IEICE TRANS. INF. & SYST., VOL.E88–D, NO.3 MARCH 2005

PAPER

Dynamic and Adaptive Morphing of Three-Dimensional Mesh Using Control Maps*

Tong-Yee LEE[†a)] ***and*** **Chien-Chi HUANG**[†], ***Nonmembers***

SUMMARY This paper describes a dynamic and adaptive scheme for three-dimensional mesh morphing. Using several control maps, the connectivity of intermediate meshes is dynamically changing and the mesh vertices are adaptively modified. The 2D control maps in parametric space that include curvature map, area deformation map and distance map, are used to schedule the inserting and deleting vertices in each frame. Then, the positions of vertices are adaptively moved to better positions using weighted centroidal voronoi diagram (WCVD) and a Delaunay triangulation is finally used to determine the connectivity of mesh. In contrast to most previous work, the intermediate mesh connectivity gradually changes and is much less complicated. We demonstrate several examples of aesthetically pleasing morphs created by the proposed method.

key words: *morphing, mesh connectivity, control maps, weighted centroidal voronoi diagram (WCVD)*

1. Introduction

Three-dimensional mesh morphing is a powerful technique to create a shape transformation between two or more existing three-dimensional models. This technique has numerous applications ranging from modeling to the generation of animation sequences for the game design and movie industry. Basically, most 3D mesh morphing methods consist of two steps: 1) establishing correspondence and 2) interpolating intermediate meshes. The correspondence establishment step computes the mapping for each vertex of a source mesh to a vertex of a target mesh. The interpolation step determines the trajectories for all corresponding vertices. A very thorough survey of the previous works on 3D mesh morphing can be found in [1]. In the literature, there are two popular approaches available: merging and re-meshing. The merging approach generates a common mesh for the entire morphing sequence by overlaying source mesh with target mesh. The re-meshing approach generates a common mesh using refinement operators as known from subdivision surfaces. Therefore, both approaches have consistent mesh connectivity for the entire morphing sequence. In this manner, the size of intermediate mesh is always very tremendous. Recently, we propose a novel solution to this problem [2]. We do not need merging and re-meshing. This novel approach utilizes three primitive operations, namely vertex split, vertex removal and edge swap to gradually transform the connectivity from the source model into the target model. Therefore, there is no need of common mesh connectivity. However, this novel approach is very complicated and not easy for implementation. In this paper, the proposed scheme does not require a common mesh, either. In contrast to our previous work [2], this new approach is more intuitive and much easier to implement. Designers can easily include new strategies to control maps and thus new extensions can be easily made in their development time. However, this new method and our recent work share the same advantage over most previous approaches in term of the complexity of mesh connectivity. In the following, we survey the most related works on 3D mesh morphing. For other interesting works, please refer to [1]–[3].

Usually, the merging approach decomposes models into several corresponding patches. All paired patches are aligned by features first and then are overlaid to generate a merged mesh. For simplicity, the overlay is always performed on a 2D parametric domain. Therefore, it is required to parameterize each 3D mesh patch onto a 2D embedding. The 2D overlay problem is well known in computational geometry and several optimal algorithms are available [4]. However, the mesh overlay usually produces many times as many triangles and vertices as the input models [3], [5]–[9]. Kanai et al. [8] decompose models into corresponding patches using their approximate shortest path algorithm and employ a harmonic mapping scheme to parameterize each patch before merging. Gregory et al. [5] describe a control-mesh method to dissect models and propose a greedy area-preserving mapping to parameterize each patch. Later, several related approaches further propose improvements over [5], [8] in many respects such as feature alignments for better correspondence [3], [6], [7], optimal mesh merging [3], [7] and alternative patch topology such as cylinder-like patches instead of disk-like ones [6], [9]. Manually partitioning [3], [5], [6], [8] models is not an easy task for animators. Shlafman et al. [9] describe an automatic method of dissecting models for morphing applications. Recently, Zhao et al. [10] present a component-based morphing framework that enhances user control through the whole morphing process. Furthermore, the manual effort of partitioning models is alleviated by their component-based approach. In addition, Lee et al. [11] utilize their MAPS scheme to generate coarse models. The correspondence of the finer models is computed by going through the coarse models and MAPS algorithm.

Manuscript received September 6, 2004.

[†]The authors are with the Computer Graphics Group in Visual System Laboratory at Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan.

*This paper is supported by the National Science Council, Taiwan, under contract No. NSC-93-2213-E-006-026.

a) E-mail: tonylee@mail.ncku.edu.tw

DOI: 10.1093/ietisy/e88–d.3.646

Alternatively, Michikawa et al. [12] and Praun et al. [13] utilize re-meshing technique for mesh morphing. To establish a common morph mesh, the re-meshing technique usually requires patches to be parameterized and a surface-fitting task to be performed. Alexa [1] points out that the re-meshing approach is more attractive for mesh morphing than the merging approach. However, in particular, for sharp features, the advantage can be significantly reduced since the re-meshing can require a great number of refinements to achieve the desired accuracy.

Both merging and re-meshing techniques have the same problem of producing an intermediate mesh with tremendous size. In this paper, the main contribution is a new approach to solve this problem. In term of complexity such as vertices and faces, the proposed method generates much simpler intermediate meshes than both merging and re-meshing schemes. Furthermore, this new approach is very intuitive and easy to implement. Designers can easily include new strategies in the proposed framework and thus new extensions can be easily made in their development time. The proposed approach utilizes the idea of control maps [14] to dynamically and adaptively change the connectivity of intermediate meshes. The control maps are defined in 2D parametric domain and are used to schedule vertices insertion and removal. Furthermore, the positions of vertices can be adaptively changed using weighted centroidal voronoi diagram (WCVD) [15] based on control maps. The rest of our paper is organized as follows. Section 2 overviews the proposed system framework. The proposed techniques are presented in Sect. 3. The proposed schemes are experimentally evaluated in Sect. 4. The conclusion and future work are presented in Sect. 5.

2. System Overview

The system overview of the proposed approach is shown in Fig. 1. This approach consists of six steps: 1) models are manually partitioned into paired patches by the animators, 2) each 3D surface patch is parameterized onto a 2D square embedding, 3) extra feature vertices are selected to align the features within corresponding embeddings, 4) and 5) control maps are used to determine the vertex positions in the intermediate meshes, and 6) 2D Delaunay triangulation is finally used to determine the connectivity of intermediate meshes and then the linear interpolation is performed to compute morphs. For steps 1 ∼ 3, most 3D mesh morphing methods are very similar and are well known in morphing literature. As a comparison study, we employ our previous method [3] for these three steps in this paper. Other methods such as [5], [6], [8] can be easily integrated, too. The remaining 4 ∼ 6 steps are described in Sect. 3.

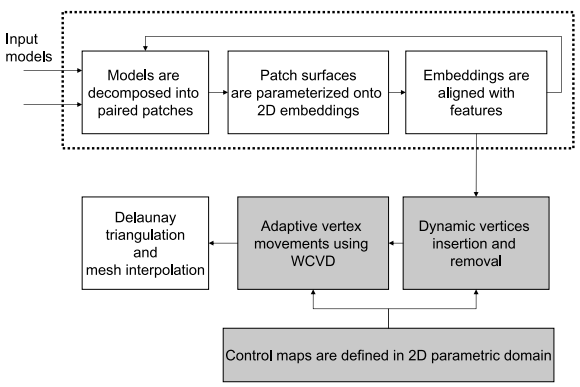


Fig. 1 System overview.

For the completeness, we briefly summarize 1 ∼ 3 steps used in [3] as follows. First, the animators select several corresponding vertex pairs on both input meshes to define corresponding patch pairs. Then, the input models are automatically dissected as an example shown in Fig. 2 (left). Second, we parameterize each patch onto a square embedding using a relaxation-based method. Lee et al. [3] utilize the following equation to relax a non-boundary vertex p_i.

$$p_i' = (1-\lambda)p_i + \lambda \frac{\sum_{j=1}^{C_i}(\omega_j p_j)}{\sum_{j=1}^{N_i}\omega_j} \tag{1}$$

In Eq. (1), p_j is p_i's 1-ring neighbor, ω_j is weight and λ is a constant to control relaxation speed, C_i is the number of p_i's 1-ring neighbors. Equation (1) can be become a linear sparse system and solved efficiently by a biconjugate gradient method. Figure 2 (right) shows an example of patch parameterization using Eq. (1). Third, for better vertex correspondence within each embedding pair, several extra feature vertices are specified and a weighted radial basis warping function is employed. Figure 3 (left) shows three feature vertices on a corresponding patch pair and Fig. 3 (right)

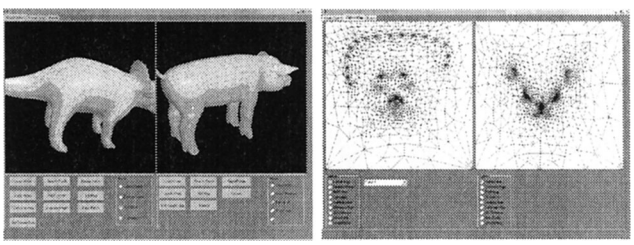


Fig. 2 Left: Two inputs are partitioned into two different patch pairs and each pair is displayed using different colors. Right: the paired patches (i.e., blue color) are parameterized onto two square embeddings. Note features such as eyes are not well aligned in this pair.

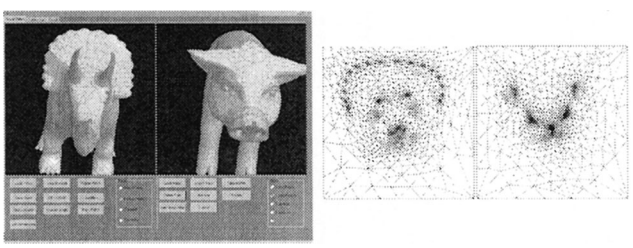


Fig. 3 Left: three extra points (red color) are selected on the corresponding patches. Right: two corresponding embeddings after feature alignments. Note that the feature alignment (i.e., eyes and nose) is better than Fig. 2 (right).

shows an example of an aligned embedding pair after warping.

3. Dynamic and Adaptive Control

Once a surface parameterization is found, Alliez et al. [14] compute several scalar maps to serve as a complete substitute for the input geometry. These maps can be easily combined to interactively control the sampling of geometry remeshing. In this paper, we borrow the idea of control maps and apply this idea to the morphing applications. With this novel application, we solve the problem occurred in merging and re-meshing morphing approaches.

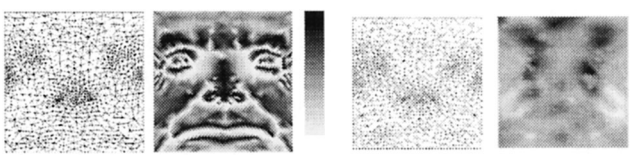
3.1 Control Maps

For our morphing application, we compute three control maps. Of course, other additional maps such as normal, colors can be also created if necessary.

Curvature maps: Once a surface parameterization is found, we can easily find a corresponding curvature map in the 2D parameterization like [14]. Figure 4 (a) shows an example of a man head in Fig. 10 and its mean curvature map. For a better contrast, the control map is visualized using gray color.

Area deformation maps: Once two corresponding patches are parameterized and aligned, we know where a vertex v_i on a source embedding is mapped onto a target embedding. First, we compute the sum of 1-ring triangle's surface area of v_i in 3D and this area sum is denoted as A_i^S. Second, for each v_i and its 1-ring neighbors, we can find all their correspondences on the target embedding. Similarly, on the target embedding, we compute corresponding area sum in 3D termed as A_i^T. To create a piecewise constant map indicating how an area is deformed, we compute the ratio A_i^T/A_i^S for each v_i and we store all ratio information in the area deformation map. If the ratio value becomes larger, it implies that the surface area is increasing or expanding in the entire morphing sequence. Therefore, we had better increase samplings in this area to avoid underestimated change in shape. On the other hand, we need to reduce its sampling density if this area is shrunk to avoid overestimated change in shape. The area deformation maps are different from area distortion maps in [14] that indicate how each triangle has been shrunk or expanded during surface parameterization. Figure 4 (b) shows an example of a man head and its deformation map using embeddings from child and man heads in Fig. 10. In this figure, the darker area indicates the larger ratio value on the deformation map. We create two area deformation maps for each corresponding patch pair.

Distance maps: For each vertex v_i on a source embedding, we know its correspondence position v_i' on a target embedding. We can first compute their 3D distance and then normalize this distance. This normalized value is stored on the distance map.



(a) (b)

Fig. 4 Curvature map (a) and area deformation map (b) for a man head model.

3.2 Scheduling Vertex Insertion and Removal

Our morphing approach gradually changes the mesh connectivity in the entire morphing sequence. Ideally, the features of two input meshes must be well maintained and be gradually transformed. The vertices of source model gradually disappear and the vertices of target model gradually appear. We use the control maps to schedule an executing order for the vertex removal of source mesh and the vertex insertion of target mesh. Figure 5 (top row) shows an area deformation map and a distance map, as well as a composition of the two maps using a per-pixel multiplication. This map is used to indicate the amount of 3D shape deformation. Furthermore, we also take features into account by linearly combining it with the curvature map as shown in Fig. 5 (bottom row). This new composition map is denoted M_v^S for a source patch. Similarly, another map called M_v^T is created for a corresponding target patch.

Next, the source vertices are sorted according to M_v^S in an increasing order and the target vertices are sorted according to M_v^T in a decreasing order. This arrangement ensures that the feature vertices in the source model have higher preservation priorities and the features in the target have higher creation priorities. Assume that source vertices are sorted in an order of $V_1^S, V_2^S \ldots V_m^S$ and m is the number of vertices on a source patch. The proposed scheduling algorithm is described as follows. First, we calculate the summation, $P = \sum_{i=1}^{m} M_v^S[V_i^S]$, where $M_v^S[V_i^S]$ is the value of stored at the map M_v^S corresponding to V_i^S. The summation P can be logically considered as the summation of deformation. To generate N intermediate frames between the source and target meshes, denoted as $f_1, f_2 \ldots f_N$, we approximately divide source vertices into N vertex sets, de-

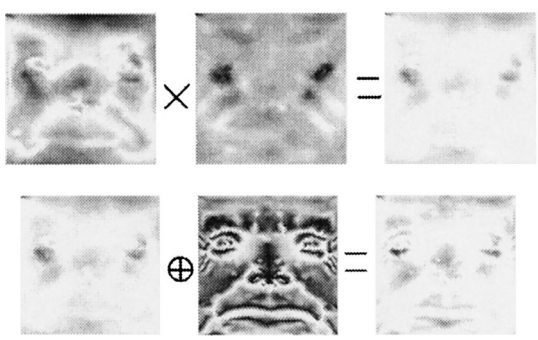


Fig. 5 On the top row, we compose distance and deformation maps using a per-pixel multiplication. Then, this newly composed map is linearly merged with the curvature map into the final control map for determining the vertex insertion and removal on the bottom row.

noted as $S_1, S_2 \ldots S_N$. Let us initialize a variable *sum* to be zero. We determine the vertex sets starting from S_1 by adding $M_v^S[V_i^S]$ into *sum* in an order of $V_1^S, V_2^S \ldots V_m^S$ one-vertex by one-vertex until *sum* reaches a bound P/N for each vertex set. Then, we determine the next vertex set and let sum be zero again. Once a vertex is assigned, it will not be assigned to other vertex sets. This process is continued until all V_i^S are assigned. To generate $f_1, f_2 \ldots f_N$, starting from S_1 to S_N, we remove all vertices belonging to S_n at frame f_n. In the same manner, we schedule the target vertices $V_1^T, V_2^T \ldots V_{m'}^T$ to appear in $f_1, f_2 \ldots f_N$, where m' is the number of target vertices. Using the above scheduling approach, the most salient feature vertices (i.e., with a large deformation or curvature) of source model will last until the end of morphing. On the other hand, the most salient target vertices will start to appear in the beginning of the morphing. Therefore, both features of input models are always preserved in the morphing sequence.

3.3 Adaptive Determination of Vertex Positions

We create a temporary parameterization map called M_{working} that contains all vertices information on M_v^S. In the morphing sequence, vertices $V_1^S, V_2^S \ldots V_m^S$ are gradually removed from M_{working} and simultaneously vertices $V_1^T, V_2^T \ldots V_{m'}^T$ are gradually inserted into M_{working} according the proposed scheduling policy in Sect. 3.2. To compute each frame f_n, we execute a 2D Delaunay triangulation over those vertices on M_{working}. The Delaunay triangulation determines the connectivity of each intermediate mesh. Thereafter, for all vertices on M_{working}, we can easily compute their corresponding 3D vertex coordinates using M_v^S and M_v^T parameterization information, and barycentric mapping. To adaptively move vertex positions, we use an additional control map called $M_v^{I_t}$ where $M_v^{I_t} = (1-t) * M_v^S + t * M_v^T$ by a pixel-by-pixel linear intensity interpolation and $0 \le t \le 1$. We use weights stored in $M_v^{I_t}$ to adaptively move vertex positions on M_{working}. Instead of using a half-toning technique [14], we use a weighted centroidal voronoi diagram (WCVD) [15] to move vertices due to the easier control of the number of vertices than half-toning [14]. Given an exact number of vertices, we can use WCVD to distribute these given vertices according to weights on $M_v^{I_t}$. Figure 6 shows a sequence of control maps $M_v^{I_t}$ and Fig. 7 shows a pseudo code of WCVD.

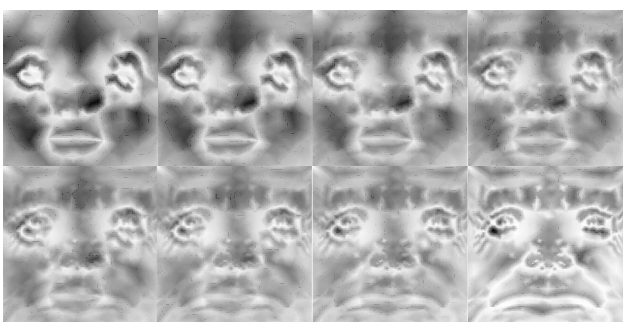


Fig. 6 A sequence of control maps $M_v^{I_t}$ for the morphing from a child to man head.

Using WCVD method, each site represents an vertex on the control map. From the sequence of Fig. 6, we can see the 2D map $M_v^{I_t}$ is gradually changing from M_V^S (i.e., child) to M_V^T (i.e., man). Therefore, we can expect the correspondicng 3D meshes are gradually transformed in the smiliar manner. Figure 8 shows an example of using WCVD to move vertices. In this figure, the darker intesity areas are sampled denser than the lighter intensity areas. In particular, after WCVD, the more number of vertices is moved to the nose area. The core of this WCVD is to compute voronoi diagram. In the current implementation, we use [16] to fast compute voronoi diagram.

Features and boundary: For input meshes, some feature vertices may be important to be preserved (i.e., can not be moved by WCVD). These feature vetices are manually selected. In addition, for maintaining consistent boundary among patches, we add some additional vertices along the boundary of the contol maps. These vetrices are static and will not be moved by WCVD, either.

3.4 Intermediate Mesh Creation and Interpolation

Once the vertices on the M_{working} are deteremined, we perform a 2D contrained Delaunay triangulation [17] over these vertices. This mesh triangulation determines the connectivity of intermediate mesh at a time t. For each vertex v on M_{working}, we know its corresponding vertices (i.e., v^S and v^T) on both source and target 2D parameteric embeddings. The vetrices v^S and v^T are then mapped into 3D using barycentric mapping. Then, we linearly interpolate them in 3D and obtain a 3D vertex coordinate of the intermediate mesh at t.

Edge Constraints: In addition to feature vertices, we

```
Procedure Weighted_Centroidal_Voronoi_Diagram
Begin
   Calculate Voronoi Diagram
   Repeat
      Move site to Weighted Centroid
      Recalculate Voronoi Diagram
   Until Convergence
End
```

Fig. 7 Pseudo code of weighted centrodial voroni diagram (WCVD).

(a) (b) (c)

Fig. 8 (a) vertices on M_{working} before WCVD; (b) control map $M_v^{I_t}$; (c) vertices on M_{working} after WCVD.

also allow the user to select several feature edges as constraints for preserving origninal edge features on both models. An example is shown in Fig. 9 for presevring the mouth edges of a child head.

4. Experimental Results and Analysis

We implemented the proposed approach and performed experiments on a PC with a Intel Pentium IV 2.2 GHz and 256 MB RAM. We experimentally compare the proposed approach with our previous work [3] in Figs. 10 and 11. For these two examples, we list some experimental statistics in Table 1 for quantitive comparsion between the proposed work with [3]. These two examples were used in [3]. The number of traingles is much smaller using the proposed method than that generated by [3]. Using the proposed method, the number of traingles varies from frame to frame. However, using [3], the number of triangles is fixed, i.e., 17138 and 57844 for all morphing meshes in child-man and cow-pig morphing examples, resepctively. These numbers are about 5 $\sim$ 15 times of those using our method. In terms of the visual apperance, there is not too much difference between two methods. Figures 12 (a) and (b) shows two additional examples created by the proposed method. Uisng [3], the number of triangles is 44670 and 43531 for Figs. 12 (a) and (b).

Finally, we should comment on the execution timings of the proposed method. On the average, the cost of determining vertex insertion or removal is very insignificant, i.e., less than 1 second per frame. The cost of WCVD ranges from 1 to 1.5 seconds per frame. The triangulation cost is various from 0.5 to 1.2 seconds per frame. These three tasks are executed per frame. This performance allows the animators to interactively create their morphing sequences. During the morphing sequence, the animators can freely add extra vertices and edges constraints to maintain desired features. All figures with color and larger size are available at:

`http://couger.csie.ncku.edu.tw/~vr/ieice/morph.htm`

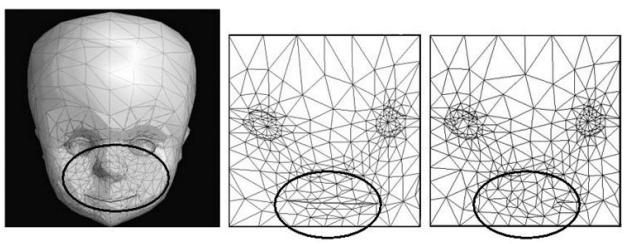
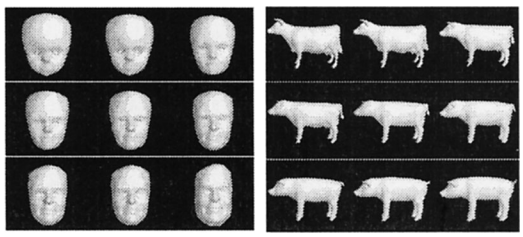


Fig. 9 Left: several feature edges are selected. Middle: Delaunay triangulation with edge constraints. Right: triangulation without edge constraints.

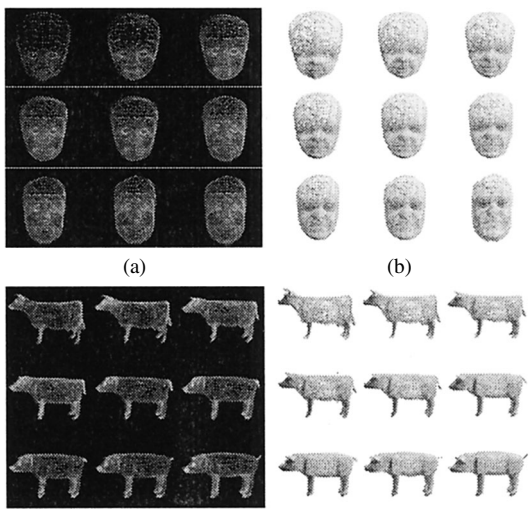
5. Conclusion and Future Work

We presented a new approach for generating a 3D mesh morphing. Using several control maps, the connectivity of intermediate meshes is dynamically changing and the mesh



(a) (b)

Fig. 10 Two morphing examples (a) child-man and (b) cow-pig.



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

Fig. 11 Wire-frame comparisons for two examples in Fig. 10. (a): by the proposed method. (b): by [3]. (c): by the proposed method. (d): by [3].

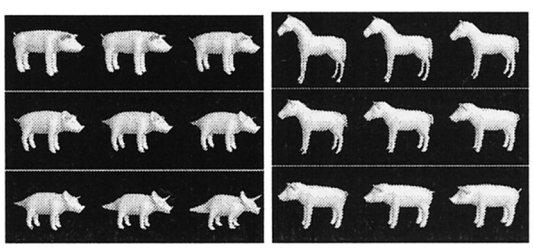


Fig. 12 Two additional examples created by the proposed schemes.

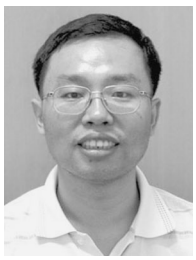
Table 1 Experimental statistics.

Keyframe #	1	2	3	4	5	6	7	8	9
Fig. 11 (a)	1148	3120	3907	4459	4899	4633	4781	4741	4697
Fig. 11 (b)	5803	10851	11804	12193	11511	9067	8353	8012	7546
Fig. 12 (a)	7964	10457	11612	12190	8410	8213	7923	7647	6393
Fig. 12 (b)	7052	10124	11431	12275	9186	8854	8610	8390	7364

vertices are adaptively modified. In contrast to traditional merging and re-meshing approaches, our approach is more promising and reasonable than other approaches in term of geometry complexity, i.e., vertices and faces. This new approach allows the user to interactively create aesthetically pleasing morphs. Many potential works can be done in near future. For example, in the current implementation, we do not optimize triangle quality such as regularity and face aspect ratio after a constrained Delaunay triangulation. In addition, we do not take spatial coherence into account as we independently triangulate the consecutive intermediate meshes.

References

[1] M. Alexa, "Mesh morphing," STAR: State of the Art Report, EUROGRAPHICS, 2001.
[2] C.H. Lin and T.Y. Lee, "Metamorphosis of 3d polyhedral models using progressive connectivity transformations," IEEE Trans. Vis. Comput. Graphics, vol.11, no.1, pp.2–12, Jan. – Feb. 2005.
[3] T.Y. Lee and P.H. Hung, "Fast and intuitive metamorphosis of 3d polyhedral models using smcc mesh merging scheme," IEEE Trans. Vis. Comput. Graphics, vol.9, no.1, pp.85–98, 2003.
[4] U. Finke and A. Hinrichs, "Overlaying simply connected planar subdivisions in linear time," 11th Annual Symposium on Computational Geometry, pp.110–126, June 1995.
[5] A. Gregory, A. State, M. Lin, D. Manocha, and M. Livingston, "Interactive surface decomposition for polyhedral morphing," The Visual Computer, vol.15, no.9, pp.453–470, 1999.
[6] M. Zockler, D. Stalling, and H.C. Hege, "Fast and intuitive generation of geometric shape transitions," The Visual Computer, vol.16, no.5, pp.241–253, 2000.
[7] M. Alexa, "Merging polyhedral shapes with scattered features," The Visual Computer, vol.16, no.1, pp.26–37, 2000.
[8] T. Kanai, H. Suzuki, and F. Kimura, "Metamorphosis of arbitrary triangular meshes," IEEE Comput. Graph. Appl., pp.62–75, 2000.
[9] S. Shlafman, A. Tal, and S. Katz, "Metamorphosis of polyhedral surfaces using decomposition," Comput. Graph. Forum, vol.21, no.3, pp.219–228, 2002.
[10] Y. Zhao, H.Y. Ong, T.S. Tan, and Y. Xiao, "Interactive control of component-based morphing," Proc. 2003 ACM SIGGRAPH/ Eurographics Symposium on Computer animation, pp.339–348, Eurographics Association, 2003.
[11] A.W.F. Lee, D. Dobkin, W. Sweldens, and P. Schröder, "Multiresolution mesh morphing," Proc. 26th Annual Conference on Computer Graphics and Interactive Techniques, pp.343–350, ACM Press/ Addison-Wesley Publishing, 1999.
[12] T. Michikawa, T. Kanai, M. Fujita, and H. Chiyokura, "Multiresolution interpolation meshes," Proc. 9th Pacific Conference on Computer Graphics and Applications, pp.60–69, IEEE Computer Society, 2001.
[13] E. Praun, W. Sweldens, and P. Schröder, "Consistent mesh parameterizations," Proc. 28th Annual Conference on Computer Graphics and Interactive Techniques, pp.179–184, ACM Press, 2001.
[14] P. Alliez, M. Meyer, and M. Desbrun, "Interactive geometry remeshing," Proc. 29th Annual Conference on Computer Graphics and Interactive Techniques, pp.347–354, ACM Press, 2002.
[15] A. Secord, "Weighted voronoi stippling," Proc. 2nd International Symposium on Non-photorealistic Animation and Rendering, pp.37–43, ACM Press, 2002.
[16] I. Kenneth, E. Hoff, J. Keyser, M. Lin, D. Manocha, and T. Culver, "Fast computation of generalized voronoi diagrams using graphics hardware," Proc. 26th Annual Conference on Computer Graphics and Interactive Techniques, pp.277–286, 1999.
[17] Shewchuk, "Triangle: Engineering a 2D quality mesh generator and delaunay triangulator," WACG: 1st Workshop on Applied Computational Geometry: Towards Geometric Engineering, WACG, LNCS, 1996.



Tong-Yee Lee received the BS degree in computer engineering from Tatung Institute of Technology in Taipei, Taiwan, in 1988, the MS degree in computer engineering from National Taiwan University in 1990, and the Ph.D. degree in computer engineering from Washington State University, Pullman, in May 1995. He is a professor in the Department of Computer Science and Information Engineering at National Cheng-Kung University in Tainan, Taiwan. He serves as a guest associate editor for IEEE Transactions on Information Technology in Biomedicine from 2000 to 2005. His current research interests include computer graphics, image-based rendering, visualization, virtual reality, distributed and collaborative virtual environment.

Chien-Chi Huang received B.S. Degree from the Department of Computer Science and Information Engineering, TamKang University, Taipei, Taiwan, in 2001, and M.S. Degree from the Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan, in 2003. His research interests include computer graphics, computer vision and image processing.