Topology-Consistent Design for 3D Freeform Meshes with Harmonic Interpolation

Bin Sheng[1] and Enhua Wu[1,2]

[1] Faculty of Science and Technology, University of Macau, Macao, China
bsheng@sftw.umac.mo
[2] State Key Lab of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
ehwu@umac.mo, enhua@iscas.cn

Abstract. One of the most exciting aspects of shape modeling is the development of new algorithms and methods to create unusual, interesting and aesthetically pleasing shapes. In this paper, we present an interactive modeling system for generating freeform surfaces using a 2D user interface. In this paper, we firstly interpret the given 2D curve to be the projection of the 3D curve that has the minimum curvature. Then we propose a topology-consistent strategy to trace the simple faces on the interconnecting 3D curves. By using the face tracing algorithm, our system can identify the 3D surfaces automatically. After obtaining the boundary curves for the faces, we apply *Delaunay* triangulation on these faces. Finally, the shape of the triangule surface mesh that follows the 3D boundary curves is computed by using harmonic interpolation. Solving Laplace's equations, we are able to construct simple and conceptual 3D models interactively. Although the incorporation of topological manipulation will introduce many new(and interesting) design problems that cannot be addressed in this paper, we show that automatically generated models are both beneficial and feasible.

1 Introduction

Nowadays, most designers still prefer to express their creative design idea through 2D sketches. However, space curve sketching using the 2D user interface has a challenging and active research problem of interpreting 2D sketches of curves and surfaces as a 3D entity. Obviously, this kind of reconstruction should need more information such as general assumptions or experiences, database of 2D-3D geometrical correlations or given models of reconstructed objects. it is well known that the fundamental difficulty is in choosing the appropriate one.

In this paper, we attempt to provide a physically sound and topologically elegant method of 3D model construction from 2D curves. Similar to the approach used in [1], we try to interpret the 3D curve corresponding to the given 2D curve to be the one that minimizes the maximum normal curvature in 3D. However, until now our construction of the 2D curves can only support the network of the 3D curves as the control profiles of smooth surfaces, and do not provide general

L. Ma, R. Nakatsu, and M. Rauterberg (Eds.): ICEC 2007, LNCS 4740, pp. 121–132, 2007.

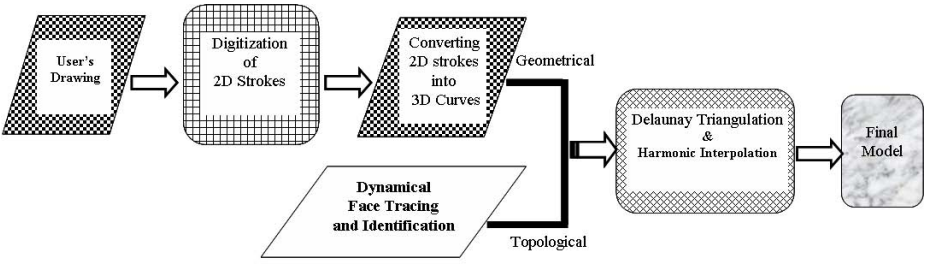

Fig. 1. The pipeline for Topology-consistent Design with Harmonic Interpolation

topology. In order to identify the faces on the mesh with topology consistency, we use topology graph operations which are similar to Euler's operations and based on graph embedding. The biggest advantage of our operations is that they are extremely simple and always guarantee topological consistency. We also developed a data structure, called *Bidirectional Face Table*(BFT), to support topological computation and identify the faces on the mesh efficiently.

The second issue we address in this paper is the generation of smooth surfaces on the above identified simple faces on the mesh. The 3D surface shape is generated in the form of a 3D parametric surface, which is obtained by transforming the 2D plane into 3D surfaces and automatically interpolating the mesh vertices of the 2D triangle mesh into 3D.

Primarily, there are four main contributions in this paper as follows:

- The topology of meshes can be updated automatically during the sketch process simply by adding or removing the key points and curves.
- Smooth free-form surfaces can be generated with simple user input, and adopt a face tracing algorithm for the simple face identification.
- Given the articulated sketches, we present an effective algorithm to generated shape of the surface mesh using a finite element method (FEM). All the internal vertices on the mesh are interpolated by using the harmonic functions.
- We developed an interactive modeling system to test a lot of drawings. It is proved that the system can create many reasonable shapes from the profile and silhouette sketches

2 Related Work

Some shape modeling systems are based on tensor product parametric surfaces such as tensor product B-spline [2]. However surfaces can only support quadrilateral meshes as control meshes for smooth surfaces and do not provide general topology [3]. And the restriction to quadrilateral meshes also makes the modeling process difficult for users. There have been significant efforts in solving this problem by still staying in the parametric surface realm [4].Recently, some papers have been presented on fitting surfaces to a network of curves.

On the other hand, three-dimensional shape modeling systems that use a volumetric data structure directly are relatively new [5,6] as compared with other popular modeling primitives, such as polygons, NURBS, and subdivision surfaces. Recently, a scripting language [7], octree [8], subdivision volume [9], and level set [10] have been used as volumetric modeling methodologies. Sketch-based modeling using standard mouse operations became popular in the past decade. One of the earliest sketching systems was Viking [11], which was designed in the context of prototypic CAD models. Later works include SKETCH [12] and Teddy [13]. The SKETCH system is intended to sketch a scene consisting of simple primitives, such as boxes and cones, while the Teddy system is designed to create rotund objects with spherical topology. Although improvements to the original Teddy system have been recently proposed [14,15], extending the topological variety of creatable models is still an unsolved problem.

3 Curve Construction

In our implementation, the given 2D curves are interpreted to be the projection of the 3D curve that has minimum curvature, by incorporating the idea from [1].In our implementation, the resulting 3D space curve, though a very close approximation of the curvature minimizing curve, has non-smooth, high frequency transitions of the depth values across the critical points. We use the *Laplacian filter* to obtain a final smooth space curve as it is extremely fast and simple as required by interactive sketching applications.

After transforming the 2D curers into 3D, our system also has the ability to form these curves into a network (graph). In this process, we call the end points of a curve to be the linkage points. And every point on the drawn curves could also be the linkage point, assigned by the users. Hence user could interactively specify the connectivity between the linguae points on the 2D interface. In this way, the curve network could be generated. Afterwards, we use an automatical recognition process of patch identification, where the the resulting network is the input. Once the patch boundary connectivity is known, the topological triangular mesh can be constructed, and then we can generates 3D surfaces by using Laplacian interpolation.

3.1 Conversion of 2D Curves to 3D Outlines

Usually, the 2D curves drawn freehand in sketching applications are not Bezier curves, but a sequence of edge connected points. Hence, we have to adapt the curvature minimizing scheme for the depth extraction for a generic 2D curve. Obviously, one way is to fit a Bezier curve to the input point sequence and interpolate the depth of this 2D Bezier curve. However, this is a computationally expensive operation and most importantly, the error in curve fitting will provide a distracting and unexpected feedback to the user.

In this subsection, we give a fast and simple method for finding the depth of the sketched 2D curve points that approximates the minimum curvature 3D

curve. We assume that the depth values of the control points are equally spaced and the closest point on the curve from any control point has the same depth value as the control point. Hence if we identify these key points on the curve that are closest to the control points and space their depth values equally, we will have the first approximation of the space curve. Since we do not have these 3D Bezier curve control points and hence of its 2D projection, identifying even the correct number of key points is difficult.

Even though we do not have an explicit Bezier representation of the 2D sketched curve, the number of critical points in the curve is the lower bound on the number of control points of the hypothetical Bezier representation of the same curve. The critical point need not be the closest curve point (key point) to the corresponding control point. The fundamental source of approximation in our algorithm for curvature minimizing depth interpolation comes from the fact that we assume that these critical points are key points.

The second source of approximation comes from the fact that, there need not be a critical point corresponding to every control point. Hence all the required control points cannot be found by just one sequence of critical points. Differing from the previous approaches which hierarchically subdivide the curves, we solve this issue of selecting the key points, by simplifying the original 2D curve from the first point location. In practice, the hieratical method usually obtain the slow convergence, because that it takes many iterations to capture the key points, due to the local smoothness of the 2D curve. On the other hand, if we assume the starting point could be the key point, the alterative solution comes naturally and effectively. The basic idea of simplification is eliminating intermediate nodes in a straight boundary segment except two end nodes. Based on this idea, the algorithm inherently includes a routine for simplifying a curved boundary exactly made of many small segments. This is an artifact caused by the digitalizing the user's drawing. Meanwhile, what is important, is that if we perform the curve simplification to select the key points, we can easily apply curvature minimizing method to interpret the 2D curves, and the geometrical feature and the smoothness of curves still hold well for each curved segments.

Thus, the key points are extracted by the curve simplification. And the specific algorithm proposed here is based on the concept that newly simplified boundary segments are within a preset tolerance from original curve segments from 2D user's drawing. At first, two deviation tolerance schemes were examined. The one is limiting a maximum error between newly generated simplifying stroke segments and original stroke segments. The other is limiting a sum of errors between two different stroke segments. The latter scheme gave locally big deviations between the two boundaries although it limited a total accumulation of errors. So the former scheme was chosen for simplifying the curve from the original drawing. The summary of the algorithm for finding the key points by simplifying the curve is given as pseudo-codes in Algorithm 3-1.

Algorithm 3-1. KeyPointSelection(Curve C))

Input: Sample points of a 2D curve C.

Procedure:

Let LS be {All the sample points of C}.
$\{v_0, v_1, v_2, ..., v_{n-1}\}$ is the point sequence from LS.
Let KS be { point $v | v \in C \wedge$ v is a key point }.
Let τ be a preset tolerance to limit the difference between the original curve from LS by and the simplified curve from KS.
Let i and j be the node indices of original curve.
$KS \leftarrow \{v_0, v_{n-1}\}$, $i \leftarrow 0$, $j \leftarrow 2$.
Loop:
$KS = KS \cup \{v_i\}$
`IF` j==n-1 `THEN`
return; {Break Looping.}
`ELSE`
Calculate a line, L_{ij} , connecting v_i and v_j.
Calculate the distances from the intermediate nodes,v_k, where $i < k < j$, to L_{ij}.
Let the distances be d_k and the maximum among d_k be d_{max}.
if $d_{max} > \tau$ **then**
$KS = KS \cup \{v_{j-1}\}$;
$i \leftarrow j - 1$.
end if
$j \leftarrow j + 1$.
`END ELSE.`
END Loop.
return KS.

3.2 Automatic Face Tracing and Identification

After transforming the 2D curves into 3D, our system also has the ability to form these curves into a network (graph), which will directly control the generation of the free-form surfaces on the curved boundaries. In this process, we call the end points of a curve to be the linkage points. And any other point on a (already drawn) curve could also be the linkage point, assigned by the users. Then user could interactively specify the connectivity between the linkage points on the 2D interface. In this way, the curve network could be generated. Afterwards, we use an automatical recognition process of the face tracing and identification, where the the resulting topology for the input control graph can have the on-the-fly change.

3.3 Topological Operations on the Curve Networks

Our goal is to generate the 3D free-form surfaces from a network of input curves. Hence, in our current implementation, once the user sketches the desired shape

of the object in the form of outline curves, the system should have some "intelligence" to help the user identify the face forming the surface. Please note that our system allows various models including smooth and sharp boundaries, and we consider the designed meshes can be divided into simpler surfaces, which are called *faces*. So the 3D curves here are nested to form the inter-connected boundaries of these faces.

Theoretically speaking, the face recognition is a face-tracing work in the graph represented by the network of curves. In order to make system automatically recognize the faces on the curve network and accelerate the further triangulation on the faces, we apply the graph rotation system [16] on the nested curves, as shown in Fig. 2. So that we can identify the following topological operations on the rotation systems as essential for the mesh modeling:

- **Face-Trace(f)** outputs a boundary walk of the face f. This is related to reconstructing a "polygon"(face) in the curve graph and to triangulation of the faces on the mesh.
- **Vertex-Trace(v)** outputs the edges incident on the vertex v in the (circular) ordering of the rotation at v. This is useful when the constructed mesh needs reshaping.
- **Insert(v_1,v_2,e)** inserts the new edge e between the face corners v_1 and v_2(a face corner is a subsequence of a face boundary walk consisting of two consecutive edges plus the vertex between them). This is used in revision of the curve network and triangulation of the face on the mesh, changing the topology of the mesh dynamically.
- **Delete(e)** deletes the edge e from the current embedding. This is the converse operation of edge insertion and is also used in changing the topology of the mesh.
- **CoFacial(v_1,v_2)** returns true if the two face corners v_1 and v_2 belong to the same face of the current embedding and false otherwise. This operation is useful in maintaining topological consistency of the mesh.

We have initiated the face tracing and identification algorithms on graph rotation systems in order to implement these operations efficiently. We have first observed that if a rotation system of a graph is represented in the edge-list form, which for each vertex v contains the list of its incident edges, arranged in the order according to the rotation at v, the representation does not efficiently support the operations listed above. Here we introduce a new data structure and

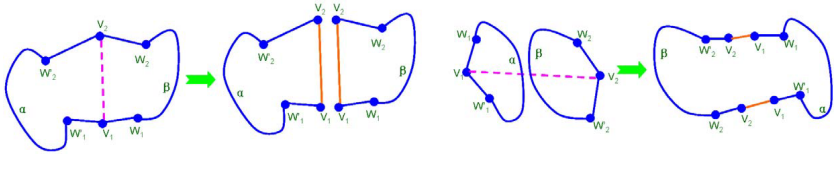

Fig. 2. Inserting an edge between two corners of the same face(*left*). Inserting an edge between two corners from two different faces.(*right*).

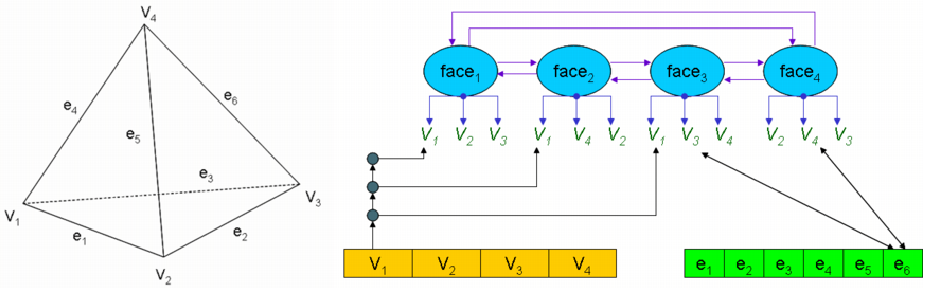

Fig. 3. An illustration of the BFT data structure for a tetrahedron

show that the new data structure efficiently supports all of the operations listed above.

Each face is given by a sequence of vertices corresponding to a boundary traversing of the face. The vertex appearances in the sequence will be called vertex nodes. Note that two consecutive vertex nodes in the sequence correspond to an edge side in the embedding. The sequence is represented by a cyclically concatenatable data structure. For specific discussion, we will use 2-3 trees for this concatenatable data structure. For readers' convenience, we recall 2-3 tree is a balanced tree whose depth is always logarithmic in the number of nodes in the tree. Moreover, operations on 2-3 trees such as inserting a node, deleting a node, splitting a 2-3 tree into two 2-3 trees, and concatenating two 2-3 trees into a single 2-3 tree, can all be done in logarithmic time.

Definition 1. Let $\rho(G)$ be an embedding of a graph $G = (V, E)$ with face set F. A *Bidirectional Face Table*(BFT) for the embedding $\rho(G)$ is a triple $L =< F, V, \epsilon >$, where the face list F consists of a set of $|F|$ sequences. Each is given by a 2-3 tree and corresponds to the boundary walk of a face in the embedding $\rho(G)$. Moreover, the roots of the 2-3 trees are connected by a circular doubly liked list. The vertex array V has $|V|$ items. Each $V[v]$ is a linked list of pointers to the vertex nodes of v in the 2-3 trees in F. The edge array ε has $|E|$ items.Each $\varepsilon[e]$ is doubly linked to the first vertex nodes of the two edge sides of the edge e in the 2-3 trees in F.

Figure 3 gives an illustration of the BFT data structure for a tetrahedron. It can be shown that the BFT structure and the BFT structure used in computational geometry can be converted from one to the other in the linear time. This implies that the computer space used by a BFT structure to represent a mesh modeling is linear in the size of the mesh, which is the best possible.

4 Surface Construction

Once the boundary connectivity for each face is known, our system constructs topological triangle mesh, and then generates a smooth freeform surface based

on the given boundary curve network, using Laplacian interpolation followed by surface refining.

4.1 Triangulation and Harmonic Interpolation

After we obtain all the simple faces with their boundary curves, the generation of the meshes for the 3D profiles has the following two steps:

1. ***Topological Triangulation***-The input to this step is a closed sequence of connected curves. The output is a topology of the triangulation (with unrefined geometric coordinates for the vertices) that consists of input vertices on the curve and the newly introduced vertices in the interior of the identified face.
2. ***3D Interpolation of Surface Mesh***-The triangle mesh is automatically extended to the shape that represents the freeform surface by harmonic interpolation. Afterwards, mesh refining is processed to reduce the geometrical noise in generated mesh.

Specifically, in step 1, the boundary curves of the identified faces that are created in previous process are projected onto a plane that best approximates the curve points. Hence our triangulation method is applied on 2D domain, generally based on the *Delaunay triangulation*, where a triangles is created by joining nearest neighboring nodes as its edges. Since all the vertices having been projected onto the plane, such triangulation actually is a 2D Delaunay triangulation, called *topological triangulation*. Here,the input for mesh generation is a Planar Straight Line Graph (PSLG), including the vertices of the boundary curves projected onto the plane. Quality mesh generation for a PSLG is a standard computation involving two control parameters, a triangle size tolerance (μ) and a minimum angle tolerance (θ). A quality mesh generation program computes a constrained Delaunay triangulation that is a refinement of the input PSLG, with no triangle exceeding μ in size and no angle less than θ, except possibly for small angles between constrained edges in the input PSLG. The modeler uses the freely distributed program program Triangle. As a result, the topological triangle mesh that is suitably fine for modeling the surface is obtained.

On the other hand, the nodes in the meshes which are obtained by triangulating the surface domain with the boundary curve constraints in 2D, are interpolated to give a 3D shape by using the transformed profiles in 3D. The surface mesh vertices are interpolated to conform to the profiles transformed in 3D using Laplacian interpolation. We seek to define the parametric surface by the function $f(r,s) = (x(r,s), y(r,s), z(r,s))$ representing the 3D surface mesh shape, where r and s are the parameters for a position in the 2D plane domain Ω. We now specify $\partial\Omega$ be the outline of the faces projected onto the 2D plane, and let Γ be the original 3D surface profiles which is interactively modeled and automatically captured before. There are two kinds of boundary conditions applied to the domain Ω, the one is a *Neumann* boundary condition defined at$\partial\Omega$ because$\partial\Omega$ is considered as free, and the other is a *Dirichlet* boundary condition defined at Γ, which are illustrated in Figure 4. Let $g(r,s)$ be the forced boundary condition at Γ, which is

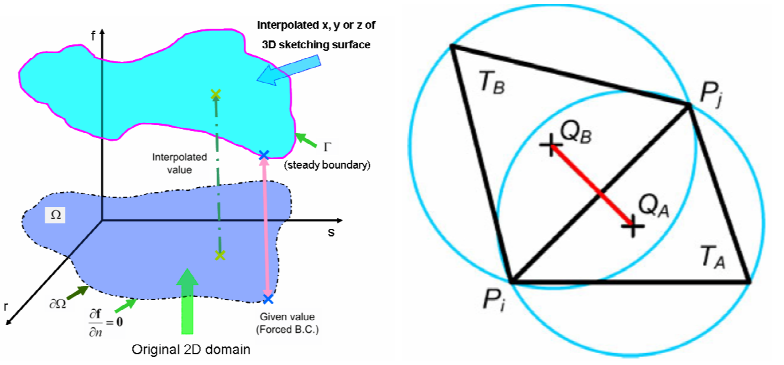

Fig. 4. Interpolation of 3D surface mesh coordinates(left), neighboring triangles of a Delaunay edge(right)

given by the 3D curves generated interactively as discussed previously. The interpolation problem is mathematically defined as follows.

$$\nabla^2 f = \frac{\partial^2 f}{\partial r^2} + \frac{\partial^2 f}{\partial s^2} = 0 \qquad \text{for } (r,s) \in \Omega \tag{1-1}$$

$$f(r,s) = g(r,s) \qquad \text{for } (r,s) \in \Gamma \tag{1-2}$$

$$\frac{\partial f}{\partial n} = 0 \qquad \text{for } (r,s) \in \partial\Omega \tag{1-3}$$

Functions that satisfy Laplace's equation, $\nabla F = 0$, are called *harmonic functions*. The Laplace's equation is interpreted as partial differential equations governing an elastic membrane problem. The function f specified by Equations(1) can be viewed as 3 harmonic functions defined in Ω that interpolates $g(r,s)$ on Γ. Now the Laplacian interpolation problem of the surface mesh is solved by a *finite element method* (FEM) defined for the discretized domain. However, we do not need much of the apparatus of the FEM, because in this simple case, the linear equations for the coordinates, x_k, y_k, and z_k, can be formed from a single parameter, γ_{ij}, associated with the mesh edge between vertices P_i and P_j, as shown in Figure 4. Because we are using a Delaunay triangulation, γ_{ij} has a simple geometric description. For an edge internal to the mesh, let T_A and T_B be the mesh triangles that share the common edge between P_i and P_j. Let Q_A and Q_B be the circumcenters of T_A and T_B, then $\gamma_{ij} = ||Q_A - Q_B||/||P_i - P_j||$; which is discussed in detail by [17].

If edge P_i to P_j is on the boundary of the surface, then T_A has no neighbor T_B. In this case, Q_B is the midpoint of edge P_i to P_j and T_A must be acute so that Q_A stays in T_A. This restriction on the triangles at the 3D surface boundary basically requires the mesh to be adequately fine and well shaped near the boundary.

To compute the discretized coordinate function, $x(r, s)$, the FEM method specifies a system of linear equations of the form $Ax = b$ where x_k is the value of the coordinate function at mesh vertex Pk. In Equation (2), we give the form of the equation associated with a mesh vertex, P_i, not on the surface skeleton, that is connect to m neighboring mesh vertices, $P_{j_k}, k = 1$ to m by mesh edges.

$$(\sum_{m}^{k=1} \gamma_{i,j_k})x_i - \sum_{m}^{k=1} \gamma_{i,j_k} x_{j_k} = 0 \tag{2}$$

If $Pi = (r_i, s_i)$lies on the surface profiles,then the equation is simply $x_i = x_{sk}(r_i, s_i)$. This is a system of linear equations with sparse matrix, A, which can be solved by most sparse matrix techniques or software.

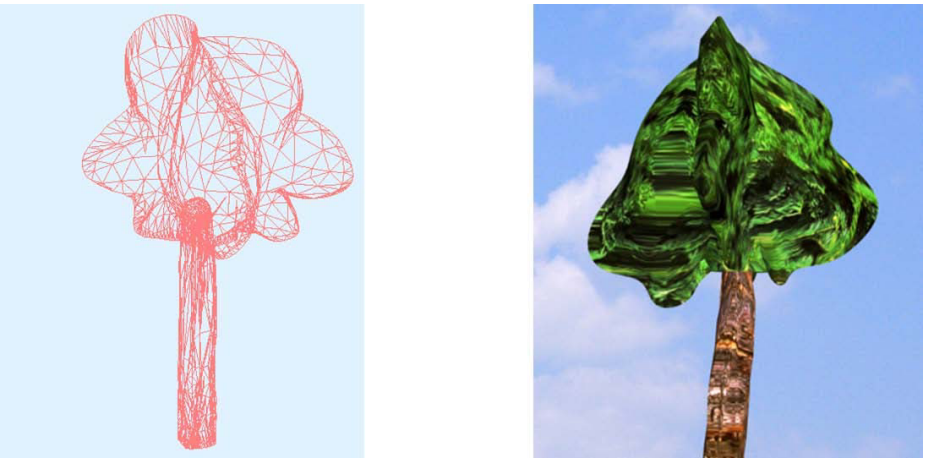

Fig. 5. Our system creates the mesh model for tree design(left). The textured tree model is rendered in the virtual environment(right).

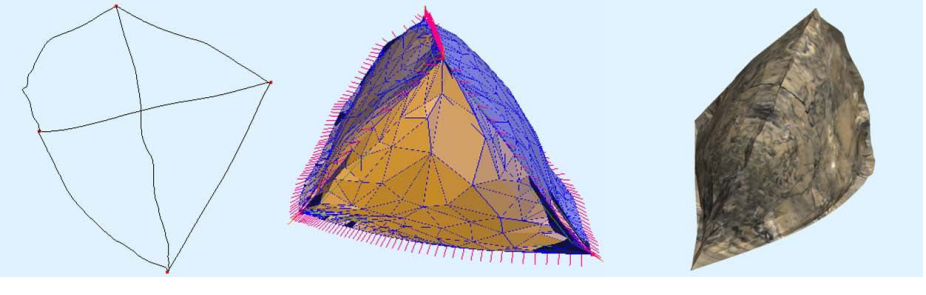

Fig. 6. One tetrahedron model generated by our system: the input sketches(left), the generated mesh(middle) and the texture-mapped appearance(right)

5 Experimental Results

We have tested our approach generating a number of simple models. Fig.5 and Fig.6 show some of the sketched-based models accursed using our interactive method. Most of the test cases are used with normally six to ten curves, producing a similar number of the simple faces on the model surface.

Our system provides a simple interface where the user can draw lines, rotate the view and continue drawing until the desired shape is achieved. Basic utilities like picking, moving and smoothing curves, or indicating corner points, are part of the interface. The identification of the simple faces on the mesh is automatically completed, but still needs the user's indication in some unusual cases. So the recognition work will be improved to be further self-acting and more interactive in future. In order to reduce the effect of the possible outliers in the surface fairing procedure, we damp the migration of vertices, only allowing displacements proportional to the length of surrounding edges. This way the relative movement of vertices is controlled by a user-defined parameter. The surface generated gives a coarse representation of the object and can be further refined using subdivision method.

6 Conclusions and Future Works

In this paper, we have also also proposed a conceptual framework for the development of systems for modeling topologically consistent meshes. We combine the graph rotation system representation and the harmonic interpolation scheme into one integrated system, in which the graph rotation system is used for an internal face recognition for models, while the harmonic interpolation is used for surface generation. We also provide efficient topological operations for edge insertion and deletion, and vertex and face tracing. In particular, we have also explored the physically-based method(FEM) to utilize the Laplacian interpolation to shape the mesh from the silhouette and profile curves.

Our planned future work includes more intelligent and automatic shape identification for surface generation, e.g., currently we are trying to use heuristics to determine the more plausible shape to be generated. In addition, future investigation will attempt to improve the surface reconstruction algorithm, including extensions to sketches that cannot be described by a connected graph. Finally, we believe that developing efficient ways to couple topology-guidance with geometrical optimization techniques is a fertile ground for future research.

References

1. Das, K., Diaz-Gutierrez, P., Gopi, M.: Sketching free-form surfaces using network of curves. In: Proceedings of EUROGRAPHICS Workshop on Sketch Based Interfaces and Modeling (SBM05), Eurographics Association (2005)
2. Bartels, R.H., Beatty, J.C., Barsky, B.A.: An Introduction to splines for use in Computer Graphics and Geometric Modeling. Morgan Kaufmann, San Francisco (1987)

3. Loop, C., Rosa, T.D.: Generalized b-spline surfaces with arbitray topology. Computer Graphics 24, 101–165 (1991)
4. Loop, C.: Smooth spline surfaces over irregular meshes. Computer Graphics 28, 303–310 (1994)
5. Galyean, T.A., Hughes, J.F.: Sculpting: An interactive volumetric modeling technique. In: Proceedings of SIGGRAPH 1991. Computer Graphics Proceedings, Annual Conference Series, ACM, ACM Press / ACM SIGGRAPH, pp. 267–274 (1991)
6. Murali, T.M., Funkhouser, T.A.: Volume sculpting. In: Proceedings of ACM Symposium on Interactive 3D Graphics 1995, pp. 151–156. ACM Press, New York (1995)
7. Cutler, B., Dorsey, J., McMillian, L., Muller, R., Jagnow, R.: A procedural approach to authoring solid models. ACM Transactions on Graphics 21(3), 302–311 (2002)
8. Perry, R.N., Frisken, S.F.: Kizamu: A system for sculpting digital characters. In: Proceedings of SIGGRAPH, ACM, ACM Press / ACM SIGGRAPH, pp. 47–56 (2001)
9. McDonnell, K.T., Qin, H.: Dynamic sculpting and animation of free-form subdivision solids. The Visual Computer 18(2), 81–96 (2002)
10. Bærentzen, J.A., Christensen, N.J.: Interactive modelling of shapes using the level-set method. International Journal of Shape Modeling 8(2), 79–97 (2002)
11. Pugh, D.: Designing solid objects using interactive sketch interpretation. In: Proceedings of the 1992 symposium on Interactive 3D graphics, pp. 117–126. ACM Press, New York (1992)
12. Zeleznik, R.C., Herndon, K.P., Hughes, J.F.: Sketch: An interface for sketching 3d scenes. In: Proceedings of ACM SIGGRAPH 1996, pp. 163–170. ACM Press, New York (1996)
13. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A sketching interface for 3d freeform design. In: Proceedings of ACM SIGGRAPH, ACM, ACM Press / ACM SIGGRAPH, pp. 409–416 (2001)
14. Karpenko, O.A., Hughes, J.F., Raskar, R.: Free-form sketching with variational implicit surfaces. Computer Graphics Forum 21(3) (2002)
15. Karpenko, O.A., Hughes, J.F.: Smoothsketch: 3d free-form shapes from complex sketches. ACM Transaction on Graphics 25(3), 589–598 (2006)
16. Chen, J.: Algorithmic graph embeddings. Theoretical Computer Science 181(2), 247–266 (1997)
17. Letniowski, F.W.: Three-dimensional delaunay triangulations for finite element approximations to a second-order diffusion operator. SIAM Journal of Science and Statistical Computation 13, 765–772 (1992)