http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 12 (24): 8555-8562, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.8555.8562

An Interactive Shape Manipulation Method Based on Feature Lines

[1,2]Guijuan Zhang, [1,2]Dianjie Lu, [1,2]Ailin Wang and [1,2]Hong Liu
[1]School of Information Science and Engineering, Shandong Normal University,
[2]Shandong Provincial Key Laboratory for Novel Distributed Computer Software Technology,
Jinan, 250014, China

Abstract: Interactive technique for shape manipulation is highly desirable in geometric modeling community. In general, designers are often required to manipulate the shape of large, complex or specific 3D models with no smooth features. In this study, we assume that the large, complex or special 3D models can be defined and characterized by a rather small set of feature lines. The goal of shape manipulation is achieved by deforming the important geometric features interactively. To this end, we first pre-analyzed the 3D large, complex or specific shapes to get the feature lines. The feature lines are defined as the ridge-valley lines of 3D models. Next, the interactive operations for manipulating the feature lines are proposed. We design a brush operation to manipulate the local regions of focus vertex. The new positions and normals of the vertices in the local region are approximated. The new model is finally reconstructed according to compactly supported basis function with the vertices and their normals after manipulation. Experimental results show that our method enable designers to control the shape of 3D models effectively.

Key words: Shape manipulation, interaction system, feature lines

INTRODUCTION

Shape manipulation is one of the most important tasks in geometric modeling. The goal of shape manipulation is to allow designers to edit the 3D shapes freely. For example, the users could deform the general shape of geometric surfaces interactively while preserving surface details. According to these techniques, designers or even novel users can convert their design concepts to 3D digital models effectively. Therefore, providing interactive techniques for shape manipulation is highly desirable in the geometric modeling community.

In geometric modeling world, researchers have developed many methods to allow users to edit the shape interactively (Botsch and Sorkine, 2008). They are widely applied in controlling the shapes of organic objects such as human body, animals etc. The methods enable users to edit objects shape interactively while preserving the surface details. Since the 3D objects are assumed to be composed by a homogeneous, rubber-like material in the methods, the 3D surface deforms uniformly. The method is suitable for controlling the shape of the objects that has smooth geometric features.

In practice, designers often encounter 3D models that have following characteristics. First, the 3D models with high resolution representation. In general, they are too large and too complex to be viewed in a browser in real-time even with regard to be manipulate and deformed interactively. Second, no smooth features for uniform manipulation are included in the 3D models. In our daily life, a great number of objects are composed by flat surfaces. For example, the furniture, electronic devices etc. Current methods are difficult to process the flat surfaces in these models. So how to manipulate the objects with the above characteristics is of great importance in geometric modeling and still remains a challenging issue.

We propose an interactive method to address the above problems effectively. In our study, we assume that the large, complex or special 3D models can be defined and characterized by a rather small set of curves. The curves can describe the most important geometric features of the 3D models effectively. Compared to the complex 3D models, the small set of curves take up less space in memory. As a result, we complete shape manipulation by interactive controlling the curves even the corresponding 3D model is large and complex. Therefore, by manipulating the important geometric features, the goal of controlling the shape of the large, complex or special 3D models can be achieved effectively.

Our method is based on the notion of feature lines. We define our feature lines to be the ridge-valley lines of 3D shapes (Ohtake *et al*., 2004). The lines are curves on surface along which the surface bends sharply. So, they are powerful shape descriptors. Our method includes

Corresponding Autor: Guijuan Zhang, School of Information Science and Engineering, Shandong Normal University, Jinan, 250014, China

three unique parts. We first pre-analyzed the 3D large, complex or specific shapes to get the feature lines. They are importance geometric features of a 3D model. Next, we design interactive operations to manipulate the importance geometric features. A controllable brush is defined to manipulate the local regions of a focal point. The new positions and normals of all the vertices in this local region are approximated. Finally, the surfaces of manipulated model will be reconstructed by compactly supported basis functions. Results show that our method can help designers to manipulate the model shapes interactively.

RELATED WORK

In geometric modeling society, shape manipulation is one of the most essential tasks. Early work about shape manipulation often adopts the concept of space deformation. It is also called free form deformation. The main idea of shape deformation is to deform shapes according to lattices of the control objects (Milliron *et al*., 2002).

The technique is very powerful because it has two advantages: (1) Suitable for various 3D models. For example, it can process models with free-form surfaces and models that have multiple connected components. (2) High efficiency and robust. The space deformation method is easy to be implemented. What's more, since the cost of the deformation is depended on the complexity of the controlled 3D models rather than on the deformed shape, the space deformation method is efficient. However, these direct control methods are hard to apply in the applications concerning large and complex 3D objects. The main reason is that the large and complex 3D object is difficult to show in real-time due to the limitation of hardware, even with regard to manipulate them interactively. In addition, the underline structure of the 3D models is often destroyed after space deformation.

To address the problems appeared in early work (Singh and Fiume, 1998) as well as Orzan *et al*. (2008) proposes methods to represent the entire objects and image with a small set of curves. For example, Singh and Fiume (1998) process the free-form parametric surface. So, the curve can be easily obtained by the parameter of the controlled surface. As a result, the complexity of shape manipulation is reduced significantly. Inspired from the above two studies, Gal *et al*. (2009) present i Wires to allow interactive manipulation for man-made shapes. The authors also use the notion of wires presented by Singh and Fiume in their method to represent the curves of the key structural features of man-made shapes. Therefore, the method can edit 3D geometry shapes freely while preserving the structure features. The method can well manipulate the 3D objects with regular edges. Differently, the feature lines proposed in this study are more general ones compared to the edge of objects. They are more powerful descriptors for 3D objects.

In this study, we extend the curve method to process more general models not limited to parametric models or models with regular edges. For example, we can manipulate the shapes of models represented by triangles. In addition, the feature lines used in this study are more general curves. They are suitable for describing any important features of the 3D models. Therefore, our method can manipulate their shapes effectively and interactively by simplifying the 3D objects with the important feature lines.

OUR METHOD

Overview: Our method is based on the fact that large, complex or specific 3D objects can be described by a set of feature lines. The feature lines describe the most important characteristics of the 3D objects effectively. Therefore, manipulating 3D shapes can be converted into controlling a set of feature lines effectively. After deforming the feature lines, techniques should also be provided to reconstruct the new 3D model with the deformed 3D feature lines and the original 3D model. Figure 1 demonstrates the framework of our method. Our framework can be divided into of three unique parts.

Feature line extraction: Given 3D models, the first step is to define the feature lines of the models. In this study, we use the notion of ridge-valley lines (Ohtake *et al*., 2004) to represent our feature lines. Since the ridge-valley lines describe the surface bends information that is the most prominent characteristic of a surface, they can fulfill our goal well. Our shape manipulation method deals with the 3D models efficiently and interactively due to such a set of feature lines. So the feature line extraction process is the fundamental part of our method.

Interactive shape manipulation: We provide an interactive system and design an interactive operation to allow users to manipulate 3D models interactively. In our system, users handle the shape deformation by sketching the feature lines conveniently similar to most graphical tools. We propose a brush operation to assist users in controlling the shape of feature lines. Using the brush operation, users can select and drag a vertex on the feature lines. The operation is then propagated to the other vertices in the local region of the selected vertex.

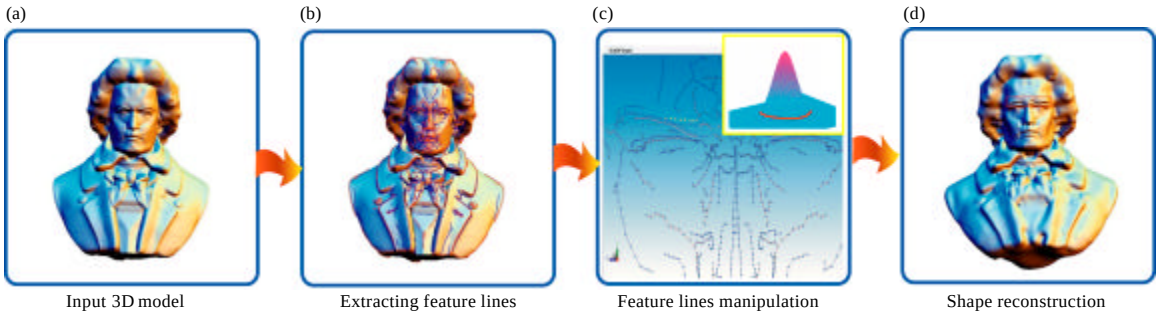

Fig. 1: Framework of our method

After the interactive operation, we approximate the position and normal of each moved vertex on the feature lines. We use a Gaussian kernel function to calculate the new position of each controlled vertex. As for the normal of the new vertex, we approximate it using the points in the adjacent regions. After construct a set of triangles around the vertex and averaging the normals of the triangles, the mean normal will be set as the new normal of the controlled vertex.

Model reconstruction: After shape manipulation in the second step, we get the new feature lines. They are composed by the original vertices on the feature lines as well as the dragged vertices. We will reconstruct the 3D model with the new feature lines. In this study, shape reconstruction from a set of feature lines are base on a multi-scale approach. Given new feature lines, we get the implicit representation of the new model according to a set of compactly supported basis function. After interpolating the implicit surface, we compute its explicit geometric representation with marching cubes algorithm. Therefore, the new model is also triangle-based.

Feature lines extraction: To extract the geometric feature lines used in this study, we need to approximate the curvature tensor and curvature derivatives for each vertex on the mesh. Since the 3D model is composed by many discrete triangles, they are not suitable for computing the curvature derivatives. To alleviate this problem, we compute a continuous implicit function that approximates the original triangle mesh. Then the vertices on the triangle mesh are mapped to the implicit surface. The implicit function is continuous representation and is very suitable for derivative computation. Thus we can readily compute the curvature tensor and curvature derivatives according to the implicit surface. After that, the ridge and valley lines can be obtained and the feature lines in this study are computed finally. The following two steps give the details of the method.

Implicit function approximation: Given a triangle mesh, we compute the implicit representation of the surface according to a set of Radial Basis Function (RBF). In this study, we use the method presented in (Ohtake *et al*., 2003) to build the implicit surface from the input triangle mesh. According to the method (Ohtake *et al*., 2003), the implicit surface can be denoted by:

$$F(x) = \sum_{p_i \in P} \psi_i(\boldsymbol{x}) = \sum_{p_i \in P} [g_i(\boldsymbol{x}) + \lambda_i]\phi_\sigma\left(\left\|\boldsymbol{x} - p_i\right\|\right) \quad (1)$$

In Eq. 1, P is a point set on the triangle mesh, $p_i \in P$ denotes an arbitrary point in the point set P, $\psi_i(x)$ is the basis function. Note that $\phi_\sigma(\|x-p_i\|)$ $\phi(\|x-p_i\|/\sigma)$, $\phi\sigma(r) = \phi(r/\sigma)(r = \|x-p_i\|/\sigma)$ is Wendaland's compactly supported RBF (Wendland, 1995) with σ as its support size. $g_i(x)$ is an implicit function that represents the shape of the implicit surface in a local region of p_i. λ_i denotes the coefficient.

Apparently, in Eq. 1, only $g_i(x)$ and λ_i are unknown. To get the final implicit function F(x) for the triangle mesh, we need to compute $g_i(x)$ and λ_i respectively. We assume that the surface shape in a small local region of p_i is a quadric surface. So, the implicit function $g_i(x)$ can be written as:

$$\begin{aligned} g_i(x) &= w - h(u, v) \\ &= w - \left(au^2 + 2buv + cv^2\right) \end{aligned} \quad (2)$$

where, $w = h(u, v) = au^2+2buw+cv^2$. Here, (u, v, w) is defined as the new local coordinates with p_i as the origin position. (u, v) represents the plane that orthogonal to the surface normal n_i. The positive direction of w coincides with the direction of n_i. In Eq. 2, the coefficients a, b and c can be computed according to the least squares method

$$\text{Min} \sum_{p_j=(u_j,v_j,w_j)\in P} \phi_\sigma\left(\left\|p_j - p_i\right\|\right)\left(w_j - h\left(u_j, v_j\right)\right)^2 \quad (3)$$

So, we complete computing $g_i(x)$.

In order to get λ_i, we calculate function value of Eq. 1 for each p_i:

$$F(p_j) = 0 = \sum_{p_i \in P} \left[g_i(p_j) + \lambda_i \right] \phi_\sigma \left(\| p_j - p_i \| \right) \tag{4}$$

Let, $\Phi ij = \phi_\sigma(\|p_j-p_i\|)$, Eq. 4 can be rewritten as:

$$\sum_{p_i \in p} \lambda_i \Phi_{ij} = -\sum_{p_i \in p} g_i(p_j) \Phi_{ij} \tag{5}$$

Since the right side of Eq. 5 can be computed readily, we finally get a sparse linear equation about the unknown λ_i. By solving the sparse linear equation with iterative method, we finally obtain the coefficient λ_i.

Vertex projection: After getting the implicit function of the triangle mesh, we project the vertex of the triangle mesh onto the implicit surface $F(x) = 0$. The principle of the projection is to find the nearest point $\hat{v}$ on the implicit function as the projection point of the vertex v on the triangle mesh. According to Newton-iterative process, we get:

$$\begin{aligned} &\hat{v}_0 = v \\ &\hat{v}_{k+1} \leftarrow \hat{v}_k - \frac{F(\hat{v}_k)\nabla F(\hat{v}_k)}{\|\nabla F(\hat{v}_k)\|^2} \\ &\text{until} \frac{|F(\hat{v}_n)|}{\|\nabla F(\hat{v}_n)\|} < \frac{\varepsilon}{d} \end{aligned} \tag{6}$$

where, ε controls the precision and d is the length of the main diagonal of the bounding box of the triangle mesh.

Feature lines computation: According to the definition of ridge and valley lines, we need to compute the extrema of the principal curvatures along the curvature directions. Let k_{max} and k_{min} ($k_{max} \geq k_{min}$) denotes the maximal and minimal principal curvatures. Their corresponding principal directions are t_{max} and t_{min}, respectively. The derivate of the principle curvatures along the principal directions is $e_{max} = \partial k_{max}/\partial t_{max}$ and $e_{min} = \partial k_{min}/\partial t_{min}$.

After computing the e_{max} and e_{min} of each vertex, we will test if there exists an extrema on the triangle edge. If so, we will compute the ridge or valley vertex as the extrema point. Take the ridge vertex for example. We first make sure that the angle between $t_{max}(v_1)$ and $t_{max}(v_2)$ is obtuse. Otherwise, we will flip $t_{max}(v_2)$. If the constraints Eq. 7, 8 are satisfied, we will interpolate to get the theoretical ridge vertex with Eq. 9:

$$\begin{cases} k_{max}(v_i) > |k_{min}(v_i)|, i = 1,2 \\ e_{max}(v_1) e_{max}(v_2) < 0 \end{cases} \tag{7}$$

$$Y(v_i) = e_{max}(v_i)\left[(v_{3-i} - v_i) t_{max}(v_i) \right], i = 1, 2 \tag{8}$$

$$p = \frac{|e_{max}(v_2)| v_1 + |e_{max}(v_1)| v_2}{|e_{max}(v_1)| + |e_{max}(v_2)|} \tag{9}$$

In the end, ridge lines will be obtained by connecting the ridge vertices detected on the triangle edges. Similar operation is also executed to compute the valley lines. Both the ridge and valley lines form the feature lines used in this study.

Interactive manipulation: Obviously, the ridge-valley lines define the surface creases so that, they can describe complex shapes effectively. By extracting ridge-valley lines from the triangle meshes, we can simplify the large and complex triangle mesh significantly. Our shape manipulation is built upon the effective feature lines. So, the manipulation method allows us to control high resolution and complex models in real-time.

To this end, we define a controllable brush. The brush makes it possible to manipulate the local region that corresponds to a selected vertex on the feature lines. After dragging the vertex on feature lines, we also compute the new positions and new normals of the vertices that in the local region of the focal point.

Regional controllable brush: The main idea of our interactive manipulation is to control the deformation of the shape using its feature lines. We define a brush tool here. According to the brush tool, the local region in the small vicinity of the focal point will be manipulated. Note that the focal point is a selected vertex on the feature line. We define the local region with a sphere here. The radius of the sphere r can be adjusted for different shapes or different part of the shapes. Different control radius will affect different number of vertices on feature lines.

Position and normal computation: When we adjust the position of the selected vertex x_0, the vertex in the local region of x_0 will also be affected by the brush operation. Therefore, we are required to approximate the new position and the normal of each vertex that in the local region of the selected vertex.

Obviously, the vertex that is more close to the dragged vertex is more affected. As shown in Fig. 2, the vertex's position changed according to the distance from current vertex to the focal (center) point. We adopt the Gaussian kernel to compute the new positions for the controlled vertices. The Gaussian kernel is defined:

$$d(x_0, x) = \exp\left(-\frac{\|x_0 - x\|^2}{2r^2} \right) \tag{10}$$

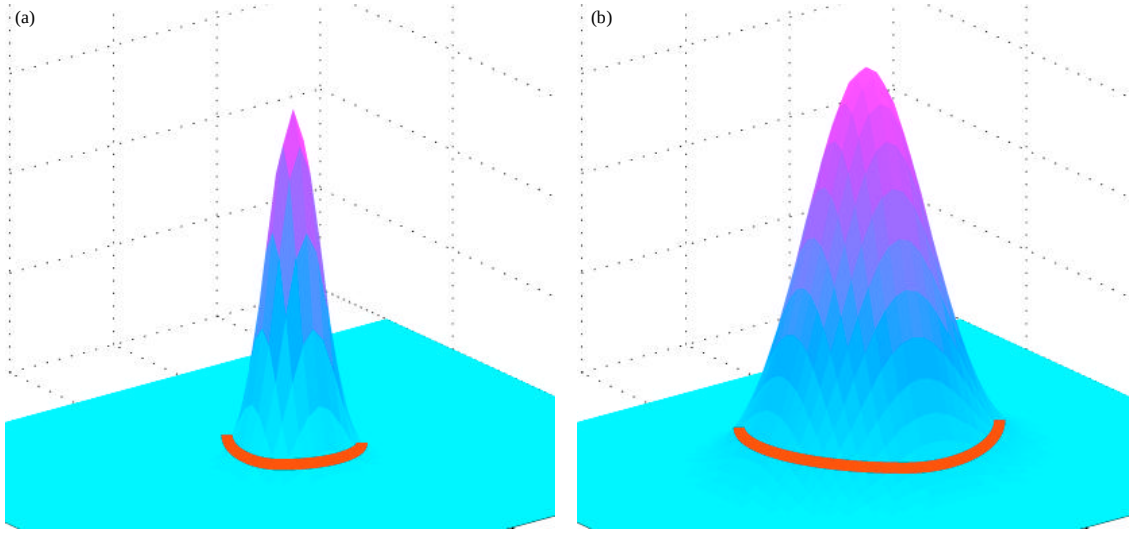

Fig. 2(a-b): Controllable brush with different radius

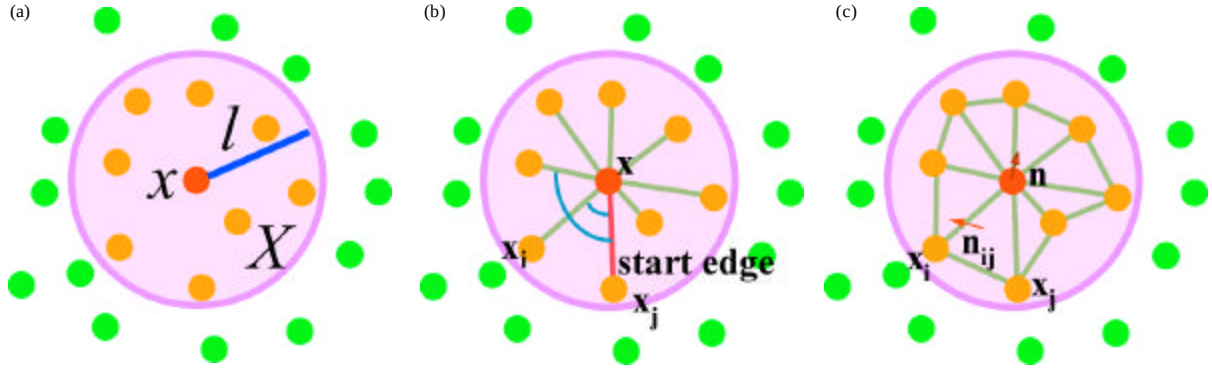

Fig. 3(a-c): Procedure of the normal approximation, (a) Local region with control radius l, (b) The angles between the start edge and the other edges and (c) The triangles formed by adjacent edges

where x_0 denotes the selected vertex (focal point), x is the vertex in the local region of x_0, r is the control radius of the brush, d denotes the position changes. Note that different brush radius will affect different size of local region as shown in Fig. 3. In the two images, the positions of the corresponding vertices in the controlled local region change due to the Gaussian kernel.

As a result, for a vertex x that in the local region of x_0, the new position can be computed by:

$$x = x+d(x_0, x) \quad (11)$$

In addition to the new position, we are also required to compute the normal for the x in the local region. In this study, we compute the vertex's normal with its adjacent vertices. We first create several triangles and compute the normal of each triangle. As for a vertex x, let l be the radius of the local region. So, the adjacent vertices set X can be obtained by:

$$X = \{x_i \mid \|x_i-x\| \le l\} \quad (12)$$

As shown in Fig. 3a, l is the radius of the local region. The vertex in this region (also denoted as X) is employed to compute the normal vector of the selected vertex x.

Next, we make a straight edge between two vertices x and x_i($x_i \in X$). The direction of each edge is $x\text{-}x_i$. Let an edge $x\text{-}x_j$ be the start edge. As shown in Fig. 3b, the angle between the start edge and all the other edges can be computed. For example, the angle between $x\text{-}x_i$ and the start edge $x\text{-}x_j$ can be obtained by:

$$\theta_{i,j} = \arccos\left(\frac{(x - x_i)\cdot(x - x_j)}{|x - x_i||x - x_j|}\right) \quad (13)$$

After calculating all the angles, we re-order them to find a sequence of edges. Every two adjacent edges form a triangle as shown in Fig. 3c, the normal vector of the triangle $\Delta xx_j x_i$ is:

$$n_{ij} = \text{normalize}\ ((x\text{-}x_i)\times(x\text{-}x_j)) \quad (14)$$

Finally, the normal of vertex x is obtained by averaging the normal of the each triangle that co-vertex at x:

$$n = \frac{1}{|X|}\sum_{\substack{x_i,x_j\in X \\ adja(x_i,x_j)}} n_{i,j} \quad (15)$$

Model reconstruction: After processing all the controlled vertices, we reconstruct the new model by constructing the implicit function as:

$$F(x) = \sum_{p_i\in P_{new}} \psi_i(x) = \sum_{p_i\in P_{new}} [g_i(x) + \lambda_i]\phi_\sigma(\|x - p_i\|) \quad (16)$$

Here, $F(x)$, $\psi_i(x)$, $g_i(x)$, λ_i and $\phi_\sigma(\|x\text{-}p_i\|)$ are defined in the same way as Eq. 1. Differently, the point set here is p_{new}. It is a hierarchal point set which is composed by the vertices randomly selected on the original mesh and the vertices on the new feature lines after manipulation.

Constructing the hierarchal point set: To compute the implicit representation of the new model in Eq. 16, we are required to construct the hierarchal point set $p_{new} = \{P_1, P_2, \ldots P_n\}$. To achieve this goal, we first find the points that belong to the lowest level of P_n and then construct the hierarchal structure iteratively.

The points in P_{on} are composed by two sets of points:

$$P_n = P_{on} \cup P_{FL} \quad (17)$$

where points in P_{on} are random vertices on the triangle mesh while points in P_{FL} are ridge or valley vertices on feature liens. Both the two point set are uniformly distributed on corresponding geometry objects. Since the feature lines describe the most prominent geometric characteristics of a mesh, we sample more points from the feature lines. Therefore, we set $|P_{FL}| >> |P_{on}|$ in this study. We randomly select ridge and valley vertices on feature lines to get P_{FL}. As for P_{on}, since the area of each triangle of the 3D model is varied greatly from each other, we adopt an algorithm to sample reasonable points on them. The algorithm is consists of two unique parts: selecting a random triangle and generating a random point inside the triangle. It will stop when enough samples are obtained. In the algorithm, we first compute the area of each triangle and then generate sample points under the guidance of the triangle area. To do this, we calculate the sample probability of each triangle according to its area. So the triangle with larger area will be sampled more often in the sampling procedure.

Note that the sample points on triangle surface are computed by barycentric coordinates. Every point inside a triangle can be represented by its three vertices. Given two random numbers u and v with uniform distribution in [0,1] ($u+v\leq 1$), we compute the position of the sample point as:

$$\begin{aligned} w &= 1 - (u + v) \\ p_i &= u\times v_0 + v\times v_1 + w\times v_2 \end{aligned} \quad (18)$$

where v_0, v_1, v_2 are three vertices of the triangle.

Computing the explicit representation: Given the hierarchal point set P_{new}, we use a multi-scale approach based on compactly supported radius function (Ohtake *et al*., 2003) to get its implicit representation as shown in Eq. 16. The construction process of the implicit function is described in the section of feature line extraction. The only difference is that the point set differs to each other.

After obtaining the implicit representation, the explicate triangle mesh of the final model after interactive manipulation will be obtained by marching cubes method (Lorensen and Cline, 1987). The method can extract a polygonal mesh given an implicit function by constructing a table with 15 cases. According to the table, the polygons(s) will be determined to denote the iso-surface that passes through the cube. The explicit representation of the new model will be obtained by fusing the individual polygons finally.

RESULTS

We build an interactive system for manipulating shapes of 3D models in this study. All the results in this study are gathered on a PC with 2.6 GHz Intel Core 2 Duo CPU and 2 GB memory.

Figure 4 shows the interface of our system. In the system, we can select the drag the vertex (focal point) on the feature lines. The controlled brush allows manipulating the local shape in a small region.

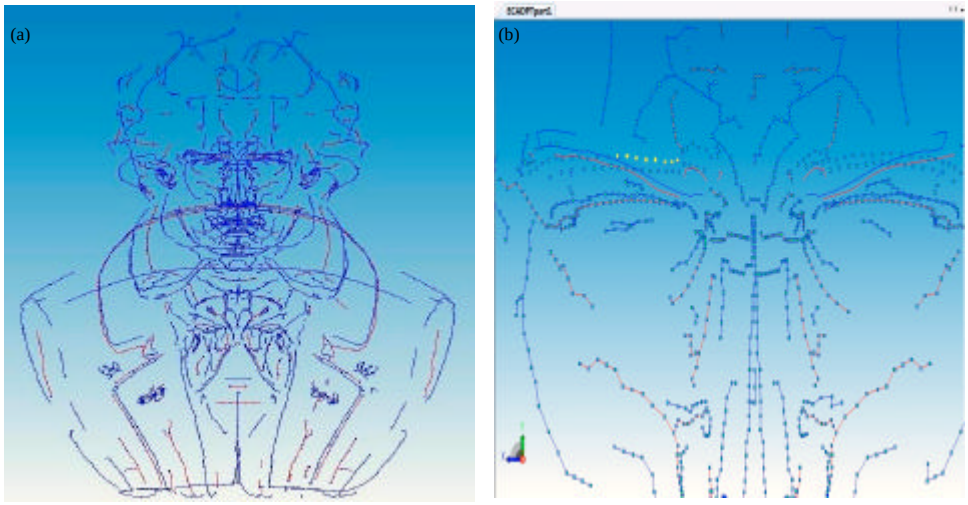

Fig. 4(a-b): Interface of our interaction system, (a) Feature lines showed in our system interface and (b) Selecting vertices on feature lines using our controllable brush

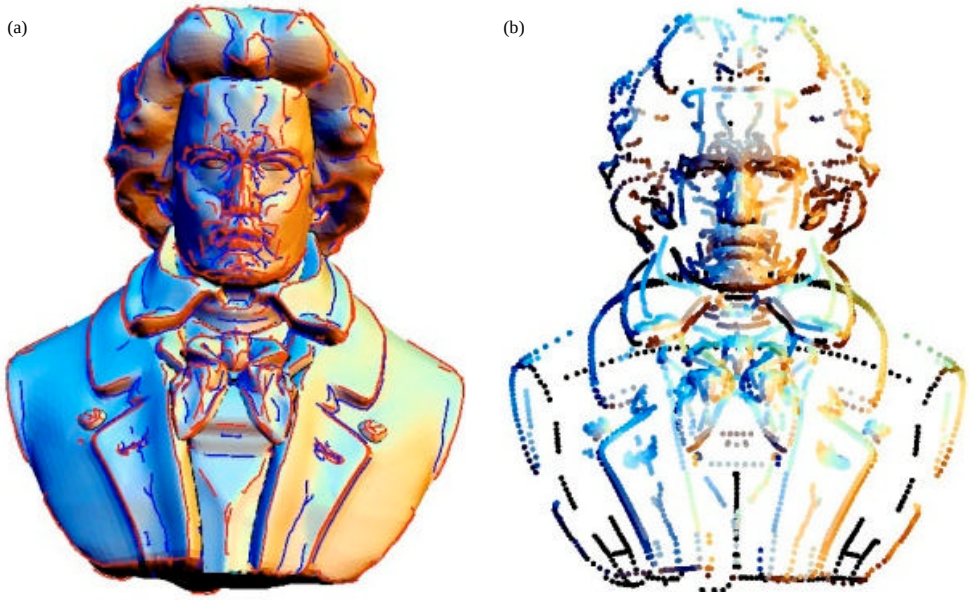

Fig. 5(a-b): Original feature lines and deformed feature lines

To complete the manipulation process, we first extract the feature lines from the input model. We take a Beethoven model as the input. Figure 5 shows the extracted ridge and valley lines in our system. Note that the red lines denote ridge lines and the blue lines denote the valley lines. They are formed the feature lines in this study. Right image of Fig. 5 shows the new hierarchal point set P_{new}. The points are formed by samples from feature lines and samples from common triangle surfaces, respectively. Note that feature lines in the eyebrow of the Beetooven model are deformed during interactive manipulation.

In Fig. 6, the new model is apparently deformed after surface reconstruction. We achieve the goal by selecting the vertices on the feature lines of the eyebrow and dragging the vertices according to brush operation. Right image of Fig. 6 shows the new model after deformation. Results show that our method can completed the shape manipulation interactively and effectively.

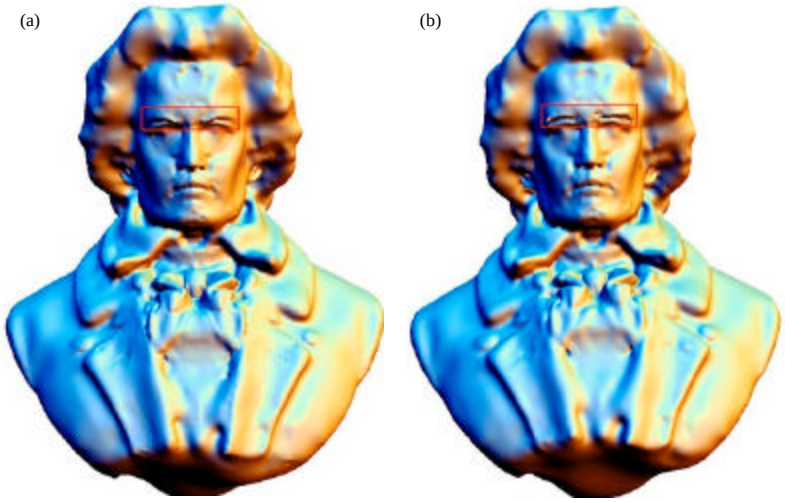

Fig. 6: Models before and after shape manipulation

CONCLUSION

In this study, we propose an interactive method for manipulating the shapes of 3D objects based on the notion of feature lines. Since the lines are curves on surface where the surface bends sharply, they are good shape descriptors. We extract the feature lines according to the definition of ridge and valley lines. A brush operation is also defined to assist the manipulation. The surfaces of new model after manipulation will be reconstructed by compactly supported basis functions. Results show that our method allows designers to manipulate the model shapes interactively.

We will extend our interactive manipulation method to allow more flexible and effective shape deformation applications in our future work.

ACKNOWLEDGMENT

This study is supported by National Natural Science Foundation of People's Republic of China under Grant No. 61202225, 61272094, Project of Shandong Province Higher Educational Science and Technology Program under Grant No. J13LN13, Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant No. 20133704120009.

REFERENCES

Botsch, M. and O. Sorkine, 2008. On linear variational surface deformation methods. IEEE Trans. Visualization Comput. Graphics, 14: 213-230.

Gal, R., O. Sorkine, N.J. Mitra and D. Cohen-Or, 2009. iWIRES: An analyze-and-edit approach to shape manipulation. ACM Trans. Graphics, Vol. 28. 10.1145/1531326.1531339

Lorensen, W.E. and H.E. Cline, 1987. Marching cubes: A high resolution 3D surface construction algorithm. ACM Siggraph Comput. Graphics, 21: 163-170.

Milliron, T., R.J. Jensen, R. Barzel and A. Finkelstein, 2002. A framework for geometric warps and deformations. ACM Trans. Graphics, 21: 20-51.

Ohtake, Y., A. Belyaev and H.P. Seidel, 2003. A Multi-scale approach to 3D scattered data interpolation with compactly supported basis functions. Proceedings of the Shape Modeling International, May 12-15, 2003, IEEE Computer Society, pp: 153-161.

Ohtake, Y., A. Belyaev and H.P. Seidel, 2004. Ridge-valley lines on meshes via implicit surface fitting. ACM Trans. Graphics, 23: 609-612.

Orzan, A., A. Bousseau, H. Winnemoller, P. Barla, J. Thollot and D. Salesin, 2008. Diffusion curves: A vector representation for Smooth-shaded images. ACM Trans. Graphics, Vol. 27. 10.1145/1360612.1360691

Singh, K. and E. Fiume, 1998. Wires: A geometric deformation technique. Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, July 19-24, 1998, Orlando, pp: 405-414.

Wendland, H., 1995. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. Adv. Comput. Math., 4: 389-396.