An effective third-order local fitting patch and its application

Zhong Li[1,2,3], Brian Barsky[3], Xiaogang Jin[2]
[1] Department of Mathematics and Science, Zhejiang Sci-Tech University, Hangzhou 310018, P.R. China
[2] State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, P.R. China
[3] Computer Science Division, University of California, Berkeley, CA 94720, USA
E-mail: lizhong@zstu.edu.cn

***Abstract*—In this paper, we extend Razdan and Bae's second-order local fitting method [11] to construct an effective third-order fitting patch. Compared to other estimation algorithms, this weighted bicubic Bézier patch more accurately obtains the normal vector and curvature estimation of a triangular mesh model. Furthermore, we define the principal geodesic torsion of each vertex on the mesh model and estimate it through this local fitting patch. In the end of this paper, we apply the third-order fitting patch for the mesh smoothing and hole-filling which can get the satisfactory results.**

***Keywords*—bicubic Bézier surface; normal vector; curvature; torsion**

1. INTRODUCTION

With the fast development of geometry scanners such as 3D scanner, CT and MRI, robust and efficient geometry processing becomes increasingly desirable. Triangular meshes have become a popular representation for 3D geometry modeling. For triangular meshes, it is important to precisely estimate the relevant geometric properties such as the normal vector and curvatures of discrete data points, which influence further operations such as feature recognition, segmentation, or shape analysis [1-4]. The estimation of the normal vector and curvatures is fundamental work for digital geometry processing [5].

For normal vector estimation, many researchers have used a fixed number of Euclidean nearest neighboring points to estimate the normal vector at a given point. These nearest neighbors were used to fit the tangent plane at the point or to fit local quadric surface [1,2]. Other researchers constructed a polygonal mesh surface for a point cloud set and used the polygonal mesh to identify the neighboring points according to the connected polygonal facets [3,4]. Recently, OuYang and Feng [5] provided a new method of normal vector estimation based on the local Voronoi mesh and quadric curve fitting, which produced better normal vector estimation than other existing algorithms.

There are two broad categories to estimate curvature of triangular meshes: discrete and continuous. The first approximates curvatures by formulating a closed form for differential geometry operators which work directly on the discrete representation of the underlying surface that generated the triangular mesh [6-8]. The latter fits a local surface, then computes curvatures by interrogating the fitted surface [9,10]. Recently, Razdan and Bae [11] gave a new curvature estimation based on the weighted local bi-quadratic Bézier patch. They add a weighted factor and the smooth equations in the traditional Bézier patch which can obtain the better curvature estimation.

Torsion, as one of the differential geometric properties, is an important invariant value for the rigid transformation. For the point on the smooth space curve, the torsion value can be calculated from the Frenet formula [12]. For the point on the given space point set, Lewiner [13] estimated the torsion by constructing the parametric curve via the least squares method. For the point on the smooth surface, the torsion is normally measured by the geodesic torsion. Because the curve passing the given point on the surface is not determined, we generally compute the geodesic torsion according to some specific curves passing through the given point on the surface. For the torsion estimation on the vertex of the discrete mesh model, there is little attention and are few applications in the mesh processing [14,15].

In this paper, we provide an effective third-order local fitting patch to estimate discrete geometric properties on the mesh model such as the normal vector, curvature and torsion. We also apply this local fitting patch for the mesh smoothing and hole-filling. This paper has the following contributions:

(1). When constructing the third-order fitting patch, we find a desirable adjusting matrix in the traditional bi-cubic Bézier surface. It gives a better estimation of the normal vector and curvature of each vertex on the mesh model than the weighted second-order fitting patch.

(2). Currently there is little research on the torsion estimation of the mesh model. We define the principal geodesic torsion of each vertex and use this local fitting patch to estimate the torsion property of each vertex on the discrete mesh model.

(3).When smoothing the mesh model and filling the hole of the mesh model, we use this local fitting patch to obtain the satisfactory results.

2. THE THIRD-ORDER FITTING PATCH

In a mesh model, in order to estimate geometric properties such as normal vector and curvature, a popular approach is to use quadratic or cubic algebraic surface to fit each vertex and its neighborhood by the least squares method. Recently, Razdan and Bae [11] developed a weighted bi-quadratic Bézier surface to fit the vertex and local neighborhood. The advantages of the parametric surface fitting method are that it is easily designed and for mesh surfaces with noise, it can add an adjusting matrix and factor to fit the local area, which gives a more accurate fitting surface.

Since curvature is related to the computation of second-order derivatives and the third-order surface is a better fit to the shape of a local area than the second-order surface is, here we use the weighted bicubic Bézier surface to fit the vertex p_i and its 2-ring neighborhood, as shown in Fig. 1.

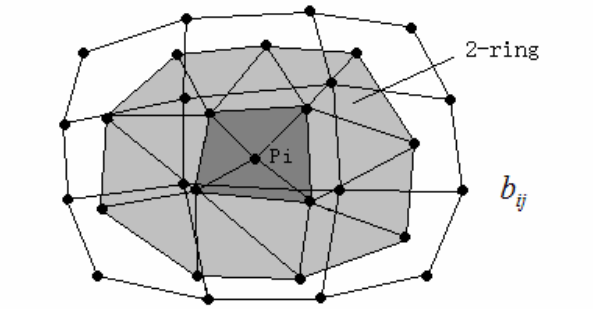


Fig. 1. Fit the vertex and its 2-ring neighborhood by bi-cubic Bézier surface

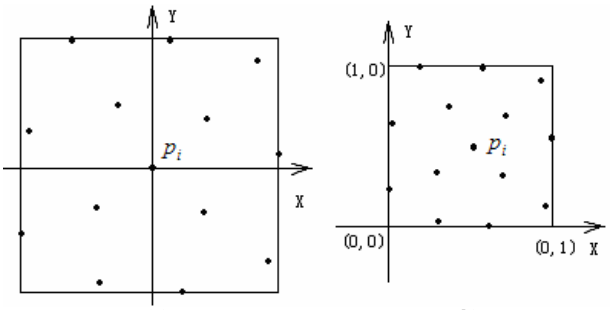
A bicubic Bézier surface is written as

$$B(u,v)=\sum_{i=0}^{3}\sum_{j=0}^{3}B_{i,3}(u)B_{j,3}(v)b_{ij}\ ,\ u,v\in[0,1],$$

where $B_{i,3}(u)$, $B_{j,3}(v)$ are Bernstein basis functions and b_{ij} are the Bézier control points which form the Bézier control net.

To fit a given vertex p_i and all vertices in the 2-ring neighborhood by a bicubic Bézier surface, we need to compute the positions of control points b_{ij} according to these given vertices. Denoting the parameters of each vertex in the fitting surface by (u_i,v_i), the parameters of these vertices are required to be calculated at first. They are acquired by projecting vertices onto a tangent plane and scaling them to the $[0,1]^2$ range.

Assuming there are n vertices in the 2-ring neighborhood of the vertex p_i, we first compute the approximate normal vector N at p_i by the arithmetic average of all neighboring triangles' normal vectors. Second, we build a tangent plane which is vertical to N and set p_i as the origin of its coordinate system. Then we construct the local Cartesian coordinates in this plane. The direction from one projected point to the vertex p_i is set as the x-axis and one vertical direction as the y-axis. The coordinates of all projected points are enclosed by a bounding rectangle. This rectangle is finally scaled to $[0,1]^2$. The coordinates of projected points in this range are regarded as the corresponding parameters of given vertices, as shown in Fig. 2. When projecting these vertices in the neighborhood onto the tangent plane, if there are folding cases or other particular situations, for example, some projected points of vertices in the tangent plane are coincident or some lines connected by two adjacent projected points are self-intersecting. We change another fitting point as the origin of the coordinate system to reconstruct a new tangent plane or restrict vertices in the 1-ring neighborhood as fitting points to avoid these problems.



(a) (b)

Fig. 2. The parameter acquisition of all fitting vertices. (a) The coordinates of projected points on the tangent plane. (b) Scaling these coordinates to $[0,1]^2$.

After obtaining the corresponding parameters (u_i,v_i) of vertex p_i and other fitting vertices, we build a linear equation system $Ax=B$, where

$$A=\begin{bmatrix} B_0^3(u_0)B_0^3(v_0) & B_0^3(u_0)B_1^3(v_0) & \cdots & B_3^3(u_0)B_3^3(v_0) \\ B_0^3(u_1)B_0^3(v_1) & B_0^3(u_1)B_1^3(v_1) & \cdots & B_3^3(u_1)B_3^3(v_1) \\ & & \vdots & \\ B_0^3(u_n)B_0^3(v_n) & B_0^3(u_n)B_1^3(v_n) & \cdots & B_3^3(u_n)B_3^3(v_n) \end{bmatrix},$$

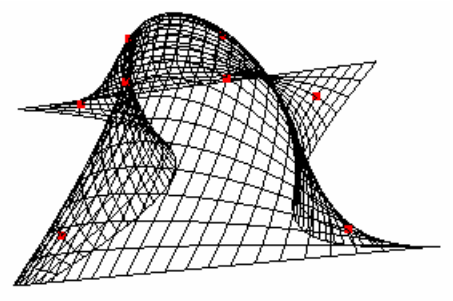
$$x=[b_{0,0}\quad b_{0,1}\quad \cdots\quad b_{3,3}]^T,\ B=[p_0\quad p_1\quad \cdots\quad p_n]^T.$$

From this system of equations, we compute x to determine the control points b_{ij}, which construct the bicubic Bézier surface.

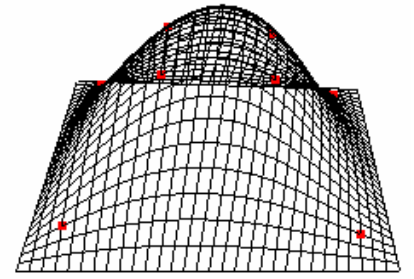
When using the least squares method to solve above system, the fitting surface may not be the effective fitting surface. To construct the fitting patch more accurately, we add an adjusting matrix and a factor to modify the system as follows

$$\begin{bmatrix} aA \\ (1-a)S \end{bmatrix}[x]=\begin{bmatrix} aB \\ 0 \end{bmatrix},$$

where A, x, B are defined above, the matrix S is added to control the shape of fitting surface. Choosing different matrices S influences the fitting precision of the local patch, as shown Fig. 3.



(a)



(b)

Fig. 3. The bicubic Bézier surface using different matrix S to fit 8 vertices. (a). Using random matrix S. (b). Using our matrix S.

To let the fitting patch be smooth, one method is to make the control point distribution of the bicubic Bézier surface as uniform as possible. From the experimental result, we find the following matrix S is a good solution which means to minimize the second differences of the boundaries in the control net

$$S=\begin{bmatrix} 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The factor a is in [0,1] and is adjusted according to the noise density. When noise is dense, a is set lower, whereas, a is set higher. Currently the noise density is determined by the subjective judgment. We set a to 0.8 for the general mesh model.

For the above system, we use the least squares method to determine the solution to obtain the control points of a bicubic Bé zier surface. At this time, vertex p_i is identical to the corresponding point $B(u_i,v_i)$ on the fitting surface. We estimate the geometric properties of pi according to the point $B(u_i,v_i)$.

2.1 Normal estimation of the mesh model

The normal vector is estimated by above parametric surface. For each point p_i, representing the point $B(u_i,v_i)$ on the bicubic Bézier patch, the unit normal vector is computed as $n = B_u \times B_v / | B_u \times B_v |$. The normal vector of the bunny mesh model is shown in Fig. 4.

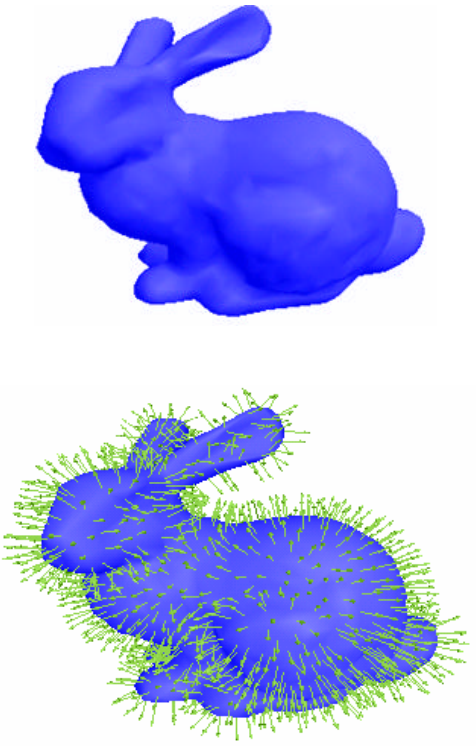
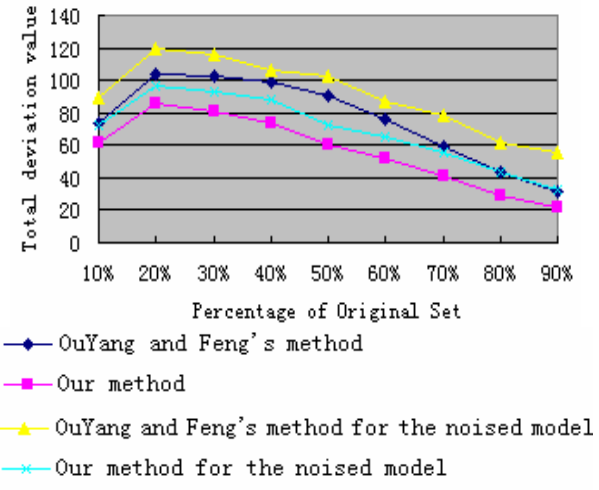


Fig. 4. Bunny model and its normal vector model

For practical mesh models composed of a point cloud set, the true normal vector at each point is unknown, so direct evaluation of the normal vector estimation errors was not possible. Nonetheless, we are able to estimate the normal vectors satisfactorily if the point density goes to infinity. Comparison of the methods was thus made according to the consistency in normal vector estimation at reduced point densities for each method [5].

TABLE 1
COMPARISON OF OUR NORMAL VECTOR ESTIMATION WITH OUYANG AND FENG'S METHOD FOR BUNNY MESH MODEL



For the Bunny mesh model, the normal vector n_0 at a point in the original point cloud data set was

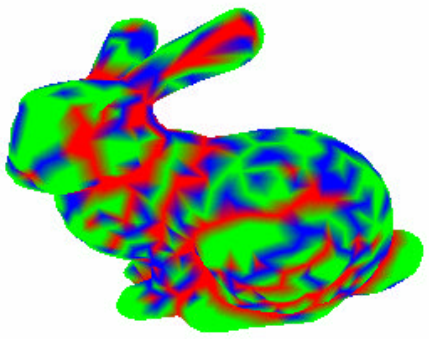
evaluated first by the estimation method and regarded as the reference. Reduced point density was achieved by randomly deleting 10-90% of points from the original set. The normal vector n_i is then estimated again at each point in the reduced point sets. The deviation was quantified as the angle between n_0 and n_i. We compare our method to OuYang and Feng's estimation method [5] which is claimed to be better than other existing methods. From the result comparison, we find our method is more consistent than OuYang and Feng's method, as shown in Table 1.

2.2 *Curvature estmation of the mesh model*

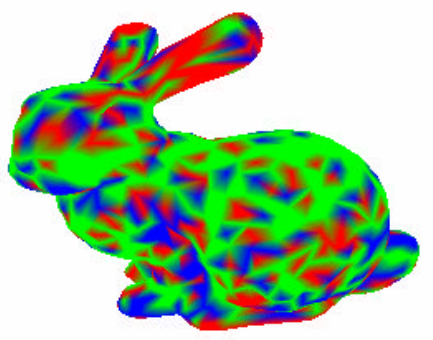
We also use above parametric surface to estimate both mean curvature and Gaussian curvature. For point p_i, representing the point $B(u_i,v_i)$ on the bicubic Bézier surface, after the unit normal vector is computed, we suppose $E = B_u \cdot B_u$, $F = B_u \cdot B_v$, $G = B_v \cdot B_v$, $L = B_{uu} \cdot n$, $M = B_{uv} \cdot n$, $N = B_{vv} \cdot n$. According to the differential geometry formulas, the Gaussian curvature and mean curvature on its point are computed as

$$K = \frac{LN - M^2}{EG - F^2},\ H = \frac{NE - 2MF + LG}{2(EG - F^2)}.$$

The mean curvature and Gaussian curvature in the bunny model are shown in Fig. 5. We compare our curvature estimation result to Razdan and Bae's method [11] which obtains the satisfactory curvature estimation. We also use the same comparison strategy that judges the deviation consistency by reducing the point density. From the result comparison, we find our cubic-order curvature estimation result is more accurate than Razdan and Bae's second-order method, as shown in Table 2.



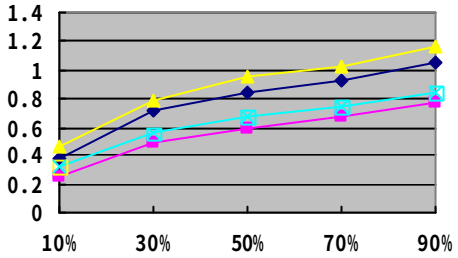
(a)



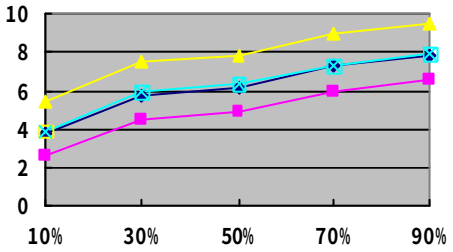
(b)

Fig. 5. Bunny mesh model. (a) Mean curvature model. (b) Gaussian curvature model. (Red: high values; Green: medium values; Blue: low values)

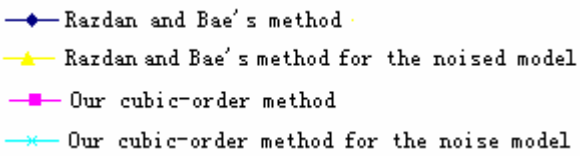
TABLE 2
COMPARISON OF OUR CURVATURE ESTIMATION RESULT WITH RAZDAN AND BAE'S METHOD FOR BUNNY MESH MODEL



(a) Mean curvature comparison



(b) Gaussian curvature comparison



(*x*-axis is the percentage of original set. *y*-axis is the total deviation value)

2.3 *Torsion estimation of themesh model*

Torsion, like curvature, is an important geometric property of the object's rigid transformation. The research on torsion can provide us a more powerful tool for the digital geometric processing, but it seems there is little attention on the torsion topic of the discrete mesh model. For the point on the smooth surface, the torsion is normally measured by the geodesic torsion. Because the curve passing the given point on the surface is not determined, it brings us the difficulty in estimating torsion. We generally compute the geodesic

torsion according to some specific curves passing through the point on the surface. One curve is the curvature line, but we know the geodesic torsion along the curvature line is zero from the differential geometry knowledge. Another curve is the torsion line, namely, the direction along the angle bisector of two principal directions. We know the geodesic torsion attains the maximum along the torsion line on the given point. Here we define it as the principal geodesic torsion. For the discrete mesh model, there is the similar torsion property. In this paper, we introduce the principal geodesic torsion and estimate it by the local fitting patch. For the principal geodesic torsion, we have the following definition and theorem.

Definition. For the point on the surface, the geodesic torsion reaches the maximum value on two orthogonal directions along the angle bisector of two principal directions, we define the largest of geodesic torsion values as the principal geodesic torsion $t_{\max}$.

Theorem. For the principal geodesic torsion on the point of the surface, its value can be calculated by the mean curvature H and Gaussian curvature K, i.e., $t_{\max} = \sqrt{H^2 - K}$.

Proof. Supposing q is the angle between the direction composed of $du : dv$ and the curvature line whose v value is constant. From the differential geometry formula, we know the geodesic torsion is

$$t = \frac{1}{2}(k_2 - k_1)\sin 2q ,$$

where k_2, k_1 are principal curvatures respectively and $k_2 \geq k_1$.

When the direction is along the angle bisector of two principal directions, namely, $q = \pm p/4$, the geodesic torsion value reaches the maximum value. So we have

$$t = t_{\max} \sin 2q .$$

From the Euler formula, we know

$$(k_n - k_2)\cdot(k_n - k_1)$$
$$= (k_1 - k_2)\cos^2 q \cdot (k_2 - k_1)\sin^2 q = -t^2 .$$

Hence,

$$t^2 = -(k_n - k_2)\cdot(k_n - k_1) = -k_n^2 + 2Hk_n - K .$$

This equation is understood that t is a quadratic function of k_n. Obviously, when $k_n = H$, t gets the maximum value, namely, $t_{\max} = \sqrt{H^2 - K}$.

After estimating the mean curvature and Gaussian curvature through the local fitting patch, we estimate the principal geodesic torsion from above theorem for each vertex of the mesh model. The principal geodesic torsion can be used to describe the torsion property of each vertex on the mesh model, the principle geodesic torsion model of the bunny model is shown in Fig. 6.

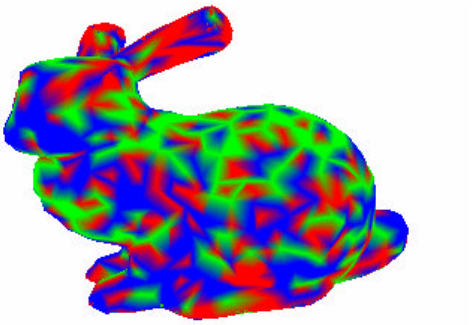
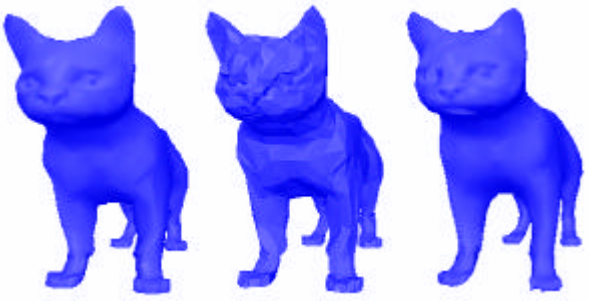


Fig. 6. The principal geodesic torsion model of bunny mesh model. (Red: high values; Blue: medium values; Green: low values)

3. PATCH APPLICATION FOR SMOOTHING THE NOISED MESH MODEL

We apply this third-order local fitting patch for the noised mesh smoothing. Because the weighted factor is set according to the noise and we add the smoothing equations in the new third-order fitting patch, the point on the fitting surface can be regarded as the smoothed position for the original vertex in the mesh model.

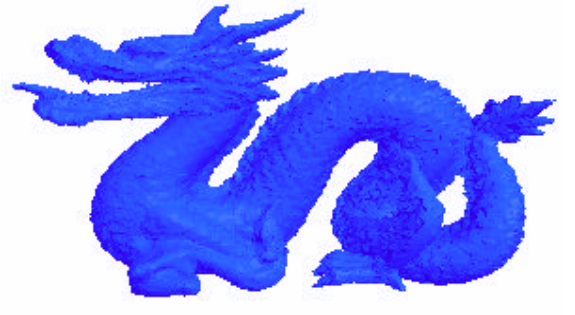
We use some mesh models to test our mesh smoothing algorithm. Fig. 7(a) is a simplified cat mesh model. Fig. 7(b) is the noised mesh model. Fig. 7(c) is our smoothing result. To our surprise, we find our smoothing result can restore the original shape such as the lips. We use Desbrun's method [16] to measure the ratio of the original volume and the volume after the model is smoothed. The ratio of our method is 0.9936. This shows our smoothing method efficiently prevents the volume shrinkage. Fig. 8(a) is a original dragon mesh model, Fig. 8(b) is its noised mesh model. Fig. 8(c) is Ohtake et al's denoising result [17]. Fig. 8(d) is our smoothing result. We find our smoothing algorithm is applicable to the large mesh model and has the same feature-preserving advantage as Ohtake's denoising method.



(a) (b) (c)

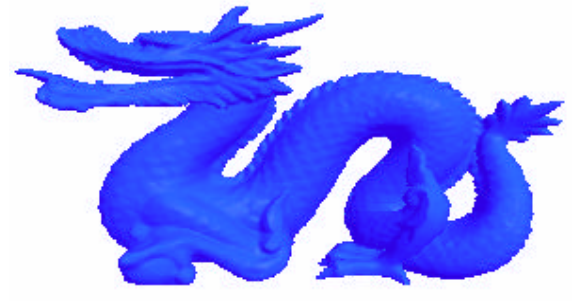
Fig. 7. Cat mesh model. (a). Simplified mesh model (b). Noised mesh model (c). Our smoothing result

(a) Original mesh model



(b) Noised mesh model

(c) Ohtake's denoising result



(d) Our smoothing result

Fig. 8. Dragon mesh model

4. PATCH APPLICATION FOR THE HOLE-FILLING OF THE MESH MODEL

This third-order fitting patch can also be used to fill the holes of a mesh model. We firstly find the influencing area of each hole and set vertices in the influencing area as the fitting points. Secondly we project these vertices onto a tangent plane to obtain the parameters of fitting points in $[0,1]^2$. Then we construct the weighted bicubic Bé zier patch to approximate the hole. Finally we use the Delaunay triangulation algorithm to acquire the triangulation of a closed polygon composed of border vertices in the parameter field of the tangent plane. The triangles in the tangent plane are matched to the 3D triangles approximating the third-order Bé zier patch, which are used to fill the hole in the mesh model. For example, there are four holes in the bottom of bunny mesh model. We construct four third-order fitting patches to fill them respectively. The hole-filling result is shown in Fig. 9.

If the hole has a complex shape, we need to use the feature extraction [18] and the curve blending technique to obtain the completed feature curves in the hole in advance. These feature curves divide the hole into several simple sub-holes, then we use the third-order bicubic Bé zier patch to fill each sub-hole. For example, there is a hole in the fandisk model in Fig. 10(a), we first construct two completed feature curves in the hole to separate it into three sub-holes, as shown in Fig. 10(b), then we use three fitting patches to fill them, the hole-filling result is shown in Fig. 10(c). The original mesh model is shown in Fig. 10(d). We find our hole-filling method can efficiently restore the feature part of the object.

One problem we need to notice is that the current filling surface $B(u,v)$ is an approximating surface by the least squares method, which may not pass through all border vertices of the hole. We can modify it by the Lagrange interpolating method. Supposing the border vertex of the hole is v_i and there are n border vertices, the new filling surface is written as

$$B'(u,v) = B(u,v) + \sum_{k=0}^{n} d_i \cdot l_k(u,v), \; u,v \in [0,1],$$

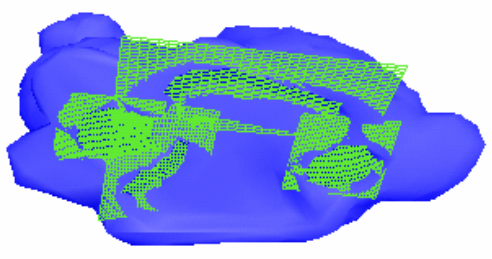
where $B(u,v)$ is the previous Bé zier patch,

$$d_i = v_i - B(u_i, v_i),$$

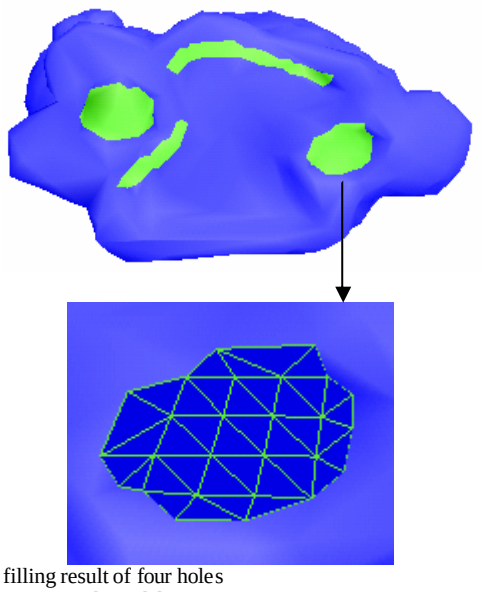
$$l_k(u,v) = \prod_{i=0, i\neq k}^{n} \frac{(u-u_i)(v-v_i)}{(u_k-u_i)(v_k-v_i)}.$$

If Lagrange interpolation with high degree causes numerical instability problem, we uniformly divide the parameter u,v space into small subspaces, each of which contains fewer interpolating points. Then we use $B'(u,v)$ as the combination of the Bé zier patch and the interpolating surface with low degree to replace the fitting surface $B(u,v)$. The original vertices of the 3D triangles filling the hole which come from $B(u,v)$, are now replaced by corresponding points on each modified surface $B'(u,v)$. This measure makes sure the filling 3D triangles pass through all border vertices of the mesh hole.

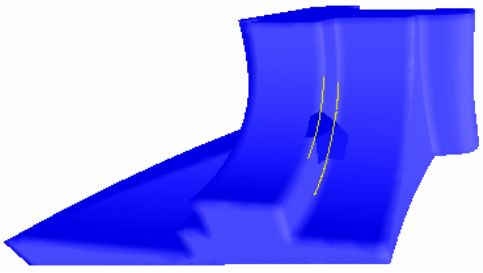
(a) Four holes in the mesh model



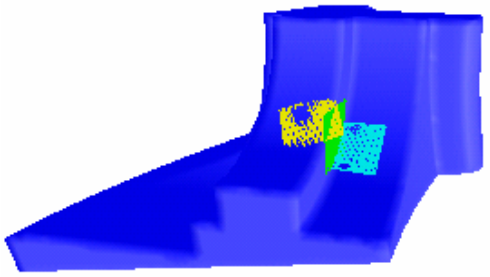
(b) Four hole-filling patches



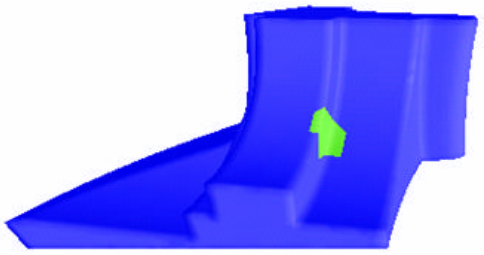
(c) The filling result of four holes
Fig. 9. Bunny mesh model



(a) Constructing two feature curves to divide the hole into three sub-holes



(b) Using three third-order fitting patches to fill three sub-holes



(c) Our hold-filling result

(d) The original mesh model
Fig. 10. Fandisk mesh model

5. CONCLUSION AND FUTURE WORK

In this paper, we provide an effective third-order local fitting patch. Compared to other algorithms, this weighted bicubic Bé zier patch more accurately obtains the normal vector and curvature estimation of the mesh model. Furthermore, we define the principal torsion of each vertex in the mesh model and estimate it by this third-order local fitting patch. As for the application, we apply this third-order local fitting patch for the mesh smoothing and hole-filling. The experiments show that it obtains satisfactory results.

Our study has some limitations, which suggest topics for our future work. If there are no enough vertices in the 2-ring neighborhood, we need to look for other vertices in the 3-ring neighborhood for the patch construction, but the size of the neighborhood will influence the stability of our algorithm. The tangent plane is built based on the approximate normal, which may be not accurate. The factor a which reflects the noise density currently is chosen by the subjective judgment. In the mesh smoothing and hole-filling application, we need to compare our method to other popular algorithms.

ACKNOWLEDGEMENTS

The authors gratefully appreciate the anonymous referees for the useful comments and suggestions which improved this paper considerably. This research was supported by the Open Project Program of the State Key Lab of CAD&CG (Grant No.A0913), Zhejiang University; the Foundation for Person with Ability of the Ministry of Education Key Lab of Advanced Textile Materials & Manufacturing Technology (Grant No.2008QN01), Zhejiang Sci-Tech University; the National Natural Science Foundation of China (Grant No.60533080).

REFERENCES

[1] Huang J, Meng CH. Automatic data segmentation for geometric feature extraction from unorganized 3-D coordinate points. IEEE Trans Robotics Automation. 2001, 17(3):268-279.

[2] Yang M, Lee E. Segmentation of measured point data using a parametric quadric surface approximation. Computer-Aided Design. 1999, 31(7):449-457.

[3] Milroy MJ, Bradley C, Vickers GW. Segmentation of a wrap-around model using an active contour. Computer-Aided Design. 1997, 29(4):299-320.

[4] Woo H, Kang E, Wang S, et al. A new segmentation method for point cloud data. The International Journal of Machine Tools and Manufacture. 2002, 42(2):167-178.

[5] OuYang D, Feng H. On the normal vector estimation for point cloud data from smooth surfaces. Computer-Aided Design. 2005, 37(10):1071-1079.

[6] Meyer M, Desbrun M, Schroder P, et al. Discrete differential geometry operators for triangulated 2-manifolds. In: Hege H-C Poltheir K, editors. Visualization and mathematics, vol. 3. Berlin, Springer, 2002:34-57.

[7] Taubin G. Estimating the tensor of curvature of a surface from a polyhedra approximation. ICCV 1995:902-907.

[8] Surazhsky T, Magid E, Soldea O, et al. A comparison of Gaussian and mean curvatures estimation methods on triangular meshes. 2003 IEEE international conference on robotics and automation, Taipei, Taiwan, 14-19 Sep, 2003:1021-1026.

[9] Goldfeather J, Interrante V. A novel cubic-order algorithm for approximating principal direction vectors. ACM Transaction on Graphics. 2004, 23(1):45-63.

[10] Rusinkiewicz S. Estimation of curvatures and their derivatives on triangle meshes. Second international symposium on 3D data processing. 2004:486-493.

[11] Razdan A, Bae M. Curvature estimation scheme for triangle meshes using biquadratic Bé zier patches. Computer-Aided Design. 2005, 37(14):1481-1491.

[12] Faux I, Pratt M. Computational geometry for design and manufacture. New York, Halsted Press, 1979.

[13] Lewiner T, Gomes J, Lopes H, Craizer M. Curvature and torsion estimators based on parametric curve fitting. Computers & Graphics. 2005, 29(5):641-655.

[14] Yuen P, Mokhtarian F, Khalili N, Illingworth J. Curvature and torsion feature extraction from free-form 3D meshes at multiple scales. Proceeding of VISP, 2000, 147(5):454-462.

[15] Troyani N, Perez A, Baiz P. Effect of finite element mesh orientation on solution accuracy for torsional problems. Finite elements in analysis and design. 2005, 41(14):1377-1383.

[16] Desbrun M, Meyer M, Schroder P, Barr A: Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proceedings of SIGGRAPH. 1999:317-324.

[17] Ohtake Y, Belyaev A, Alexa M. Sparse low-degree implicit surfaces with applications to high quality rendering, feature extraction, and smoothing. Eurographics Symposium on Geometry Processing. 2005:149-158.

[18] Stylianou G, Farin G. Crest lines for surface segmentation and flattening. IEEE Transactions on Visualization and Computer Graphics. 2004, 10(5):536-544.