Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

Smooth feature line detection for meshes*

GUO Yan-wen (郭延文)†[1,2], PENG Qun-sheng (彭群生)[1,2], HU Guo-fei (胡国飞)[1], WANG Jin (王 进)[1]

([1]*State Key Laboratory of CAD & CG,* [2]*Department of Mathematics, Zhejiang University, Hangzhou 310027, China*)

†E-mail: ywguo@cad.zju.edu.cn

Received Aug. 12, 2004; revision accepted Nov. 1, 2004

Abstract: In this paper the authors present a novel semi-automatic feature line detection technique for meshes. Taking into account the distance and orientation between two vertices on meshes and the curvature information of vertices, they first find an initial feature line which connects some user-specified vertices on meshes; then parameterize the "feature strip" surrounding the feature line onto a planar domain using a vertex flattening technique; and refine the flattened feature strip using the 2D snakes approach to make the feature line smoother and more accurate; lastly they get the feature line by mapping the refined line back to the original meshes. Experimental results showed that their method can extract the feature line rapidly and precisely. As an application, they propose a mesh decomposition method based on the detected feature line.

Key words: Feature line, Image snakes, Mesh parameterization, Mesh decomposition
doi:10.1631/jzus.2005.A0460 **Document code:** A **CLC number:** TP390

INTRODUCTION

With the rapid development of graphical hardware, triangular mesh has become one of the most popular representations for 3D objects in computer graphics and virtual reality. Many techniques have been developed such as mesh simplification, mesh editing, mesh signal processing and parameterization, etc., to process the triangular mesh.

Feature detection is a key technique in mesh processing. In general, features include ridges, ravines, corners and boundaries, etc., which can provide us important shape information and frequently require special treatment to achieve high accuracy and reliability. Feature detection serves as a pre-processing technique which seeks and extracts features for various applications such as feature-preserving mesh simplification, 3D morphing ensuring rigid correspondences between the features of meshes, and mesh enrichment striving to smooth out roughness of meshes.

Although various feature detection algorithms have been proposed during the past several years (Khaneja *et al*., 1998; Belyaev *et al*., 2000; Hubeli and Gross, 2001; Andrews, 2002; Page *et al*., 2002; Ohtake and Belyaev, 2001; Ohtake *et al*., 2004; Rössl *et al*., 2000; Lee and Lee, 2002), it is still a big challenge to find smooth and meaningful features on 3D geometry. Most of the previous methods mainly focused on global feature detection via curvature analysis, but paid less attention to conveniently extracting and controlling local feature regions. Moreover, few previous methods can precisely extract smooth feature line; which is important to mesh processing. For instance, mesh decomposition must segment model along a smooth feature line; mesh optimization should rely on the smooth feature line to improve mesh quality.

This paper provides a semi-automatic feature line extraction method for meshes. The method can detect the smooth and accurate feature line effectively. A mesh decomposition method is given as an appli-

*Project supported by the National Natural Science Foundation of China (Nos. 60403038, 60033010) and the National Basic Research Program (973) of China (No. 2002CB312101)

cation example.

RELATED WORK

Feature detection for meshes

Previous feature detection methods (Belyaev *et al*., 2000; Ohtake and Belyaev, 2001; Rössl *et al*., 2000; Page *et al*., 2002) are mainly based on curvature analysis. The feature regions are distinguished from the other part of meshes by a classification of the curvature. Usually these methods detect automatically all the feature regions on the mesh once. However, they cannot easily extract the local feature precisely and the result is not ready to be used in other mesh processing applications.

Some researchers with knowledge on image processing focus on the interactive feature detection method. Khaneja *et al*.(1998) provided a dynamic programming algorithm for extracting globally optimal geodesic curves on human brain surfaces. Andrews (2002) presented a feature detection technique for the 3D triangular meshes which is an extension of the 2D image processing technique; the input of his method is the parameterization result of the surface and feature curve is detected through the minimal path approach. But feature curve detected by this technique is uniquely determined between the specified source and destination points and cannot be controlled by the user, no wonder that the curve sometimes may fail to capture a desired feature.

Lee and Lee (2002) proposed a method called geometric snakes which is an extension of the image snakes. First the user specifies an initial position on meshes through the geodesic line connecting the source vertex and destination vertex, then the geometric snake automatically slithers to a nearby feature by minimizing an energy function. The geodesic line is so far the most useful curves on meshes; it is the shortest path on meshes connecting two vertices specified by the user, i.e. the starting vertex and the ending vertex (Kimmel and Sethian, 1998; Kanai and Suzuki, 2000). But the geodesic line cannot capture the feature on the mesh generally, which will affect the stability and convergence of the snake.

Hubeli and Gross (2001) presented a semi-automatic method to extract the salient features of surfaces. Given an input mesh, its progressive mesh representation is constructed and the features are detected on the coarse mesh first; then the input mesh is reconstructed from the coarse mesh and the features are refreshed progressively. The main drawback of this method is that it needs to tune a few parameters for optimal performance. In addition, the simplification and reconstruction of mesh are usually time-consuming for large meshes.

A recent work briefly touches on Ohtake *et al*.(2004)'s proposed simple and effective method for detecting ridge-valley lines defined via first- and second-order curvature derivatives on meshes. The high-order surface derivative is achieved by combing multi-level implicit surface fitting and finite difference approximations. One limitation of their method is the low speed for computing the approximation of the curvature derivatives on globally defined surface.

Image snakes approach

Image snakes, namely, active contour, were first brought out by Kass *et al*.(1998) and widely studied by many researchers (Cohen, 1991; Xu and Prince, 1998). It is an important technique in image processing and can be used for image segmentation, tracking, registration, etc.

An active contour is a closed or open curve that moves towards the shape feature line in a way controlled by minimizing an energy function E_{snake} usually including the internal forces E_{int}, such as rigidity and elasticity of the curve, and the image force E_{image} that attracts the contour to certain features, such as edges in the image. This is done by creating an attractor image, which defines how strongly each point in the image attracts the contour.

The contour can be parametrically defined as $v(s)=[x(s), y(s)]$, where $s\in[0,1]$ is the normalized arc length along the contour, then the total energy to be minimized may be written as:

$$\begin{aligned} E^*_{\text{snake}} &= \int_0^1 E_{\text{snake}}(v(s))\mathrm{d}s \\ &= \int_0^1 (E_{\text{int}}(v(s)) + E_{\text{image}}(v(s)))\mathrm{d}s \end{aligned} \tag{1}$$

We use $\alpha(s)$, $\beta(s)$ to represent the elasticity and rigidity of the snakes, $\nabla I(x,y)$ represents image gradient at pixel (x, y). So the internal curve energy:

$$E_{\text{int}} = \alpha(s)\,|\,v'(s)\,|^2 + \beta(s)\,|\,v''(s)\,|^2 \tag{2}$$

while the image force has the form of:

$$E_{\text{image}} = -|\nabla I(x,y)|^2 \quad (3)$$

Minimizing the energy function Eq.(1) is equivalent to solving the corresponding Euler-Lagrange equation:

$$\alpha(s)v''(s) - \beta(s)v^{(4)}(s) = -\nabla(|\nabla I(x,y)|^2) \quad (4)$$

.

In practice one does not study the contour at all points. Instead, the contour $v(s)$ must be discretized for numerical recipe, thus $v(i)=(x(i), y(i))$, $i=1,\ldots,n$. At each point $v(i)$, thus derivatives in Eq.(4) can be approximated by finite differences:

$$v''(i)=v(i-1)-2v(i)+v(i+1) \quad (5)$$

$$v^{(4)}(i)=v(i-2)-4v(i-1)+6v(i)-4v(i+1)+v(i+2) \quad (6)$$

Using above expressions, Eq.(4) can be rewritten as:

$$\boldsymbol{A}v(s)+\nabla(|\nabla \boldsymbol{I}(x,y)|^2)=0 \quad (7)$$

where $\boldsymbol{A}$ is the matrix given by Eqs.(5) and (6) and insertion of $\alpha(s)$ and $\beta(s)$. A contour can be calculated with iterations according to Euler's method as:

$$(\boldsymbol{I}+\boldsymbol{A})v_{k+1}(s)=v_k(s)+\nabla(|\nabla \boldsymbol{I}(x,y)|^2) \quad (8)$$

with $v_0(s)$ being the initial contour specified by user.

After every iteration, the contour moves so far in the direction of the image force and captures the feature at last.

FEATURE LINE DETECTION ALGORITHM

The proposed method is semi-automatic. With several key vertices specified by the user near the feature region, we get an initial guess of the feature line by connecting these vertices. However, this feature line consisting of the irregular edges of the meshes is not smooth and accurate enough. To extract a smooth feature line, we adopt the snakes' approach which is now commonly used in image processing. Nevertheless, it generally deals with 2D situation. So we first parameterize the "feature strip" surrounding the feature line to a planar domain, then apply image snakes to the planar feature strip. The stages of the whole algorithms are described as follows (Fig.1):

1. Generation of initial feature line. We interactively select several key vertices, and detect feature lines with a "marching feature search" method. A refinement will be done later to make feature lines smoother.

2. Parameterization. Map the initial feature line onto a planar domain, and then parameterize the whole feature strip surrounding the initial feature line onto the planar region.

3. Refinement with snakes. Refine the embedded feature line to be smoother and more accurate by the snakes approach.

4. Feature line extraction. Map the refined line back to the original meshes to get the final feature line.

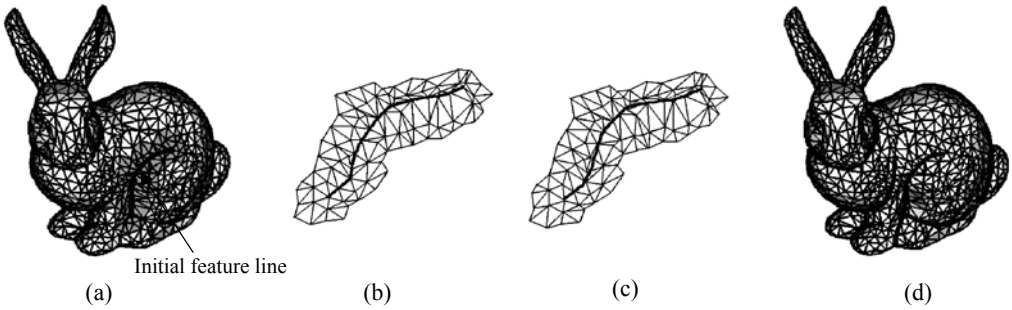


Fig.1 The whole process of our algorithm

(a) Create an initial feature line (the black line) connecting two selected vertices (the black dots) on a bunny; (b) Map the feature line to the planar region and parameterize the feature strip; (c) Refine the feature line using snakes approach; (d) Then finally detected smooth feature lines by mapping the refined feature lines back to the original meshes

Notations

For convenience, we introduce the following notations (Fig.2):

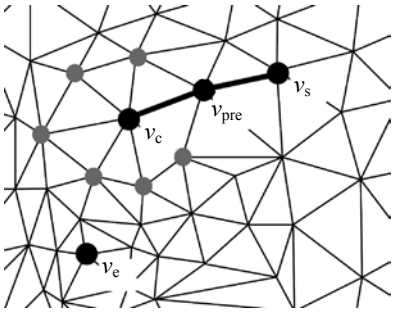


Fig.2 Some notations

v_s, v_e: the specified starting and ending vertex; the black line: the detected feature line; v_c: the current vertex; v_{pre}: the previous vertex relative to v_c on the feature line; V_a includes the gray vertices and v_{pre} which are also the 1-ring of v_c simultaneously

The feature line denoted by L connects the vertices set $P=\{v_{pi}|i=1,\ldots,n\}$; the edge denoted by $e_{i,i+1}$ connects v_{pi} with $v_{p(i+1)}$; the feature strip denoted by S is the local meshes strip surrounding L; the starting vertex and the ending vertex specified by the user are denoted by v_s and v_e respectively; the current vertex is v_c and its topologically adjacent vertices are $V_a=\{v_1,v_2,\ldots,v_n\}$; the previous vertex of v_c on the feature line is denoted by v_{pre}; the successive vertex of v_c is v_{suc}.

C_v represents the mean curvature of vertex v; we use the formula presented by Borrelli (1993) to compute the mean curvature of a vertex on the meshes; $\boldsymbol{V}_{i\_j}$ represents the vector from vertex v_i to v_j, particularly, the vector from v_s to v_e is $\boldsymbol{V}_{s\_e}$.

Topological distance between two vertices v_i, v_j is the minimal number of edges connecting v_i and v_j; k-ring of v_i are those vertices whose topological distance to v_i is k; k-ring meshes of v_i are the meshes which are surrounded by the k-ring of v_i; k-ring meshes of vertices set P are the combination meshes of k-ring meshes of every vertex in the set P.

Generation of initial feature line

Given a starting vertex v_s and an ending vertex v_e, it is time-consuming and not feature preserving to simply connect v_s and v_e with geodesic line. In this paper, we take into account orientation, distance and curvature information of the vertices on the mesh, and detect initial feature using a marching search scheme, which invokes actually local optimization. This scheme begins from the starting vertex v_s, finds its successive vertex using an aim function; then regards the new vertex as current vertex, continues to find v_{suc} iteratively; ends up until the ending vertex v_e is found. Now, we introduce the method of finding the successive vertex v_{suc} of the current vertex v_c.

v_{suc} should be chosen from V_a of v_c. If v_e is included in V_a, then we think that v_e is just v_{suc}. Otherwise for every vertex $v_i \in V_a$, we consider four cost functions: two orientation functions: f_{O1}, f_{O2}; a distance function: f_D; a curvature function: f_C; and one aim function F. The meanings of these functions are listed below:

$f_{O1}(v_i)$: inverse proportion to the angle between $\boldsymbol{V}_{c\_i}$ and $\boldsymbol{V}_{s\_e}$; $f_{O2}(v_i)$: inverse proportion to the angle between $\boldsymbol{V}_{c\_i}$ and $\boldsymbol{V}_{pre\_c}$; $f_D(v_i)$: inverse proportion to the Euclidian distance from v_i to v_e; $f_C(v_i)$: inverse proportion to $|C_{vi}-C_{vc}|$; $F(v_i)$: the sum of $f_{O1}(v_i)$,$f_{O2}(v_i)$, $f_D(v_i)$ and $f_C(v_i)$, namely,

$$F(v_i)=f_{O1}(v_i)+f_{O2}(v_i)+f_D(v_i)+f_C(v_i) \quad (9)$$

f_{O1} along with f_D guarantee that the feature line will move along the direction of $\boldsymbol{V}_{s\_e}$ approximately and reach v_e finally; f_{O2} ensures the smoothness of the line; in general the features are located in high-curvature regions and v_e is a selected vertex with high curvature, every vertex on the feature line should have high-curvature similar to v_e; the curvature function f_C has this effect and guarantees the feature line across the feature region. The cost functions are defined as discrete functions, for example, if the angle between $\boldsymbol{V}_{c\_i}$ and $\boldsymbol{V}_{s\_e}$ is minimum, $f_{O1}(v_i)$ is assigned to n (the number of A_a); otherwise if the angle is maximum, $f_{O1}(v_i)$ is assigned to 1.

Compute $F(v_i)$ for each vertex v_i included in V_a. The vertex with the maximal F is chosen as the successive vertex of the current vertex on feature line. In fact, the effects of these cost functions are not always the same. For example, the curvature function f_C should have larger effect in the aim function F than other functions because the vertices on the feature boundary posses generally high curvatures. For reflecting the difference among their effects, we give them different weights: α, β, γ, δ and evaluate them according to their effects. Thus, the aim function is

rewritten as:

$$F = \alpha f_{O1} + \beta f_{O2} + \gamma f_D + \delta f_C \quad (10)$$

According to our experiments, α, β, γ and δ selected in proportion with 1, 1, 1 and 2 will achieve satisfactory initial feature line results.

The algorithm of the marching search scheme is summarized below:

1. Specify two vertices on the feature line to be detected: one is regarded as the starting vertex v_s, the other is regarded as the ending vertex v_e;

2. Set the starting vertex as current vertex: v_c=v_s;

3. Get the topological adjacent vertices V_a of the current vertex v_{suc}, check if v_e belongs to V_a, if it is, then end up. Otherwise compute the aim function F for every vertex $v_i \in V_a$;

4. Choose the vertex with the maximal value of F as the next vertex v_{suc} on the feature line;

5. v_c=v_{suc}; Go to Step 3;

To get a complete feature line, we may need to specify more than two key vertices. In this case, the selects its next one as v_e, uses the above algorithm to generate the feature line between two vertices. The complete feature line is obtained by connecting all of them.

Fig.3 gives the example of the feature lines detected by the above algorithm and its comparison with the geodesic line.

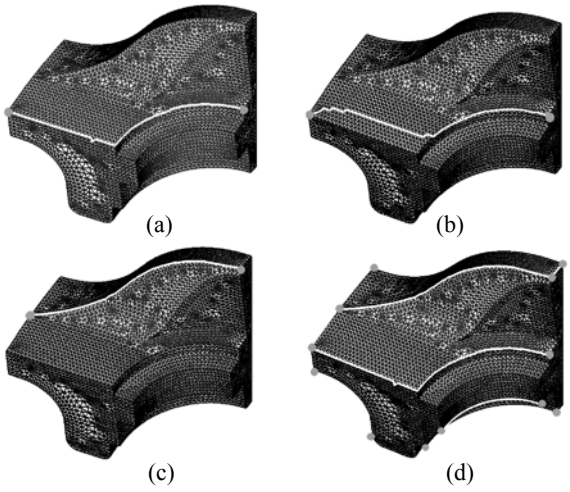


Fig.3 Compare initial feature line with the geodesic line
(a) The feature line connecting two specified vertices (the grey dots) created using our method in Section 3; (b) The geodesic line connecting the two vertices; (c), (d) Another two created feature lines using our method.

Parameterization of feature strip

As discussed above, the feature strip is the local mesh strip surrounding the feature line; Fig.4 shows the feature strip of a feature line, which in fact is k-ring meshes of the vertices on the feature line. When the mesh density near the feature line is sparse, we select a smaller k value; otherwise we select a larger one. The topological structure of feature strip is long and narrow. The initial feature line can be considered as a medial axis of the feature strip. Hence we can first map the initial feature line onto a planar region, and then flatten progressively the triangles in the feather strip surrounding the feature line onto the plane.

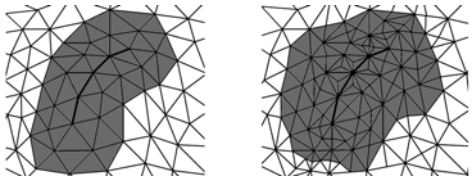


Fig.4 Feature strip
The black line: initial feature line; the grey meshes: feature strip; (a) we choose 2-ring meshes as the feature strip, for sparse mesh density; (b) 3-ring meshes is chosen as the feature strip for high mesh density

1. Embedding the initial feature line. First we map the edge $e_{1,2}$ (see the notation) onto the plane, thus v_{p1}, v_{p2} are projected to the planar point p_1 and p_2 accordingly; then it is easy to find a planar point p_3 satisfying two conditions: one is that the distance between p_2 and p_3 is equal to the length of $e_{2,3}$, the other is that the angle consisting of p_1, p_2 and p_3 is equal to the angle between $e_{1,2}$ and $e_{2,3}$.

Iteratively, we can embed the vertices in the feature line into the plane. Suppose the corresponding 2D vertices are

$$P^* = \{p_i | i=1,\ldots,n\},\ p_i \text{ corresponds to } v_{pi}. \quad (11)$$

2. Flattening feature strip. For any vertex v_i on the 1-ring meshes of the feature line, if it connects with two vertices v_j, v_k on the feature line and the three vertices compose a triangle on the original mesh, then we assume that v_j, v_k correspond to planar points p_j and p_k. It is easy to find a point p_i to make the triangle $\{p_i, p_j, p_k\}$ similar to the triangle $\{v_i, v_j, v_k\}$. p_i is the

projected point of v_i. Otherwise v_i connects with three or more vertices on the feature line, v_i composes two or more triangles with the vertices on the feature line. For every triangle, v_i corresponds to a point on the plane according to the above conditions. We can get the projected point of v_i by averaging all these points (Fig.5).

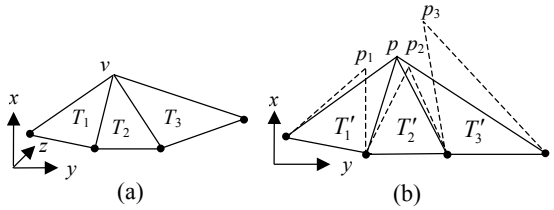


Fig.5 Flattening of the vertex *v*

The black vertices in (a) have been flattened to the black points in (b). p_1, p_2 and p_3 in (b) are the points satisfying that T_1', T_2' and T_3' are conformal to T_1, T_2 and T_3 respectively. We get p the flattened point of v by averaging p_1, p_2 and p_3

For the vertices that connect with the feature line through two edges, there must be an edge directly connecting them and the 1-ring meshes, which have been already flattened onto the plane. So we can use the above method to flatten these vertices. In the same way, the projections of other vertices can be found progressively.

When all the vertices of the feature strip are projected onto the planar region, the meshes of the strip are parameterized at the same time. It is easy to see that the complexity of this algorithm is $O(n)$, so it is efficient. One problem of this parameterization algorithm is that it may lead to self-intersections of a feature strip if the initial feature line is highly curved. But this situation seldom occurs, since the wandering feature line can be segmented into several parts to be detected. So it is not really a drawback of this algorithm as a special parameterization for the feature strip with such special topological structure (Fig.6 gives two examples of parameterization).

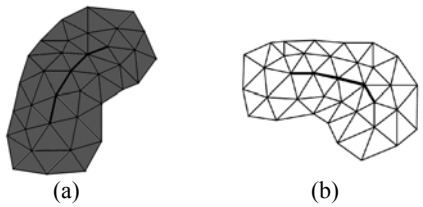


Fig.6 The parameterization result of the feature strip in Figs. 4a and 4b. The black line is the mapped feature line

(a) The feature strip; (b) Its parameterization result

Refinement with the snakes

In this section we describe the technique of refining the mapped feature line P^* by the snakes approach. Assume the arc length parametric form of P^* is $v(s)=[x(s), y(s)]$, where $s\in[0,1]$ is the arc length parameter along the feature line. To process P^* by the snakes approach, two energy functions, namely E_{int} and E_{image}, must be defined (Section 2.2).

The internal function E_{int} relates with the elasticity and rigidity of P^*. This conforms to the image snakes. The form of E_{int} is:

$$E_{\text{int}} = \alpha(s)\,|\,v'(s)\,|^2 + \beta(s)\,|\,v''(s)\,|^2 \tag{12}$$

where $\alpha(s), \beta(s)$ represent the weights of elasticity and rigidity respectively and are usually valued constants according to different requirements.

In image snakes, E_{image} attracts the snake to the feature of the image. The expression of E_{image} is generally based on the image gradient. The parameterized feature strip can be seen to be the same as the image plane, however, the meshes have no concept of gradient. To solve this problem, we substitute the curvature information of the meshes for the image gradient. For every vertex of the meshes, we compute its mean curvature C. The interior curvature of the triangle is evaluated by using barycentric coordinates interpolation of the three vertices' curvatures. We define the function E_{feature} which corresponds to the image force function, it has the form of:

$$E_{\text{feature}} = -|C(x,y)| \tag{13}$$

So the total energy function to be minimized is:

$$E^*_{\text{snake}} = \int_0^1 (E_{\text{int}}(v(s)) + E_{\text{feature}}(v(s)))\mathrm{d}s \tag{14}$$

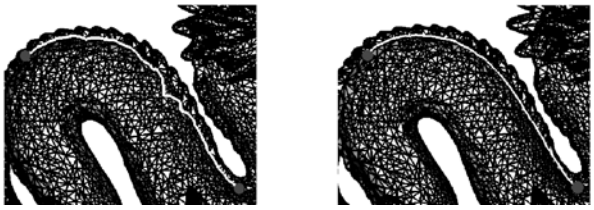
$v(s)$ is discretized into: $v(i)=(x(i),y(i))$. Using the method provided in Section 2.2 we can get the following equation:

$$(\boldsymbol{I}+\boldsymbol{A})v_{k+1}(s)=v_k(s)+\nabla(|C(x,y)|)^2 \tag{15}$$

where $\boldsymbol{I}$ is the identity matrix; $\boldsymbol{A}$ has the same shape with the matrix $\boldsymbol{A}$ in Eq.(7); $v_0(s)$ is the mapped path P^*, i.e. $v(i)=p(i)$, $i=1, \ldots, n$.

We solve the above equation iteratively. After every iteration, the feature line moves towards the place with high curvature and maintains the smoothness simultaneously. In this process if two discrete points on the feature line are far away enough, points are automatically added between the two points; on the contrary if two points are too close, one point is deleted. So the dimension of Eq.(15) is not invariable. The termination condition of iteration is that the iterative number has reached an upper limit or the points' total displacement is low enough.

Thus we get a smooth feature line on the parameterized feature strip. Mapping this line back to the original mesh model, we can have the smooth feature line. Fig.7 gives an example of the contrast between the initial feature line and the final feature line. It is obvious that the final detected feature line is smoother and more accurate.



(a) (b)

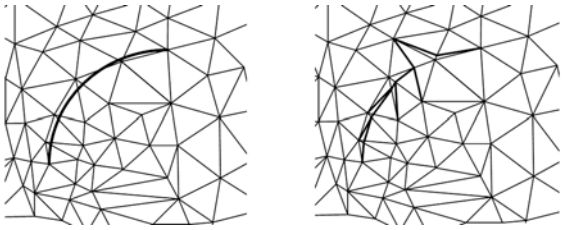
Fig.7 The contrast between the initial feature line and the feature line refined by the snakes
(a) The white lines: initial feature line; (b) The white line: the final feature line refined by the snakes approach

APPLICATION

The detected smooth feature line can be used in various applications: simplification of meshes without feature preserving will have poor quality; a smooth feature line will help local mesh editing and mesh signal processing. Here we provide a mesh model decomposition algorithm based on the extracted feature line as an application.

Mesh decomposition partitions a given mesh into meaningful components, which generally means to segment at regions of deep concavities or high convexity. That is to say, the segmentation line, i.e. the boundary between two components, is the feature line generally.

So the user selects several key vertices near the boundary between two components to be segmented first; then a smooth feature line can be found by employing our feature line extraction method; the vertices nearest to the obtained feature line are finally chosen as the segmentation line. Furthermore, if a smoother segmentation line is needed, we can resample the feature region to generate new vertices along the detected feature line. Then a smoother line passing through the resampled vertices is easily obtained. Fig.8 gives an example of remeshing along the feature line.



(a) (b)

Fig.8 Remesh along the detected smooth feature line
(a) A segment with the detected smooth feature line on the Venus which is similar to the Venus model in Fig.12b, the black dots represent the vertices needed to be remeshed; (b) The remeshed result

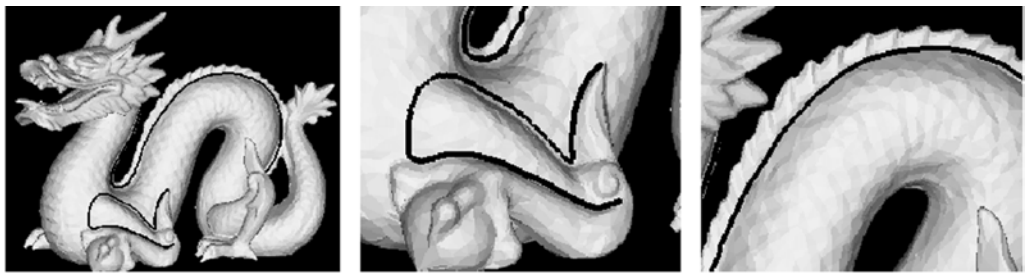
RESULTS

We experimented on several mesh models to test the validity of our algorithm on a 1.6 GHz Pentium IV, 256 RAM PC.

Figs.3a and 3b compare the initial feature line with the geodesic line between two specified vertices. The initial feature line (Fig.3a) found by our algorithm can detect the edges and corners of the fan disk, while the geodesic line (Fig.3b) is only the shortest path connecting the two vertices. Of course, there is still an artifact on the feature line in Fig.3a.

Figs.9~11 gives some feature line detection results. It is obvious that our algorithm can detect the smooth feature line precisely. Fig.12 demonstrates some mesh decomposition examples.

Table 1 gives some experimental data. The 6th~8th columns give the time of initial feature line generation, feature trip parameterization and snakes refinement of the feature region in the 7th column.



(a) (b) (c)

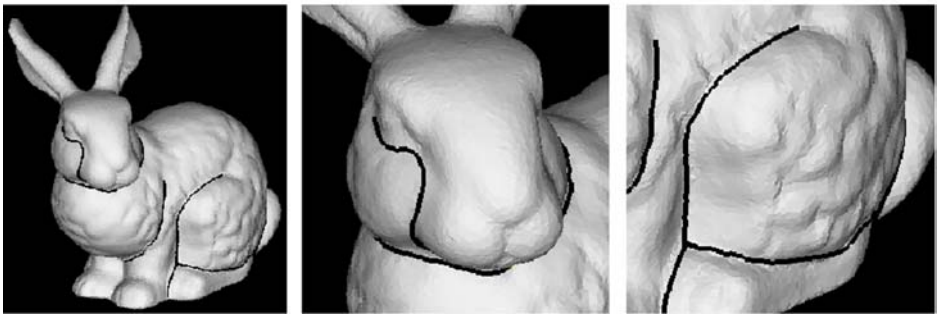
Fig.9 Several feature lines on dragon

(a) The Dragon and its feature lines; (b) The feature line on the wing of Dragon; (c) The feature line on the body

(a) (b) (c)

Fig.10 Several feature lines detected on Buddha

(a) The Buddha and its feature lines; (b) Several feature lines on the half-body of the Buddha; (c) The feature line draws the outline of the Buddha



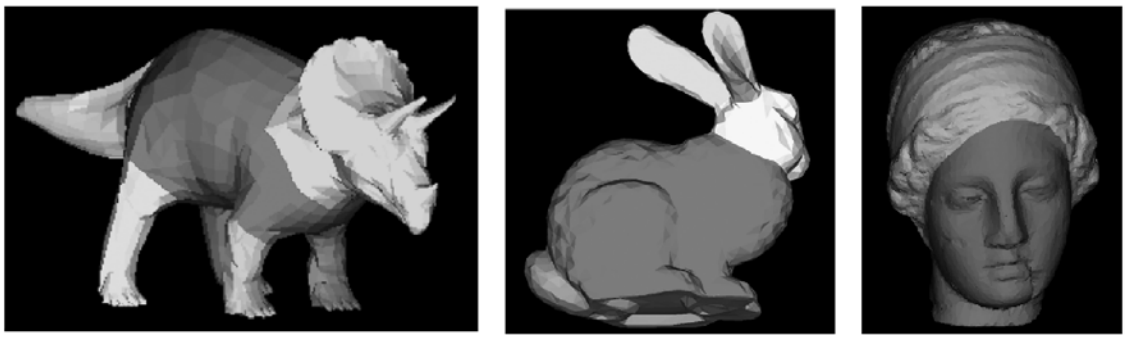
(a) (b) (c)

Fig.11 Feature lines detection result on Bunny

(a) The Bunny and its feature lines; (b) The feature lines on the head and neck of the Bunny; (c) The feature lines on leg and tail of the Bunny

Table 1 Experimental data

Model	Num. Tri	Num. Ver	Feature lines	Feature	Initial feature (s)	Parameterization (s)	Refinement (s)
Dragon	33927	17120	3	Body	0.013	0.067	0.261
Buddha	42052	20985	8	Kyte	0.003	0.012	0.098
Bunny	50236	25120	6	Leg	0.010	0.058	0.225



(a) (b) (c)

Fig.12 Decomposition examples

(a) A Triceratops (5660 triangles, 2832 vertices) which is decomposed into seven components; (b) The decomposed Bunny same to the Bunny in Fig.1a; (c) Decompose the Venus model (16532 triangles, 8268 vertices) into three components, the meshes is resampled along the segmented boundary

CONCLUSIONS AND FUTURE WORK

In this paper, an efficient method of smooth feature line detection is presented. Compared with the method proposed by Lee and Lee (2002), our feature detection algorithm is more robust because the initial feature line generated using our method is more accurate and is nearer to the feature line, which benefits the optimization of the initial feature line with the snakes. Also we give a particular method to parameterize the feature strip. In addition, an interactive mesh decomposition algorithm based on the detected feature line is proposed.

Current algorithm depends on mesh parameterization, as the snakes approach generally deals with 2D situation. Future work includes direct feature line detection method without parameterization, and some other applications such as mesh model editing and simplification.

References

Andrews, S., 2002. Interactive Generation of Feature Curves on Surfaces: A Minimal Path Approach. Master's Thesis, University of Toronto, p.22-32.

Belyaev, A., Ohtake, Y., Abe, K., 2000. Detection of Ridges and Ravines on Range Images and Triangular Meshes. Proceedings of Vision Geometry IX, p.146-154.

Borrelli, V., 1993. Courbures Discretes. Master's Thesis. Universite Claude Bernard-Lyon 1.

Cohen, L.D., 1991. On active contour models and balloons. *Computer Vision, Graphics and Image Processing: Image Understanding*, **53**(2):212-218.

Hubeli, A., Gross, M., 2001. Multiresolution Feature Extraction from Unstructured Meshes. Proc of IEEE Visualization, p.287-294.

Kanai, T., Suzuki, H., 2000. Approximate Shortest Path on a Polyhedral Surface Based on Selective Refinement of the Discrete Graph and Its Application. Proceedings of IEEE Geometric Modeling and Processing 2000 (Theory and Applications). Hong Kong, p.241-250.

Kass, M., Witkin, A., Ierzopoulos, D., 1998. Snakes: Active contour models. *International Journal on Computer Vision*, **1**(4):321-331.

Khaneja, N., Miller, M.I., Grenander, U., 1998. Dynamic programming generation of curves on brain surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(11):1260-1265.

Kimmel, R., Sethian, J.A., 1998. Computing Geodesic Paths on Manifolds. Proceedings of National Academy of Sciences, p.8431-8435.

Lee, Y.J., Lee, S.Y., 2002. Geometric Snakes for Trianglular Meshes. Proceedings of Eurographics 2002, p.229-238.

Ohtake, Y., Belyaev, A., 2001. Automatic detection of geodesic ridges and ravines on polygonal surfaces. *The Journal of Three Dimensional Images*, **15**(1):127-132.

Ohtake, Y., Belyaev, A., Seidel, H.P., 2004. Ridge-valley Lines on Meshes via Implicit Surface Fitting. Proceedings of SIGGRAPH 2004, Computer Graphics Proceedings, Annual Conference Series, ACM, p.609-612.

Page, D. L., Sun, Y., Koschan, A., Paik, J., Abidi, M., 2002. Normal vector voting, crease detection and curvature estimation on large, noisy meshes. *Journal of Graphical Models*, **64**:1-31.

Rössl, C., Kobbelt, L., Seidel, H.P., 2000. Extraction of Feature Lines on Triangulated Surfaces Using Morphological Operators. Smart Graphics 2000, AAAI Spring Symposium. Stanford University, p.71-75.

Xu, C., Prince, J.L., 1998. Snakes, shapes, and gradient vector flow. *IEEE Trans on Image Processing*, **7**(3):359-369.