}}\mathbf{B}$ in Equation (5) is ill-conditioned, then the Cholesky decomposition should be replaced by a singular value decomposition [50] to solve $\mathbf{BX} = \mathbf{Z}$. Therefore, in our implementation, we use the result of the Cholesky decomposition as a conditional flag. If there is no failure in the Cholesky decomposition for the fitting of each quadric patch, the program will continue the remaining calculation in Equation (5). Otherwise, $\mathbf{BX} = \mathbf{Z}$ is solved by the singular value decomposition. However, in all of our testing models, we did not experience a single case of failure for the Cholesky decomposition of $\mathbf{B}^{\mathrm{T}}\mathbf{B}$.

The above procedures for calculating a second-order predictor are supported by the following propositions [48].

Proposition 1
If the tangent plane at point $\mathbf{p}$ is determined by using the principal component method in the routine local_coordinate(), the local quadric co-ordinate patch: $z = f(x, y) = a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6$ sufficiently represents all major types of surfaces: elliptic, hyperbolic, parabolic and planar, at the neighbourhood of point $\mathbf{p}$.

Proposition 2
If the local quadric co-ordinate patch: $z = f(x, y) = a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6$ is used to approximate the surface S at the neighbourhood of point $\mathbf{p}$, and if surface S is second-order continuous at point $\mathbf{p}$, the discrete curvatures of this local patch converge to the curvatures of surface S at point $\mathbf{p}$ in a limit.

Our second-order predictor is illustrated in Figure 5(c), in which $\mathbf{P}$ is the current vertex and $\mathbf{Q}_1$–$\mathbf{Q}_4$ are its two-ring neighbouring vertices. $\mathbf{V}_{\mathrm{P}}^0$–$\mathbf{V}_{\mathrm{P}}^4$ are predictors for the second-order smoothing, and are determined by a vector from each vertex to its closest point on the local quadric patch, which can be located in a process for finding the distance from each vertex to the local quadric patch. Equation (3) can be rewritten in the following matrix form:

$$S(\mathbf{p}) = \mathbf{p}^{\mathrm{T}}\mathbf{A}\mathbf{p} + \mathbf{b}\mathbf{p} + c = 0 \tag{6}$$

where $\mathbf{p}^{\mathrm{T}} = [x \;\; y \;\; z]$ represents points on the surface $S(\mathbf{p})$.$c = a_6$ is a scalar. $\mathbf{A}$ and $\mathbf{b}$ are a coefficient 3×3 matrix and a coefficient 3×1 vector, respectively, and are in the following forms:

$$\mathbf{A} = \begin{bmatrix} a_1 & 0.5a_2 & 0 \\ 0.5a_2 & a_3 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} a_4 \\ a_5 \\ -1 \end{bmatrix} \tag{7}$$

For a given vertex $\mathbf{q}$, its closest point $\mathbf{p}$ on the surface S can be determined by the following geometric relationship:

$$\mathbf{q} - \mathbf{p} = t\nabla S(\mathbf{p}) = t(2\mathbf{A}\mathbf{p} + \mathbf{b}) \tag{8}$$

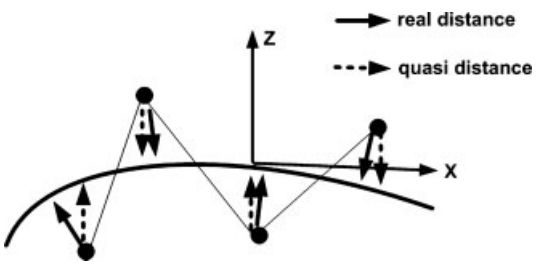

Figure 6. Quasi-distance between vertices and a local quadric surface patch.

where t is a scalar. $\nabla S(\mathbf{p})$ is the gradient of the surface, which is in the direction of surface normal and should be the same as the direction of $\mathbf{q}-\mathbf{p}$. If we apply an eigendecomposition on $\mathbf{A}$ [51], then $\mathbf{A}=\mathbf{RDR}^{\mathrm{T}}$, where $\mathbf{R}$ is an orthonormal matrix whose columns are eigenvectors of $\mathbf{A}$, and $\mathbf{D}$ is a diagonal matrix whose diagonal elements are eigenvalues of $\mathbf{A}$. The substitution of decomposed $\mathbf{A}$ into Equation (8) yields

$$\mathbf{p}=\begin{bmatrix}\mathbf{R}(\mathbf{I}+2t\mathbf{D})^{-1}\mathbf{R}^{\mathrm{T}} & 0\\ 0 & 1\end{bmatrix}(\mathbf{q}-t\mathbf{b}) \tag{9}$$

where $\mathbf{I}$ is an identity matrix. Matrices $\mathbf{R}$ and $\mathbf{D}$ have dimension 2×2. Replacing $\mathbf{p}$ in Equation (6) by the expression in Equation (9) yields a polynomial equation of degree 5 with respect to t. After a suitable root of t is found, the closest point $\mathbf{p}$ is then determined by Equation (9). However, one remaining problem is that there is no algebraic solution to a polynomial equation of degree 5, leading to an expensive computation to obtain the roots of t. Our alternative solution to this problem is to find a quasi-distance between a vertex and the local quadric surface patch, which approximates the real distance, as shown in Figure 6. The quasi-distance is basically a distance between a vertex and the local quadric patch in the local z co-ordinate direction. This approximation relies on the fact that we have a well-oriented local co-ordinate system and any outlier vertex of G^1 discontinuity has been filtered out.

After the quasi-distances are known, our second-order predictor is determined by the following procedure.

Routine: second_order_filter(M)
Pre-condition: M contains the information of a mesh, including V_{nf} (set of vertices in non-feature regions)
Post-condition: Smoothed mesh is stored in M.

(1) loop over every vertex that is not in the sharp feature regions, $\mathbf{p}\in V_{nf}$

  (1.1) call local_coordinate() routine
  (1.2) call least_squares_fitting() routine
  (1.3) loop over two-ring vertices, $\mathbf{q}\in \mathrm{Neighbor}_{2\text{-ring}}(\mathbf{p})$

    (1.3.1) calculate the predictor in Figure 5(c), $\mathbf{V}_{\mathrm{p}}^{i}$, where i corresponds to vertex $\mathbf{q}$

(1.3.2) project $\mathbf{V}_{\mathrm{p}}^{i}$ onto the surface normal $\mathbf{N}_{\mathrm{q}}$ at vertex $\mathbf{q}$

$$\hat{\mathbf{V}}_{\mathrm{p}}^{i} = (\mathbf{V}_{\mathrm{p}}^{i} \bullet \mathbf{N}_{\mathrm{q}})\mathbf{N}_{\mathrm{q}} \tag{10}$$

where $\mathbf{N}_{\mathrm{q}}$ is the surface normal determined by $\mathbf{v}_3$ in routine local_coordinate().

(1.3.3) Accumulate $\hat{\mathbf{V}}_{\mathrm{p}}^{i}$ to the overall perturbation at vertex $\mathbf{q}$

$$\mathbf{U}_{\mathrm{q}} \leftarrow \mathbf{U}_{\mathrm{q}} + \hat{\mathbf{V}}_{\mathrm{p}}^{i}, \ \ \mathrm{count}(\mathbf{q}) \leftarrow \mathrm{count}(\mathbf{q}) + 1 \tag{11}$$

where $\mathbf{U}_{\mathrm{q}}$ is the overall predictor at vertex $\mathbf{q}$. count() is an integer counter.

(2) loop over every vertex that is not in the sharp feature regions, $\mathbf{p} \in V_{nf}$

$$\mathbf{U}_{\mathrm{p}} \leftarrow \mathbf{U}_{\mathrm{p}}/\mathrm{count}(\mathbf{p}) \tag{12}$$

Note that the accumulation in Equation (11) is similar to the treatment of B-spline perturbation that is contributed by different control points. count() is used to ensure the partition of unity. The calculation of $\mathbf{V}_{\mathrm{p}}^{i}$ can be easily conducted by using the x and y co-ordinate of $\mathbf{q}$ in the local co-ordinate system to determine the z co-ordinate of the closest point on the local quadric patch: $\hat{z} = a_1 x_q^2 + a_2 x_q y_q + a_3 y_q^2 + a_4 x_q + a_5 y_q + a_6$. Consequently, $(x_q, y_q, \hat{z})$ represents the closest point of $\mathbf{q}$ in the local co-ordinate system. Finally, $\mathbf{U}_{\mathrm{p}}$ in Equation (12) is used to update the position of each vertex that is not in the sharp feature regions in the current iteration.

3. NUMERICAL EXPERIMENTS

The algorithms introduced in this paper were implemented in VC++ and tested on a HP PC with a 2.8 GHz Intel Celeron CPU. Routine median_filter () takes $O(m^2)$ of time complexity, where m is an average number of two-ring triangles. The time complexity of routine local_coordinate () is $O(m_2)$, where m_2 is an average number of two-ring vertices. Routine least_squares_fitting() is the most time-consuming module in this paper, because the Cholesky decomposition is still expensive. Its worse-case time complexity is $O(r^2 m_2)$, where $r = 6$. Routine second_order_filter () is very fast at the order of $O(m_2)$. In addition, triangle neighbour relationship needs to be set up as a pre-computation for our scheme in this paper, and it takes $O(n \log n)$ as a one-time cost, where $n = \max(N_{\mathrm{t}}, N_{\mathrm{v}})$, N_{t} and N_{v} are the numbers of triangles and vertices in a mesh model, respectively. Overall, the time complexity of our scheme is $O(nr^2 m_2 N_{\mathrm{s}})$, where N_{s} is the number of smoothing steps.

In order to evaluate the accuracy of different smoothing algorithms quantitatively, two error metrics in surface normal and vertex distance were used [20]. Another geometric distance error metric [52] could be used. The subtle difference between the two distance metrics is that the former is a measure of distances only from the surface vertices of the smoothed mesh to the base mesh, while the latter also takes account of the distances from the surface vertices of the base mesh to the smoothed mesh. Our experiments show that these two distance metrics are almost interchangeable in terms of evaluating the smoothing accuracy. Thus, in this paper we report the experiment results only in the first two error metrics.

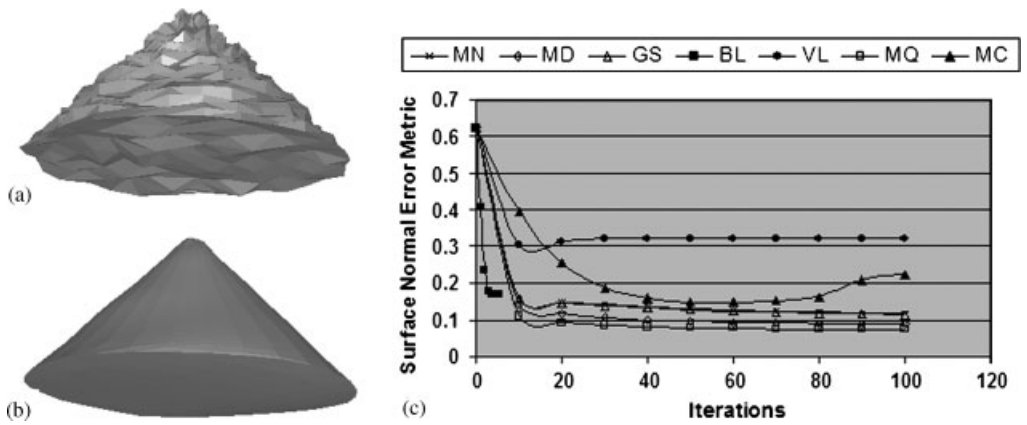

Figure 7. Smoothing of a noised cone: (a) noised; (b) smoothing result of our scheme (smoothed-100 steps); and (c) convergence test.

In this paper, surface noises are always added in the direction of surface normal at each vertex. It is reasonable to consider synthetic objects as an underlying geometry. Different types of synthetic noises are implemented, including noise generated by a random number generator, Gaussian noise, Poisson noise, and uniformly distributed noise. Due to page limitation, we report only the experimental results with noises produced by a random number generator, which generates uniformly distributed noises.

Several existing algorithms are chosen for a comparison in terms of denoising convergence and accuracy. We assign a short name for each algorithm: MN—mean filter [23]; MD—median filter [23]; GS—Gaussian filter; BL—bilateral filter [39]; MC—mean-curvature flow [28]; VL—volume-preserving Laplacian [17]; MQ—our scheme, median-quadratic filter. Most of the above algorithms do not require an input from users for a parameter except the number of smoothing steps. With the mean-curvature flow, we tried in choosing a best value of λ for each data model.

Figure 7(a) is a noised surface mesh obtained from a synthetic cone, and Figure 7(b) is the smoothing result of our scheme. The convergence test of different algorithms is shown in Figure 7(c), from which it can be seen that our scheme performs best, and then the median filter in terms of surface normal error metric that is calculated between the synthetic model and the smoothed surface mesh. Mean filter usually performs very closely with Gaussian filter such that in some of testing examples, we omit one of them. Bilateral filter [39] normally runs only five iterations so that the error metric at step 5 is what accuracy you can obtain from that algorithm. Table I shows the CPU time for different algorithms.

The boundary of a non-closed surface mesh may cause an extra problem to the convergence of smoothing algorithms. According to Figure 8(c), only our scheme shows a benign tendency of convergence. Figures 7 and 8 contain only the result of surface normal error metric, and a similar tendency was observed with the vertex distance error metric for these two models.

Since the target surface of mean-curvature flow is a minimal surface or a surface with minimum surface area per volume, it would be very interesting to observe its smoothing behaviour on a noised minimal surface. Surprisingly, the mean-curvature flow does not perform

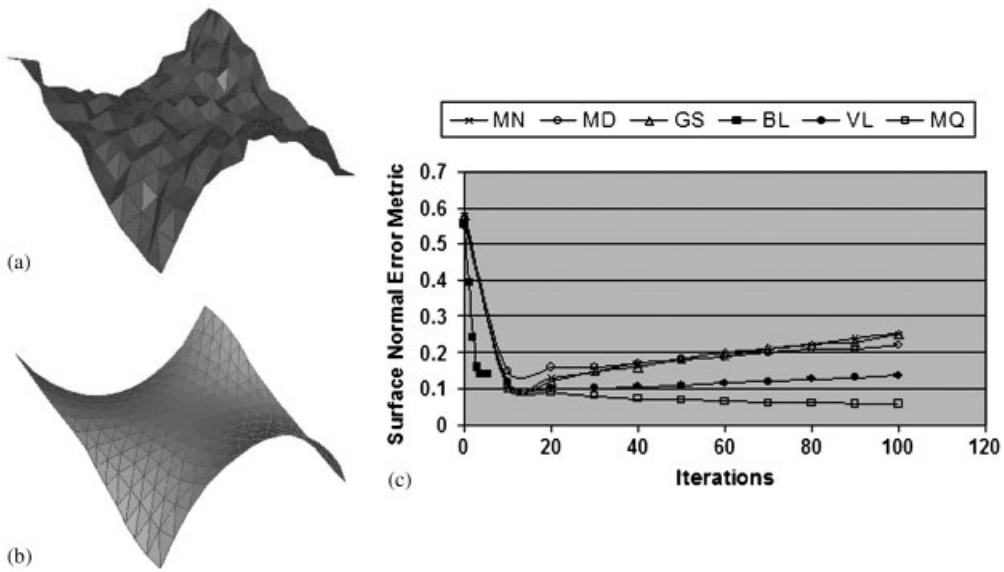

Figure 8. Smoothing of a noised monkey saddle: (a) noised; (b) smoothing result of our scheme (smoothed-100 steps); and (c) convergence test.

best on a noised catenoid model among all testing algorithms, as shown in Figure 9(c). If you look back at Figure 1(c), the same thing happens on a perfectly smooth sphere that is the case of minimum surface area per volume. Therefore, even though the mean-curvature flow is already a matured topic in mathematics and several theorems on regularity, global existence and convergence of the flow have been proved, some noticeable approximation errors must exist in the existing mean-curvature-flow smoothing algorithms. Our second-order predictor provides an relatively better convergence on second-order continuous surfaces (Figures 8 and 9) discretized by coarse meshes on which the first-order approximation error is significant. In the aspect of computational efficiency, our scheme is quite expensive, as indicated in Table I.

Figure 10(a) is a complex synthetic model (5213 vertices and 10 342 triangles) that contains a jungle of quadric surfaces: several planes, one cone, one paraboloid, three half-spheres, three cylinders and one torus. The noised model is shown in Figure 10(b), while Figures 10(c)–(f) show the smoothed model by using several testing algorithms, respectively. In terms of accuracy, our scheme performs best, while the Gaussian filter and volume-preserving Laplacian give the worst result, as indicated in Figures 10(g) and (h). Since both Gaussian and Laplacian tend to smooth out the sharp edges as the smoothing step increases, the experimental results at step 5 are presented in Figures 10(c)–(f) in favour of these two algorithms. Among all testing algorithms, our scheme, MQ, is the most time-consuming, as in Table II.

Figure 11(a) is a shape obtained from a topology optimization, while Figure 11(b) shows the smoothing result of our approach. In the cases of topology optimization, we do not know the base shape precisely such that the calculation of error metrics is not available. However, the comparison between two sub-figures in Figure 11 indicates the effectiveness of our approach in smoothing out the roughness of the surface mesh.

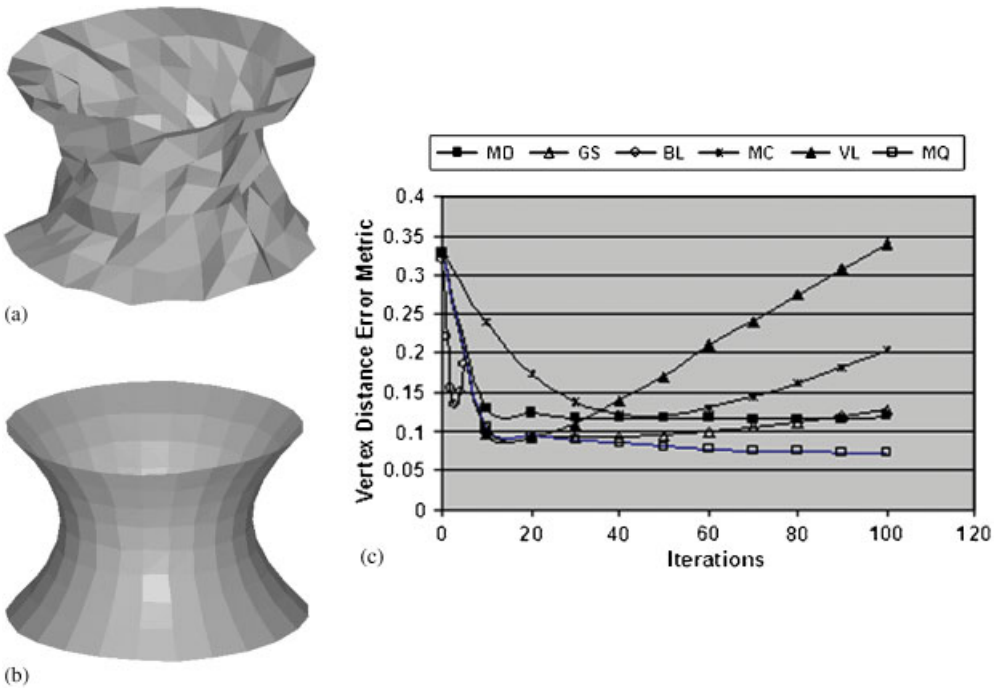

Figure 9. Smoothing of a noised catenoid: (a) noised; (b) smoothing result of our scheme (smoothed-100 steps); and (c) convergence test.

Table II. Execution time of different smoothing algorithms with various data models.

Model	# of vertices	# of triangles	Smoothing algorithms	# of smoothing step	Execution time per step (s)
Jungle of quadratic surfaces	5213	10 434	MD	100	0.33
			GS	100	0.37
			BL	5	0.6
			VL	100	0.01
			MN	100	0.06
			MQ	100	0.69

Note: MN—mean filter; MD—median filter; GS—Gaussian filter; BL—bilateral filter; MC—mean-curvature flow; VL—volume-preserving Laplacian; MQ—median-quadratic filter (our scheme).

In this paper, we focus on the convergence and accuracy in smoothing noised polygonal mesh. We believe that these two aspects are crucial for finding an accurate algorithm in the areas of reverse engineering and post-processing of topology optimization results. Although our approach is compared favourably with the existing methods in the two aspects, we still have not achieved a true convergence from a viewpoint of a minimum error. This may be partially due

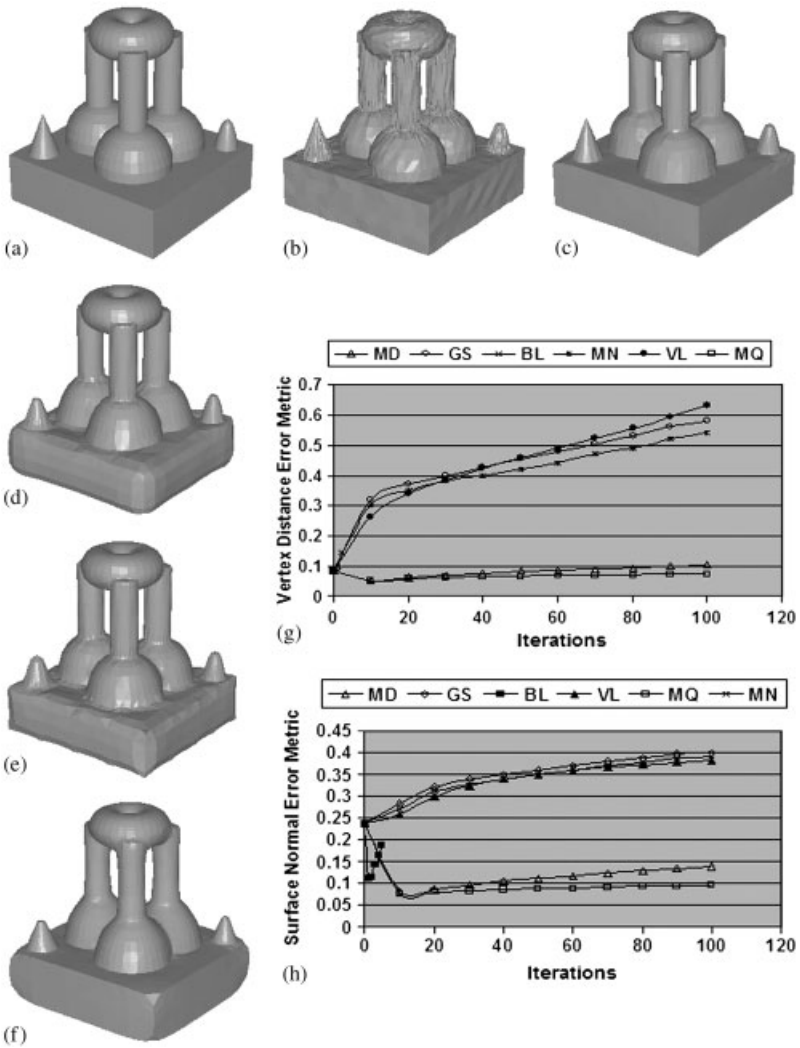

Figure 10. Surface smoothing of a jungle of quadratic surfaces: (a) original; (b) noised; (c) our (5 steps); (d) Gaussian (5 steps); (e) volume-preserving Laplacian (5 steps); (f) bilateral (5 steps); (g) surface normal error metric; and (h) vertex distance error metric.

to the intrinsic approximation nature of polygonal meshes and partially due to the imperfection of our approach in handling the randomness and uncertainty of surface noises.

The main contribution of this paper is to propose a new framework for hybrid denoising with a distinguished feature of applying different smoothing schemes, respectively, for feature and non-feature regions, which can be identified by a feature-preserving pre-smoothing and surface partitioning. The denoising accuracy and convergence of our scheme are better than all the existing approaches.

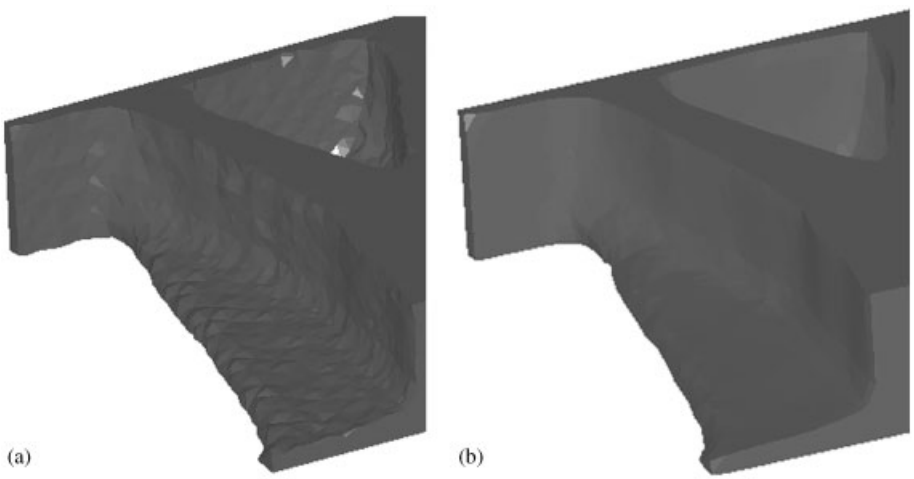

Figure 11. Smoothing of a surface mesh obtained from a topology optimization (nodes: 5516; Triangle elements: 11 030): (a) result of topology optimization; and (b) shape smoothed by our approach.

One major weakness of our approach is high computational cost in comparison with most of other smoothing algorithms. Part of future research is to improve the computational efficiency of our second-order predictor significantly and in the meantime to maintain the high smoothing accuracy and feature-preserving property.

4. CONCLUSIONS

In summary, our algorithm is the best in terms of convergence and accuracy, compared to existing smoothing algorithms. The basic idea of our scheme is to treat feature and non-feature regions, respectively, with different smoothing algorithms that are best suited. One main benefit of our approach is its consistently best performance across a wide range of different data models in terms of denoising accuracy. However, due to extra costs in pre-smoothing and surface partitioning, our scheme is quite computationally expensive.

ACKNOWLEDGEMENTS

This work was supported in part by a REEDF grant from the State of Michigan, a Rackham grant from the University of Michigan, a CEEP grant from the University of Michigan, Dearborn, an industrial funding from Altair Engineering Inc. Joseph Pleban helped in preparing some mesh data and in code implementation. Devis Shehu provided a visualization tool.

REFERENCES

1. Page D, Koschan A, Sun Y, Abidi MA. Laser-based imaging for reverse engineering. *Sensor Review* 2003; **23**(3):223–229.
2. Thompson WB, Owen JC, de St.Germain HJ. Feature-based reverse engineering of mechanical parts. *IEEE Transactions on Robotics and Automation* 1999; **15**(1):57–66.

3. Bendsoe MP. Optimal shape design as a material distribution problem. *Structural Optimization* 1989; **1**: 193–202.
4. Bendsoe MP, Sigmund O. Material interpolations in topology optimization. *Archive of Applied Mechanics* 1999; **69**:635–654.
5. Celniker G, Gossard D. Deformable curve and surface finite elements for free-form shape design. *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, vol. 25, 1991; 257–265.
6. Greiner G. Variational design and fairing of spline surfaces. *Computer Graphics Forum* 1994; **13**:143–154.
7. Hoppe H, De Rose T, Duchamp T, MaDonald J, Stuetzle W. Mesh optimization. *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, vol. 27, 1993; 19–26.
8. Hubeli A, Gross MH. Fairing of non-manifolds for visualization. *Proceedings of IEEE Visualization 2000*, 2000; 407–414.
9. Moreton H, Sequin C. Functional optimization for fair surface design. *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, vol. 26, 1992; 167–176.
10. Sapidis N. *Designing Fair Curves and Surfaces*. SIAM: Philadelphia, PA, 1994.
11. Welch W, Witkin A. Variational surface modeling. *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, vol. 26, 1992; 157–166.
12. Kobbelt L, Campagna S, Vorsatz J, Seidel H. Interactive multi-resolution modeling on arbitrary meshes. *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, 1998; 105–114.
13. Kobbelt L. Discrete fairing. *Proceedings of the Seventh IMA Conference on the Mathematics of Surface*, 1997; 101–131.
14. Polden A. Compact surfaces of least total curvature. *Technical Report*, *Preprint SFB 382 Nr. 62*, 1997.
15. Welch W, Witkin A. Free-form shape design using triangulated surfaces. *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, vol. 28, 1994; 247–256.
16. Mallet JL. Discrete smooth interpolation in geometric modeling. *Computer Aided Design* 1992; **24**(4): 178–191.
17. Taubin G. A signal processing approach to fair surface design. *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, 1995; 351–358.
18. Vollmer J, Mencl R, Muller H. Improved Laplacian smoothing of noisy surface meshes. *Computer Graphics Forum* 1999; **18**(3):131–138.
19. Taubin G. Linear anisotropic mesh filtering. *Technical Report RC222 13*, 2001.
20. Belyaev A, Ohtake Y. A comparison of mesh smoothing methods. *Israel–Korea Bi-National Conference on Geometric Modeling and Computer Graphics*, 2003; 83–87.
21. Ohtake Y, Belyaev A, Bogaevski IA. Polyhedral surface smoothing with simultaneous mesh regularization. *Geometric Modeling and Processing 2000*, 2000; 229–237.
22. Ohtake Y, Belyaev A, Seidel H. Mesh smoothing by adaptive and anisotropic Gaussian filter. *Vision*, *Modeling*, *and Visualization 2002*, 2002; 203–210.
23. Yagou H, Ohtake Y, Belyaev A. Mesh smoothing via mean and median filtering applied to face normals. *Geometric Modeling and Processing 2002*, 2002; 124–131.
24. Guskov I, Sweldens W, Schroder P. Multiresolution signal processing for meshes. *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 1999; 325–334.
25. Peng J, Strela V, Zorin D. A simple algorithm for surface denoising. *IEEE Visualization 2001*, 2001; 107–112.
26. Pauly M, Gross M. Spectral processing of point-sampled geometry. *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, 2001; 379–386.
27. Alexa M. Wiener filtering of meshes. *Proceedings of Shape Modeling International*, 2002; 51–57.
28. Desbrun M, Meyer M, Schroder P, Barr AH. Implicit fairing of irregular meshes using diffusion and curvature flow. *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 1999; 317–324.
29. Clarenz U, Diewald U, Rumpf M. Anisotropic geometric diffusion in surface processing. *Proceedings of IEEE Visualization* 2000; 397–405.
30. Clarenz U, Dziuk G, Rumpf M. In *On Generalized Mean Curvature Flow in Surface Processing*, Karcher H, Hildebrandt S (eds). Springer: Berlin, 2003; 217–248.
31. Clarenz U, Rumpf M, Telea A. Robust feature detection and local classification or surfaces based on moment analysis. *IEEE Transactions on Visualization and Computer Graphics* 2004; **10**(5):516–524.
32. Desbrun M, Meyer M, Schroder P, Barr AH. Anisotropic feature-preserving denoising of height fields and bivariate data. *Graphics Interface* 2000; 145–152.

33. Schneider R, Kobbelt L. Generating fair meshes with g1 boundary conditions. *Proceedings of Geometric Modeling and Processing*, 2000; 251–261.
34. Bajaj C, Xu G. Anisotropic diffusion of subdivision surfaces and functions on surfaces. *ACM Transactions on Graphics* 2003; **22**(1):4–32.
35. Meyer M, Desbrun M, Schroder P, Barr AH. Discrete differential-geometry operators for triangulated 2-manifolds. *Proceedings of Visualization and Mathematics*, 2002.
36. Tasdizen T, Whitaker RT, Burchard P, Osher S. Anisotropic geometric diffusion in surface processing. *IEEE Visualization 2002*, 2002; 125–132.
37. Zhang H, Fiume EL. In *Mesh Smoothing with Shape or Feature Preservation*, Vince J, Earnshaw R (eds). Springer: Berlin, 2002; 167–182.
38. Perona P, Malik J. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1990; **12**(7):629–639.
39. Fleishman S, Drori I, Cohen-Or D. Bilateral mesh denoising. *Proceedings of the 30th Annual Conference on Computer Graphics and Interactive Techniques*, 2003; 950–953.
40. Jones TR, Durant F, Desbrun M. Non-iterative, feature-preserving mesh smoothing. *Proceedings of the 30th Annual Conference on Computer Graphics and Interactive Techniques*, 2003; 943–949.
41. Tomasi C, Manduchi R. Bilateral filtering for gray and color images. *Proceedings of IEEE ICCV*, 1998; 836–846.
42. Alboul L. Optimising triangulated polyhedral surfaces with self-intersections. *Proceedings of the 10th IMA International Conference*, vol. 2768, 2003; 48–72.
43. Calladine C. *Gaussian Curvature and Shell Structure*. Clarendon Press: Oxford, 1986; 179–196.
44. Kobbelt L. Discrete fairing and variational subdivision for free-form surface design. *The Visual Computer* 2000; **16**(3/4):142–158.
45. Krsek P, Lukacs G, Martin RR. Algorithms for computing curvatures from range data. *Proceedings of the Eighth IMA Conference on the Mathematics of Surfaces*, 1998; 1–16.
46. Mangan AP, Whitaker RT. Partitioning 3D surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics* 1999; **5**(4):308–321.
47. Meyer M, Desbrun M, Schroder P, Barr AH. Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and Mathematics III*. Springer: Heidelberg, 2003; 35–57.
48. Shen J, Yoon D. A new scheme for efficient and direct shape optimization of complex structures represented by polygonal meshes. *International Journal for Numerical Methods in Engineering* 2003; **58**(4):2201–2223.
49. Hoppe H, De Rose T, Duchamp T, MaDonald J, Stuetzle W. Mesh optimization. *Proceedings of the ACM SIGGRAPH Computer Graphics*, vol. 27, 1993; 19–26.
50. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. *Numerical Recipes in C*: *The Art of Scientific Computing*. Cambridge University Press: Cambridge, MA, 1992; 994.
51. Eberly D. Distance from point to a general quadratic curve or a general quadric surface. *Technical Report*, 1999.
52. Garland M, Heckbert PS. Surface simplification using quadric error metrics. *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, 1997; 209–216.

