Fairing of Polygon Meshes Via Bayesian Discriminant Analysis

Chun-Yen Chen
Institute of Information Science, Academia Sinica.
Department of Computer Science and Information Engineering, National Taiwan University.
115, Taiwan, Taipei, Nankang
ccy@iis.sinica.edu.tw

Kuo-Young Cheng
Institute of Information Science, Academia Sinica.
Department of Computer Science and Information Engineering, National Taiwan University.
115, Taiwan, Taipei, Nankang
kycheng@iis.sinica.edu.tw

Hong-Yuan Mark Liao
Institute of Information Science, Academia Sinica.
115, Taiwan, Taipei, Nankang
liao@iis.sinica.edu.tw

ABSTRACT

Design of an anisotropic diffusion-based filter that performs Bayesian classification for automatic selection of a proper weight for fairing polygon meshes is proposed. The data analysis based on Bayesian classification is adopted to determine the decision boundary for separating potential edge and non-edge vertices in the curvature space. The adaptive diffusion filter is governed by a double-degenerate anisotropic equation that determines how each polygon vertex is moved along its normal direction in the curvature space iteratively until a steady state is reached. The determination of how much a polygon vertex should be moved depends on whether it is a potential edge or a non-edge vertex. At each fairing step, conceptually, a bi-directional curvature map whose boundary line while couched by the weight value can be plotted to understand the type of a vertex. Experimental results show that the proposed diffusion-based approach could effectively smooth out noises while retaining useful data to a very good degree.

Keywords

Polygon Mesh Smoothing, Bayesian Discriminant Analysis, Feature Detection.

1. INTRODUCTION

When one tries to acquire a 3D object model by a 3D laser scanner, the scanned object shape is, usually, represented in the form of triangular polygon meshes. However, such sampled points are always inherited with noises and the resultant polygon surface looks faceted due to the existence of noises. A technique that can remove noise while retaining polygon surface is always preferable. Such a technique falls into the category of special filter design, in which a degraded polygon shape is smoothed by gradually fairing out noise iteratively.

The information that can be obtained from a scanned 3D object model includes the position of a vertex, the passing tangent planes, and the curvatures of surfaces that are formed by tangent planes. A noisy vertex causes signification of local facet effects and thus produces low quality surfaces. It is well known that the filter design based upon a constrained optimization technique is able to generate satisfactory surface with high quality. In this paper, we let the smoothing of the data of a scanned object governed by a thin shell heat equation. In the smoothing process, each vertex is considered as a point temperature and the curvature of a vertex is considered as the energy associated with it. Therefore, the filter design problem is converted into an optimization problem. In the optimization process, the proposed algorithm tries to locate a new position for each vertex so that the associated energy can be minimized. Because different vertex types have different curvature values, they will be governed by the heat equation with different heat transfer rates.

In the literature, the most similar existing approach is the discrete fairing technique proposed by Kobbelt et al. [Kob98]. In that work, they applied a so-called umbrella algorithm together with a thin plate energy function to derive a recursive equation for designing their filter. However, they used a fixed weight value for relocating each vertex during the iterative process. It is well know that the polygon smoothing process based on the filter design concept

Journal of WSCG, Vol.12, No.1-3, ISSN 1213-6972
WSCG'2004, February 2-6, 2004, Plzen, Czech Republic.

relies heavily on the local fine structure as well as the noise that associated with the given 3-D polygon model. Without considering the vertex type and different weights for different vertices, the designed smoothing algorithm may produce some unwanted facets in the final output results.

In this paper, we propose a method for polygon smoothing using a filter design approach. The proposed scheme is able to perform intelligent data analysis and to classify each vertex as either a feature vertex or a non-feature vertex. This classification result can be used as the basis for selecting appropriate weight for each vertex at a fairing step. To differentiate our method with others, we call the filter design based on our proposed method the adaptive diffusion filter design. The experimental results show that the proposed adaptive diffusion filter can improve the resultant surface quality significantly as compared to some previous polygon filter designs.

2. FILTER DESIGN USING DIFFUSION EQUATIONS: A SURVEY

In an isotropic diffusion-based method, the design of a low pass or a band pass filter is to use a diffusion-style partial differential equation to control the process of event operation. The isotropic diffusion equation that will fit in the polygon mesh fairing process can be defined as follows:

$$\frac{\partial M}{\partial t} = \nabla^2 M , \qquad (1)$$

where ∇^2 is a Laplacian operator and M represents a given polygon mesh model. An analytic solution for equation (1) can be expressed as a form of a linear invariant Gaussian kernel convoluted with M [Tau96]. However, an analytic method that can be used to solve equation (1) in 3D space is rather complicated. Therefore, it is usually solved by a finite difference approach using the following iterative equation:

$$M^{t+1} = M^t + \lambda \nabla^2 M^t , \qquad (2)$$

where λ is a user-defined constant. For speeding up the process described in Equation (2), an umbrella operator is adopted to linearly approximate the Laplacian operator. In [Tau96], an umbrella operator proposed by Taubin is defined as follows:

$$\nabla^2(x_i) = \frac{1}{\#N(x_i)} \sum_{x_j \in N(x_i)} x_j - x_i , \qquad (3)$$

where $N(x_i)$ indicates the neighboring vertices of x_i, and $\#N(x_i)$ indicates the number of vertices being contained in $N(x_i)$.

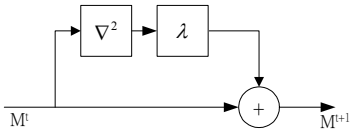


Figure 1. The Laplacian filter.

Equation (2) can be expressed as an isotropic filter as shown in Figure 1. It is basically a Gaussian diffusion filter which contains two main components: the Laplacian operator ∇^2 and the weighting constant λ. The Laplacian operator is used for calculating the normal difference between a vertex and its neighboring vertices. The value of the weighting constant determines the degree of diffusion. A filter can be called an isotropic diffusion filter if the defined weight is maintained constant in the whole process; otherwise, we call it an anisotropic one.

Taubin [Tau96] found that a Gaussian-type diffusion process can suppress noise very effectively but also create a model-shrinking problem. In order to solve the model-shrinking problem, he designed a mutually compensated diffusion filter called Taubin diffusion filter as shown in Figure 2. The process of a Taubin filter can be defined by the following iterative procedure:

$$\begin{cases} \overline{M}^t = M^t + \lambda \nabla^2 M^t \\ M^{t+1} = \overline{M}^t + \mu \nabla^2 \overline{M}^t \end{cases} . \qquad (4)$$

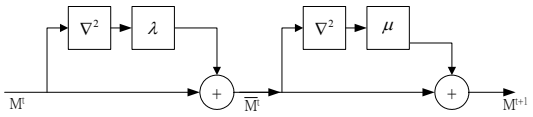


Figure 2. The Taubin diffusion filter.

Figure 2 illustrates how a Taubin filter works. In Figure 2, λ and μ are two preset constants with one positive and the other negative. The λ filter is a Gaussian diffusion filter which enforces the diffusion direction to make the model shrink a little bit. The μ filter, on the other hand, is a Gaussian diffusion filter which enforces the diffusion direction to make the model expand a little bit. With a proper selection of λ and μ, the Taubin diffusion filter can function efficiently to remove noises while retaining the original mesh data as much as possible.

In order to achieve better convergence result, Desbrun et al. [Des99] proposed an implicit integration filter design approach. The iterative equations of their approach are as follows:

$$M^{t+1} = M^t + \lambda \nabla^2 M^{t+1} \text{ or } (1 - \lambda \nabla^2) M^{t+1} = M^t . \qquad (5)$$

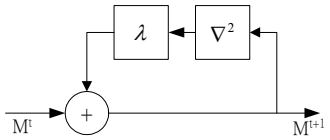


Figure 3. The implicit filter.

Figure 3 illustrates how Desbrun et al.'s approach functions. Equation (5) is basically a representation of implicit system equations. Under these circumstances, if one would like to derive M^{t+1} from M^t, a numerical method using sparse matrix inversion or pre-conditional bi-conjugate gradient is required to solve the equations. It is known that the above mentioned numerical methods will eventually bring to a stable solution, but they are time-consuming.

Desbrun et al. [Des99] pointed out that the umbrella operator is inappropriate in dealing with non-symmetric polygon mesh. This is because the vertex of a non-symmetric polygon mesh will shift away from its original position when encountering a diffusion operation. Therefore, they proposed to use a mean curvature flow operator to replace an umbrella operator. The mean curvature flow operator is defined as follows:

$$\nabla_c^2(x_i) = \kappa_{iH}\overline{n}_i, \qquad (6)$$

where ∇_c^2 is the mean curvature flow operator, x_i is the vertex subjected to the mean curvature flow operation, κ_{iH} is the mean curvature on the vertex x_i, and $\overline{n}_i$ is the unit normal vector on x_i. Usually, the vertex x_i of a straight plane will stay steady when encountering a mean curvature flow operation. As a consequence, Equation (5) can be rewritten as:

$$(1-\lambda\nabla_c^2)M^{t+1} = M^t, \quad (7)$$

and the filter design of the implicit curvature flow system can be modified and shown in Figure 4.

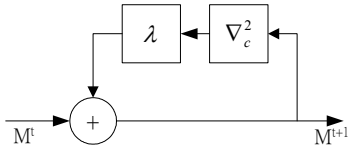


Figure 4. The implicit curvature flow system.

The filters using Gaussian diffusion, Taubin diffusion, mean curvature flow diffusion, or their combinations are classified as isotropic diffusion-type filters. The factor λ remains constant in an isotropic diffusion operation, regardless of diffusion direction. An isotropic diffusion operation can eliminate noise very effectively but also smooth out useful data. In [Des99], Desbrun et al. proposed to execute filter design based on anisotropic diffusion. An anisotropic filter uses a function of two principal curvatures, κ_1 and κ_2, as the weights for each diffusion direction. The operation of an anisotropic filter can be represented graphically as shown in Figure 5, and the equation that describes this operation is as follows:

$$M^{t+1} = M^t + w(\kappa_1,\kappa_2)\nabla_c^2 M^t, \quad (8)$$

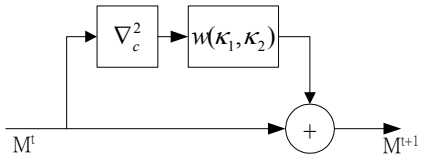


Figure 5. An anisotropic filter.

where $w(\kappa_1,\kappa_2)$ is an anisotropic diffusion weighting function. With the proper design of $w(\kappa_1,\kappa_2)$ in accordance with diffusion directions, the designed anisotropic filter can effectively remove the noise and at the same time preserve the shape of corners and edges.

According to the filter design rule described in [Mey02], if the values of $|\kappa_1|$ and $|\kappa_2|$ of a vertex are both smaller than a preset threshold, then the vertex is regarded as a noisy vertex. Under these circumstances, the noise associated with the vertex can be removed effectively by a Gaussian diffusion filter or an anisotropic filter if $w(\kappa_1,\kappa_2)$ is set to 1. If the absolute values of both principal curvatures are larger than a preset threshold value, then the vertex is regarded as a corner vertex and it is remained intact by setting $w(\kappa_1,\kappa_2)$ equal to 0. A vertex is considered an edge vertex if its minimum curvature is very small and its mean curvature very large. To determine the value of $w(\kappa_1,\kappa_2)$, the following rules will be used:

$$w(\kappa_1,\kappa_2) = \begin{cases} 1 & \text{if } |\kappa_1| \le T \text{ and } |\kappa_2| \le T \\ 0 & \text{if } |\kappa_1| > T \text{ and } |\kappa_2| > T \text{ and } \kappa_1\kappa_2 > 0 \\ \kappa_1/\kappa_H & \text{if } |\kappa_1| = \min(|\kappa_1|,|\kappa_2|,|\kappa_H|) \\ \kappa_2/\kappa_H & \text{if } |\kappa_2| = \min(|\kappa_1|,|\kappa_2|,|\kappa_H|) \\ -1 & \text{if } |\kappa_H| = \min(|\kappa_1|,|\kappa_2|,|\kappa_H|) \end{cases} \quad (9)$$

In addition to the definition of κ_1 and κ_2 which has been made previously, κ_H here represents the mean curvature of κ_1 and κ_2.

Fleishman et al. [Fle03] extended the bilateral filtering method from 2D image to 3D polygon meshes. Their method adopted two standard Gaussian filters, one is smoothing function and the other is similarity function. The similarity function identifies the degree

of similarity. If a vertex has high similarity with its neighboring vertices, then it will be smoothed without any hesitation; otherwise, it will be identified as a feature vertex and will not be smoothed. By tuning up smoothing and similarity functions, the bilateral filtering method will produce a feature preserving result for semi-regular polygon meshes.

The underlying concept of anisotropic diffusion is from the heat transfer theory, where a zero weight corresponds to an insulator and a full weight corresponds to a perfect conductor. In what follows, we shall propose a new anisotropic diffusion-based filter to execute the fairing process of 3D polygon models.

3. FILTER DESIGN BASED ON ADAPTIVE ANISOTROPIC DIFFUSION

In the previous section, we have surveyed a number of isotropic and anisotropic diffusion filters which can be applied to smooth 3D polygon meshes. In this paper, we propose a bi-directional curvature mapping function, $w(\kappa_1,\kappa_2)$, based on the heat transfer theory. A heat transfer expression is commonly used in nonlinear filtering [Wei97]. Let κ_1 be the maximum curvature and κ_2 be the minimum curvature. Assume the relation $|\kappa_1| \geq |\kappa_2|$ always holds. The weighting function $w(\kappa_1,\kappa_2)$ of a bi-directional curvature mapping can be expressed as follows:

$$w(\kappa_1,\kappa_2) = \begin{cases} 1 & \text{if } |\kappa_1| \leq T \text{ and } |\kappa_2| \leq T \\ 0 & \text{if } |\kappa_1| \geq |\kappa_2| > 2T \\ e^{-(k_2/k_1-1)^2} - e^{-1} & \text{if } |\kappa_1| > T \text{ and } |\kappa_2| \leq T \\ e^{-((2T-k_2)/k_1-1)^2} - e^{-1} & \text{if } |\kappa_1| > T \text{ and } T < |\kappa_2| \leq 2T \end{cases} \quad (10)$$

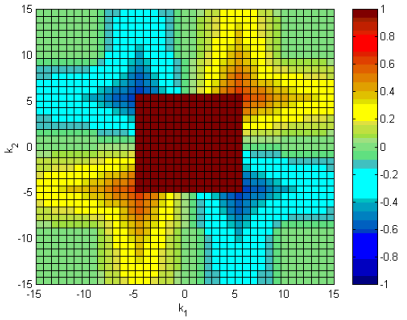


Figure 6. The bi-directional curvature map with *T=5*.

In addition to the new weighting function, we also propose a double degenerate heat equation [Kob98] to govern the anisotropic diffusion-based filter. The equation showing how this operation works is as follows:

$$M^{t+1} = M^t + \nabla_c^2[w(\kappa_1,\kappa_2)\nabla_c^2 M^t] \quad (11)$$

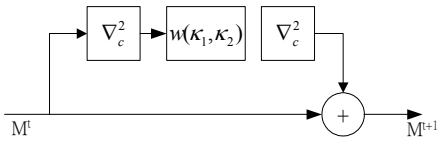


Figure 7. The double degenerate anisotropic diffusion-based filter.

Figure 7 illustrates graphically how the two degenerate heat equations are incorporated into the design of the filter. An anisotropic diffusion filter using the bi-directional curvature mapping function defined in Equation (10) works well if, at each fairing step, the threshold value T can be properly determined. The threshold T can be used to judge the vertex type in the fairing process. Having the type of a vertex, the fairing algorithm triggers an appropriate process to smooth out the vertex. Therefore, the selection of an appropriate T is of great importance to the success of a fairing process.

To the best of our knowledge, most existing anisotropic diffusion methods require user to provide a threshold value for a fairing process. However, this threshold value is data dependent (ill-posed) and different threshold values may result in different fairing results. Therefore, an automatic threshold selection procedure that can adaptively decide an appropriate threshold for different mesh models is always preferable. In this paper, we propose a method which can adaptively select an appropriate threshold value that will fit in any given 3D polygon model. The method is based on Bayesian classification which is commonly used in the field of pattern recognition [Sch92]. In Section 3.1, we shall describe how to systematically separate the feature vertices and non-feature vertices of a 3D polygon model. Section 3.2 will discuss how to distinguish the edge and corner vertices from the set of feature vertices. The convergence test described in Section 3.3 discusses how to stabilize our algorithms.

3.1 Vertex Classification

For a 3D polygon model, edge and corner vertices are both classified as feature vertices [Mey02]. The difference between an edge vertex and a corner vertex can be judged by the magnitudes of their two principal curvatures. For an edge vertex, the magnitude of one of its principal curvatures is small and the other is quite large. For a corner vertex, the magnitudes of its both principal curvatures are large. Therefore, if one would like to distinguish feature vertices from non-feature vertices, the magnitude of the maximum principal curvature can be used as a good indicator. For those non-feature vertices, the magnitude of their maximum principal curvature is small.

For distinguishing feature vertices from non-feature vertices, we assume that non-feature vertices and feature vertices of a given 3-D model are normal distributed along the maximum principal curvature axis. Let μ_1 and μ_2 be the mean values for the two classes, σ_1 and σ_2 be two corresponding standard deviations. Then, by applying Bayesian classification rule [Sch92], the best threshold value can be determined by the boundary line that best divides the two classes with the following property:

$$p(x \mid w_1) = p(x \mid w_2), \qquad (12)$$

where w_1 and w_2 are the two classes and x is the best threshold value. Under these circumstances, the best threshold value can be derived by solving equations (12), (13), and (14), where equations (13) and (14) are as follows:

$$p(x \mid w_1) = \frac{1}{\sqrt{2\pi}\sigma_1} \exp(-\frac{1}{2}(\frac{x-\mu_1}{\sigma_1})^2), \qquad (13)$$

$$p(x \mid w_2) = \frac{1}{\sqrt{2\pi}\sigma_2} \exp(-\frac{1}{2}(\frac{x-\mu_2}{\sigma_2})^2). \qquad (14)$$

The best solution for the threshold value is thus obtained as follows:

$$x = \frac{b - \sqrt{c}}{2 \cdot a}, \qquad (15)$$

where

$a = \sigma_2^2 - \sigma_1^2$, $b = 2 \cdot (\sigma_2^2 \mu_1 - \sigma_1^2 \mu_2)$, $c = b^2 - 4ad$, and

$d = \sigma_2^2 \mu_1^2 - \sigma_1^2 \mu_2^2 - 2\sigma_1^2 \sigma_2^2 \ln(\frac{\sigma_2}{\sigma_1})$.

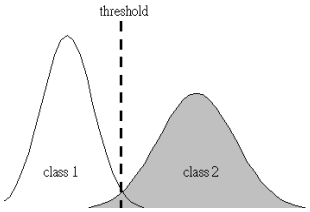


Figure 8. The dashed line shows where the best threshold is located by a Bayesian classifier.

Since the classification process is automatically performed on every given 3D model, the number of classes is an important issue. For a 3D shape such as a sphere, there is no edge or corner on its surface. As a result, there is only one class for a sphere. Therefore, before the classification task is executed, we have to check every 3D model to see if it has at least two separable classes. In order to do so, we define the following function:

$$D(\mu, \sigma, th) = (\mu - th) / \sigma, \qquad (16)$$

where *th* is the threshold value derived from equation (15). If both values of $D(\mu_1, \sigma_1, th)$ and $D(\mu_2, \sigma_2, th)$ are less than a preset δ value, then the vertices of a given 3D polygon model are not separable and all of them are regarded as non-feature vertices. For the rest of the paper, we use $\delta = 2$ in all examples. That is, there exists only one non-separable class if the distance between two calculated means, μ_1 and μ_2, is less than $2\sigma_1 + 2\sigma_2$.

3.2 Edge and Corner vertices

The reason why we have to perform feature and non-feature vertex classification is because they will be processed differently in the subsequent steps of our algorithm. A vertex is regarded as a potential edge vertex if it has at least two neighboring feature vertices in its 1-ring, where 1-ring of a vertex is the collection of common planar vertices of the vertex. If a feature vertex has a minimum curvature, which is larger than the threshold value obtained from equation (15), and it has at least three neighboring potential feature vertices in its 1-ring, then it is regarded as a potential corner vertex. If a vertex is not of one of the above types, then it is classified as a non-feature vertex. The formal algorithm that can be applied to detect edge, corner, or non-feature vertices is as follows:

Corner vertex:

If $|k_1| > threshold$ and $|k_2| > threshold$ and there are at least 3 vertices in 1-ring neighbor with their $|k_1| > threshold$.

Edge vertex:

If $|k_1| > threshold$ and there are at least 2 vertices in 1-ring neighbor with their $|k_1| > threshold$.

Non-feature vertex:

None of the above two types.

Here k_1 and k_2 represent the maximum and the minimum principal curvature, respectively, of an arbitrary vertex on a 3D model. Figure 9 illustrates an example showing how the feature vertex classification algorithm works. The left hand side of Figure 9 shows a fandisk model. On its right is the result after performing feature vertex classification.

Figure 9. On the left is a given fandisk model; On the right shows the calculated feature vertices (in dark color).

3.3 Convergence Test

A vertex is said to reach a steady state if the change of its curvature values in two consecutive iterations becomes negligible. Whenever a vertex reaches its steady state, its position should be remained intact in the subsequent iterations. Thus, we need a convergence test for each vertex to determine whether a vertex has reached its steady state. Therefore, we conduct a convergence test in which we calculate the total curvature and then use it to test the convergence of non-feature vertex. On the other hand, we use the minimum principal curvature of every vertex to test the convergence of feature vertex. The function that is adapted to the convergence test for a vertex at the *p*-th iteration is defined as follows:

$$C^p(v) = \begin{cases} (tc^p < \varepsilon) \vee \left(\left|\frac{tc^{p-1}}{tc^p} - 1\right| < \varepsilon\right) & if \quad v \notin \{F\} \\ (\left|k_2^p\right| < \varepsilon) \vee \left(\left|\frac{k_2^{p-1}}{k_2^p} - 1\right| < \varepsilon\right) & if \quad v \in \{F\} \end{cases} \quad (17)$$

where the superscript p indicates the p-th iteration and the total curvature tc can be calculated by $|k_1| + |k_2|$, $\{F\}$ is the set of feature vertices, and ε is a preset threshold value. The value of ε has to be small and we set it 0.01 in our experiments.

With the iteration goes on, we say a vertex is converged or has reached to its steady state if it passes the convergence test. When a vertex reaches its steady state, the corresponding feature value remains intact in the subsequent iteration steps. It is noted that the convergence rate of each vertex is different. Therefore, the number of vertices that reaches to steady state will decrease gradually as the iterative process goes on. The smoothing process for polygon meshes is completed when all vertices reach their steady state. Figure 10 illustrates the flow chart of the proposed adaptive anisotropic diffusion process.

4. EXPERIMENTAL RESULTS

A series of experiments was conducted to verify the effectiveness of the proposed method. In the first set of experiments, we conducted experiments on two sets of non-separable models. Figure 11(a) shows a noisy sphere and its corresponding principal curvature distribution. Since the distribution was non-separable, our algorithm triggered an isotropic diffusion process to smooth out the sphere. The result is shown in Figure 11(b). Figure 12 is another example. The distributions of the model taken from different views are shown in Figure 12(a) and (c) were non-separable. After applying an isotropic diffusion process, the results are shown in Figure 12 (b) and (d), respectively. As to the case when our algorithm is facing a separable 3D model, we have obtained the following results. Figure 13(a) shows a noisy 3D polygon model. The results obtained by applying the proposed adaptive diffusion process, the anisotropic mean curvature method, the bilateral filtering method and the Gaussian smoothing, are shown in Figure 13(b), (c), (d) and (e), respectively. It is obvious that our algorithm was powerful in terms of preserving edges and corners. The anisotropic mean curvature method was able preserve edges but fail to retain corners. The reason that led to this failure was due to the use of fixed threshold value. The bilateral method was able to preserve edge features but also damage some corner and edge vertices. This is due to the use of a fixed coefficient value to determine the similarity degree for all vertices. As to the Gaussian filtering, it failed to preserve both the edges and corners. This outcome is predictable because a Gaussian filter always filters out both noise and useful data.

In Figure 14, we show several principal curvature distributions corresponding to different iteration numbers. Figure 14(a) shows the original principal curvature distribution before our algorithm was applied to the 3D model shown in Figure 13(a). Figure 14(b) shows the distributions after our algorithm was applied 5 iterations. Finally, Figure 14(c) shows the final distribution after our algorithm completed its job (16 iterations).

In the last set of experiments, we tested our algorithm against some noisy 3D models. Figure 15(a) shows a "fandisk" model. Figure 15(b) shows a corrupted model with 0.3% Gaussian noise. Figure 15(c) and (d) show, respectively, the results after applying our adaptive smoothing algorithm and the Gaussian model. It is apparent that our algorithm could well preserve both edges and corners in this case. Figure 16(a) is a golf mesh model. Figure 16(b) shows 1% of Gaussian noise was added to the original model. The result obtained after applying our algorithm is shown in figure 16(c).

5. CONCLUSIONS

In this paper, we have proposed an anisotropic diffusion-based method to smooth 3D polygon meshes. We used a bi-directional curvature mapping function which is based on the heat transfer theory to deal with different types of vertices. In order to distinguish the vertex type, we perform Bayesian classification to separate all vertices of a 3D model into feature and non-feature vertices. Then an energy-based convergence test function was used to check whether each vertex has reached its steady state or not. Experimental results show that our approach was indeed powerful in smoothing different 3D polygon models.

6. REFERENCES

[Des99] M. Desbrun, M. Meyer, P. Schroder, and A. H. Barr. Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow. SIGGRAPH'99 Conference Proceeings, 317–324, 1999.

[Fle03] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral Mesh Denoising. SIGGRAPH '03 Conference Proceedings, 950-953, 2003.

[Kob98] L. Kobbelt, S. Campagna, T. Vorsatz, and H. P. Seidel. Interactive multiresolution modeling on arbitary meshes. SIGGRAPH'98 Conference Proceeings, 105–114, 1998.

[Mey02] M. Meyer, M. Desbrun, P. Schroder, and A. H. Barr. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds, Visualization and Mathematics Proceedings, May, 2002.

[Sch92] R. J. Schalkoff, Pattern Recognition: Statistical, Structural and Neural Approaches. John Wiley & Sons. press 1992.

[Tau96] G. Taubin. Optimal Surface Smoothing as Filter Design. Research Report RC-20404, IBM Research,. March 1996.

[Wei97] J. Weickert, A Review of Nonlinear Diffusion Filtering, in Scale-space Theory for Computer Vision, Lecture Notes in Computer Science, Vol. 1252, Bartter Haar Romeny, Ed., 3-28, Springer, New York, 1997.

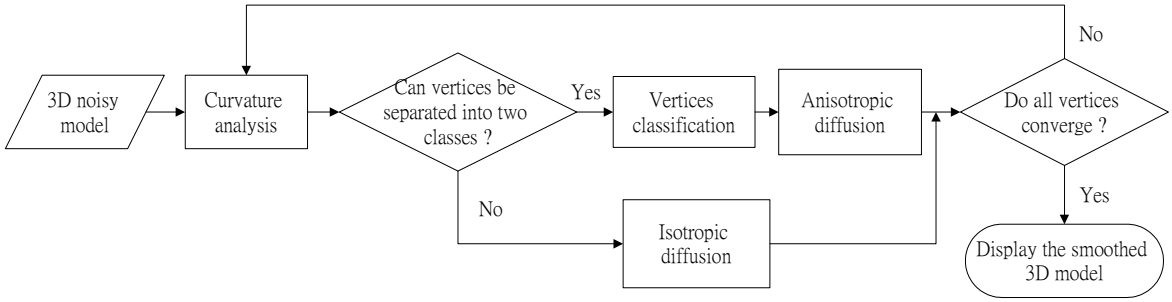
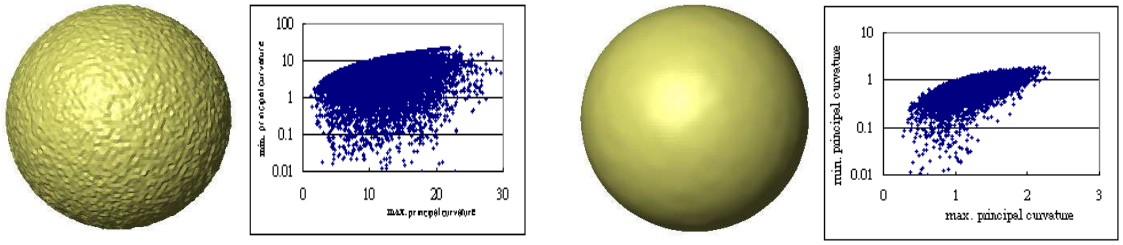
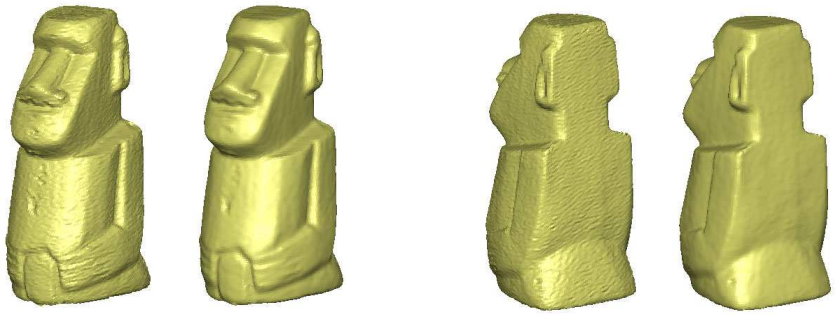


Figure 10. The adaptive diffusion filtering process.



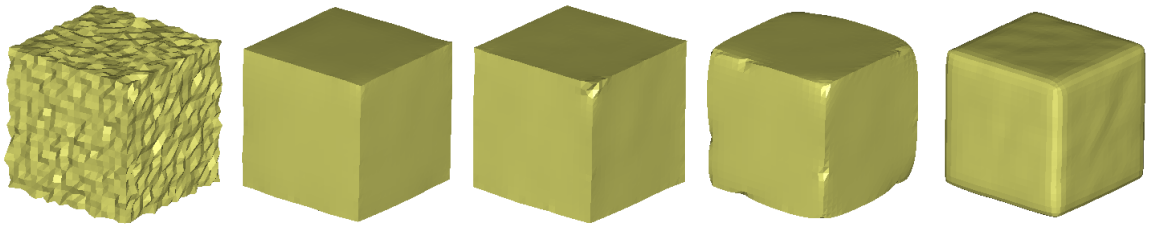
(a) **(b)**

Figure 11. (a) A noisy sphere model and its principal curvature distribution, (b) the result obtained after applying our adaptive diffusion algorithm.



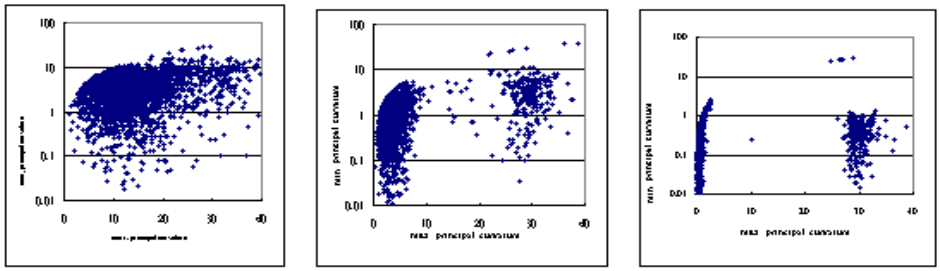
(a) **(b)** **(c)** **(d)**

Figure 12. Experimental results of another model. (a) and (c) are original model taken from different views, (b) and (d) are the results obtained after smoothing.



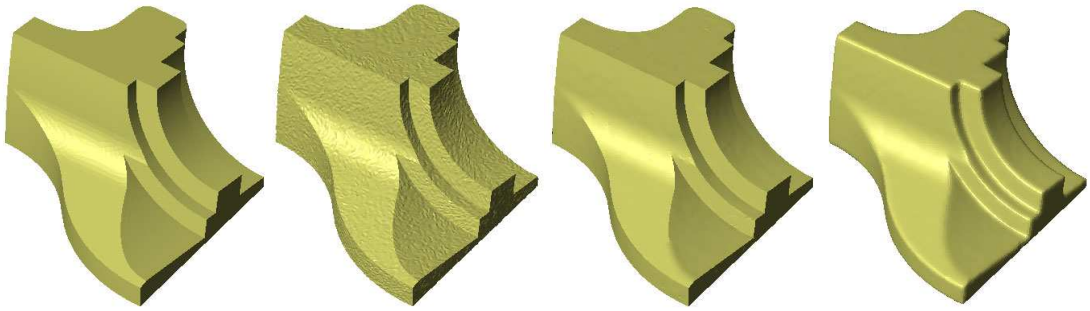
(a) (b) (c) (d) (e)

Figure13. (a) Original polygon model, and (b) the result of the proposed adaptive diffusion, (c) the result of the anisotropic mean curvature flow, (d) the result of bilateral filtering, and (e) the result of Gaussian smoothing.



(a) (b) (c)

Figure 14. The principal curvature distributions of the noisy model shown in Figure 13. (a) original data, (b) applying adaptive diffusion 5 iterations, (c) smoothing procedure finished.



(a) (b) (c) (d)

Figure 15. Experiments on a Fandisk model. (a) original polygon meshes, (b) the model with 0.3% Gaussian noise, and (c) (d) the results obtained by applying our adaptive diffusion method and Gaussian smoothing method, respectively.

(a) (b) (c)

Figure 16. Our adaptive diffusion is applied to a golf mesh model, (a) original meshes, (b) add 1% Gaussian noise, (c) the filtered result.