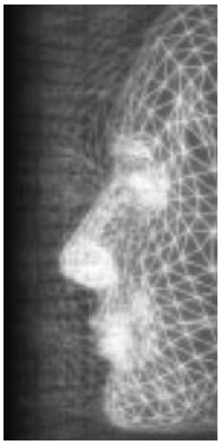
COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2005; **16**: 487–498
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cav.85



Progressive mesh metamorphosis

By Chao-Hung Lin, Tong-Yee Lee, Hung-Kuo Chu and Chih-Yuan Yao*

This paper describes a new integrated scheme for metamorphosis between two closed manifold genus-0 polyhedral models. Spherical parameterizations of the source and target models are created first. To control the morphing, any number of feature vertex pairs is specified and a fold-over free warping method is used to align two spherical embeddings. Our method does not create a merged meta-mesh or execute re-meshing to construct a common connectivity for morphs. Alternatively, a scheme for the progressive connectivity transformation of two spherical parameterizations is employed to generate the intermediate meshes. A novel semi-overlay with a geomorph scheme is proposed to reduce the popping effects caused by the connectivity transformation. We demonstrate several examples of aesthetically pleasing morphing sequences using the proposed scheme. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: metamorphosis; connectivity transformation; spherical parameterization; warping; semi-overlay; geomorph

Introduction

Three-dimensional mesh metamorphosis (or morphing) has been an active research topic in recent years. It has been widely used in many applications such as computer animation, game design, and the entertainment industry. The mesh metamorphosis technique generates a smooth transition from a source mesh into a target mesh based on considering the geometrical connectivity and materials of the two models such as position, color, and texture. In 3D mesh morphing research, the primary issue is computing vertex or triangle correspondence between the source and target meshes.

The correspondence problem in the mesh morphing is closely related to surface parameterization or called embedding.[1–6] Before surface parameterization, the models are compatibly decomposed into patches. Patches can be disk-like[1–3,5,6] or cylinder-like topology.[3,5] Then, the embedding merging is used to compute correspondence in many previous works. To bring better correspondence,[3,4,6,7] warp embeddings to align the feature vertices. However, the warping function can produce a fold-over. To avoid the fold-over, Alexa[4] attempts to warp the embedding as much as possible to align the feature vertices. Zockler *et al.*[3] employ a fold-over free warping[8] to deal with this problem. Lin and Lee[7] suggest a re-partitioning scheme to handle this problem. Alternatively, Praun *et al.*[9] and Michikawa *et al.*[10] propose re-meshing techniques to establish consistent meshes for morphing. These two works do not require mesh merging. However, the re-meshing technique requires a great number of refinements to achieve the desired accuracy (in particular for sharp features) for the input meshes.

The mentioned above approaches require a common or consistent mesh connectivity and therefore the correspondence problem is solved. However, both approaches have the disadvantage of requiring a common mesh of tremendous size.[7] Recently, Lee *et al.* propose novel solutions[7,11] to this problem. Both approaches do not require mesh merging and remeshing to establish a common mesh for morph interpolation. Alternatively, both approaches gradually change the morph mesh connectivity from the source into the target mesh. Therefore, the intermediate meshes are much simpler than those generated by previous approaches. However, they point out that the popping effects could occur due to the connectivity

*Correspondence to: T.-Y. Lee, Computer Graphics Group, Visual System Laboratory, Department of Computer Science and Information Engineering, National Cheng-Kung University, No.1, Ta-Hsueh Road, Tainan 701, Taiwan.
E-mail: tonylee@mail.ncku.edu.tw

Contract/grant sponsor: National Science Council, Taiwan; contract/grant numbers: NSC-93-2213-E-006-026; NSC-93-2213-E-006-060; NSC 94-2213-E-006-005.

transformation.[7] This would be a large disadvantage for morphing visualization.

System Overview and Contribution

A brief overview of the proposed scheme is shown in Figure 1. Our scheme proceeds by first creating spherical parameterizations of the input models. Therefore, we do not require a compatible model decomposition[1–3,5–7,9–11] which is usually a difficult and tedious task. Then, the spherical parameterizations are aligned with the feature vertices using a foldover free warping. To reduce popping effect, vertex matching and the semi-overlay feature vertices and edges are proposed before connectivity transformation. Finally, the connectivity transformation and linear interpolation with geomorph is developed to generate smooth morphs. In the proposed system, we use a spherical embedding approach[4] combined with a variant of an image foldover-free warping technique[8] to establish the correspondence between two closed manifold genus-0 meshes. The proposed method to solve foldovers by using edge-swaps can be depicted in Figure 2.

The major contribution of this paper is described as follows. A new integrated scheme for mesh metamorphosis is proposed. Without full spherical parameterization merging or re-meshing, the proposed method extends[7] to transform the spherical parameterization connectivity from the source to the target meshes. The proposed method is more reasonable in terms of the required number of vertices and faces than previous works. A novel semi-overlay with a geomorph scheme is proposed to reduce the popping effects caused by the connectivity transformation.[7]

Connectivity Transformation of Spherical Parameterizations

In this paper, we extend a recent work[7] to execute connectivity transformation of spherical parameterizations. The basic idea of this previous approach can be illustrated using Figure 3. In this figure, it first executes a sequence of VSOs (vertex split) to insert all target

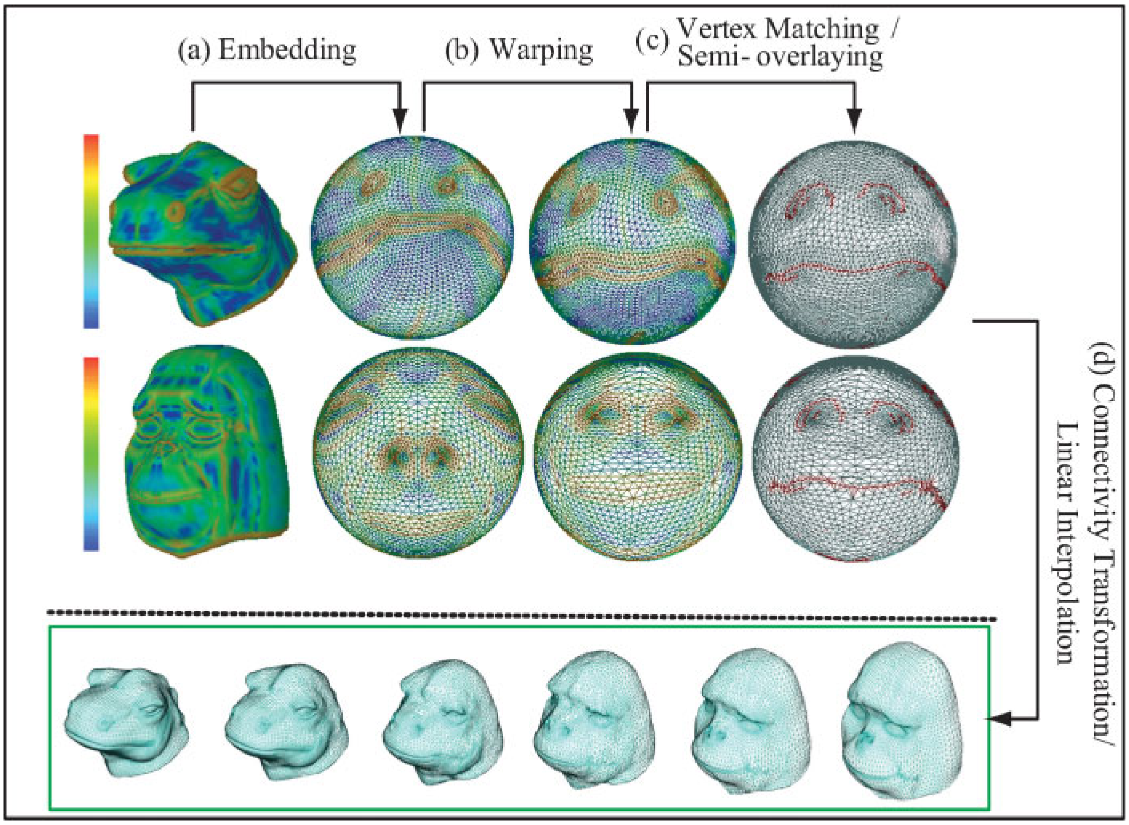

(a) Embedding
(b) Warping
(c) Vertex Matching / Semi- overlaying
(d) Connectivity Transformation/ Linear Interpolation

Figure 1. A brief overview of our scheme. Both models are colored according to curvature that increases from blue to red.

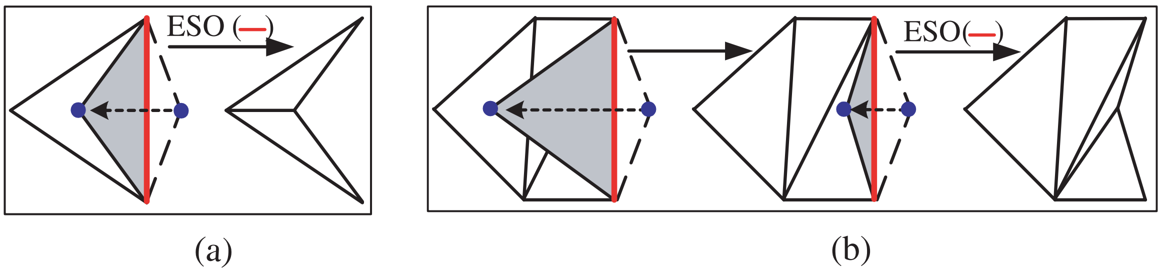


Figure 2. A simple case of face unfolding using an edge swap (ESO) is shown in (a). If a face is warped to foldover its adjacent face, the shared edge of these two faces is swapped to unfold these two faces. If the fold-over face crosses many faces as shown in (b), we first perform warping until the fold-over is simplified into a simple case, such as that shown in (a). The edge swap is then used to unfold the faces.

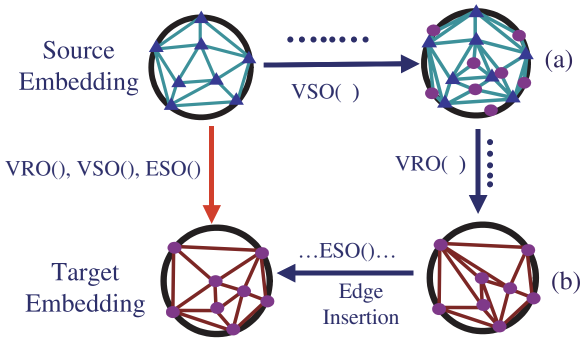


Figure 3. Progressive connectivity transformation between source and target embeddings.

vertices on the source embedding (i.e., (a)) and then it executes a sequence of VROs (vertex removal)) to remove all source vertices (i.e., (b)). While removing source vertices, the source edges are removed, too. Finally, it executes a sequence of ESOs (edge swap) to insert each target edge on the source embedding and thereafter the source embedding is transformed to the target embedding. According to some geometric metrics of vertex and edge[7] on source and target embeddings, it schedules the execution orders of VRO, VSO, and ESO operations among a sequence of morphing animation and therefore the source embedding is progressively transformed into the target embedding. In addition, to reduce the number of these three operations, the rough vertex matching is executed on embeddings. For more details, please see Reference [7].

We attempt to transform connectivity of two given spherical embeddings. Furthermore, we propose a semi-overlay (Section 'Semi-Overlaying the Embeddings') and geomorph (Section 'Geomorph') to enhance this method to reduce the popping effect. Our extension proceeds first by rough vertex matching and then merging the feature vertices/edges that are most likely to cause the popping effect on both spherical embeddings. Consecutively, similar to Reference [7], we perform three operations (VRO, VRO, and ESO) on the spheres instead of 2D embeddings. Finally, we linearly interpolate the vertices with geomorph to create morphing. To implement our spherical extension, these three operations are performed nearly the same as in Reference [7] on the 2D embeddings. Specifically, there is a major difference required when an edge is inserted using a sequence of ESOs. On the spheres, an edge is an arc (i.e., the shortest path) between two points. Therefore, we need to compute this arc and its intersections with other edges before a sequence of ESOs. We use Reference [4] approach to efficiently compute the edge intersections on spheres. Finally, we should comment that the spherical extension has a major advantage over.[7,11] On the spheres, we do not need to pay attention to the boundary problem in Reference [7,11]. Therefore, the overall algorithm becomes much terser and easier to implement.

Smooth Connectivity Transformation

Semi-Overlaying the Embeddings

We attempt to generate the intermediate meshes at a reasonable size to reduce the popping effects in the morphing sequence. The merging approach does not have a popping problem but the connectivity transformation does. To reduce popping, the semi-overlay approach combines the merging and connectivity transformation ideas. We do not fully merge two spheres but partially merge the vertices and edges that will have the most potential to cause popping. In contrast to the merging approach, we do not create too many new triangles because this method uses a semi-overlay. Specifically, the semi-overlay is performed after the rough matching rather than vice versa. This can potentially reduce the number of vertices and edges to be merged. Therefore, the semi-overlay does not increase a great number of triangles.

A mesh M is described by a pair (K, V), where K is a simplicial complex representing the connectivity of vertices, edges, and faces; V describes the geometric positions of the vertices in R^3. In the general setting of 3D morphing, two meshes $M_0 = (K_0, V_0)$ and $M_1 = (K_1, V_1)$ are given. After the correspondence of the input meshes is built, the corresponding vertex pairs on the input meshes can be obtained by a bijective mapping function ϕ (see Figure 4). Then the intermediate meshes $M(t) = (K(t), V(t))$, $t \in [0,1]$ can be obtained using shape and connectivity transforming. The vertex pair interpolation (i.e., shape transformation) does not cause popping effects. On the other hand, both $K(t)$ and $\|V(t)\|$ (i.e., number of vertices) are different between consecutive frames and therefore popping effects are created.[7]

In this paper, we quantify the popping effects as the distance variations between $M_0' = (K_0, \phi(V_0))$ and $M_1 = (K_1, V_1)$. An objective function is defined as Equation (1) that represents the distance variations between M_0' and M_1. To reduce this objective function, we propose a semi-overlay strategy. First, all vertices for the input meshes are evaluated using a function Equation (2) for the source vertices and Equation (3) for the target vertices. Both equations are formulated as curvature variations multiplied by the face area. All vertices from both models are then sorted according to Equations (2) and (3) into a single vertex queue in a decreasing manner. Once a new vertex is inserted on either the source or target embedding, both embeddings have this vertex pair and, therefore, an objective function Equation (1) is reduced. However, this insertion generates new triangles on either the source or target embedding. Therefore, a trade-off must be made between popping and the complexity of the intermediate meshes. Simply, we set up a threshold on Equation (1) and repeat selecting a vertex from the vertex queue until the threshold is reached. Once a vertex is selected from an embedding, it is inserted onto the counterpart embedding. Furthermore, if an edge $\{v_i, v_j\} \in K_0 \bigcup K_1$ and vertices v_i and v_j have been inserted, then this edge must also be inserted. In this manner, the semi-overlay inserts both the vertices and edges that have the greatest potential to cause popping effects.

$$E = \sum_{x_i \in X} \mathrm{dist}(x_i, \phi(x_i)) \tag{1}$$

Where $x_i \in R^3$ and this belongs to a finite set X that accurately fits mesh M_0'.

$$C_{M_0'}(v_i) = |\mathrm{cur}_{M_0'}(v_i) - \mathrm{cur}_{M_1}(\phi(v_i))| \times \mathrm{area}(T_1) \tag{2}$$

$$C_{M_1}(v_i) = |\mathrm{cur}_{M_0'}(\phi^{-1}(v_i)) - \mathrm{cur}_{M_1}(v_i)| \times \mathrm{area}(T_0) \tag{3}$$

Where $\mathrm{cur}_M(v)$ represents the mean curvature of vertex v on mesh M; triangles $T_0 \in K_0$ and $T_1 \in K_1$ (see Figure 5). The position of $\phi(v_i)$ is located at the interior, an edge or a vertex of a triangle. To compute area(T_1), we average two triangle areas sharing this edge or we average 1-ring triangle area sharing this vertex for edge and vertex cases, respectively. We compute area(T_0) in a similar manner. The feature edges, i.e., creases where the geometry has a discontinuous tangent plane, preservation is important for a smooth morph. Reference [7] assign a high priority for feature edges. However, the feature edges will still be handled by ESO or VRO in a morphing sequence and a visual discontinuity could be produced. Therefore, the feature edges in both the input meshes are also defined as candidates to perform a semi-overlay. A simple thresholding technique is adopted to detect the feature edges. If the normal vector variation of the neighbor faces of an edge is larger than a user-defined threshold, this edge is identified as a feature edge. We will also merge these feature edges in semi-overlay.

Figure 5 shows the semi-overlay results. The red regions in Figure 5 (c–e) represent the merged vertices or edges. This also implies that these regions tend to cause popping effects in a morphing sequence if local merging is not performed (i.e., semi-overlay). After performing local merging, the variation between M_0' and M_1 are reduced (especially at the nose and the neck).

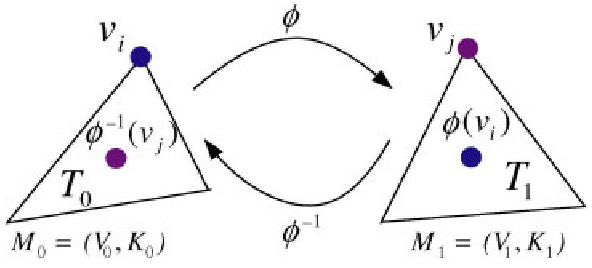


Figure 4. A bijective mapping function ϕ between two meshes M_0 and M_1.

Geomorph

Hoppe[12] presents a nice property of the connectivity transformation, that is, the geomorph. The geomorph

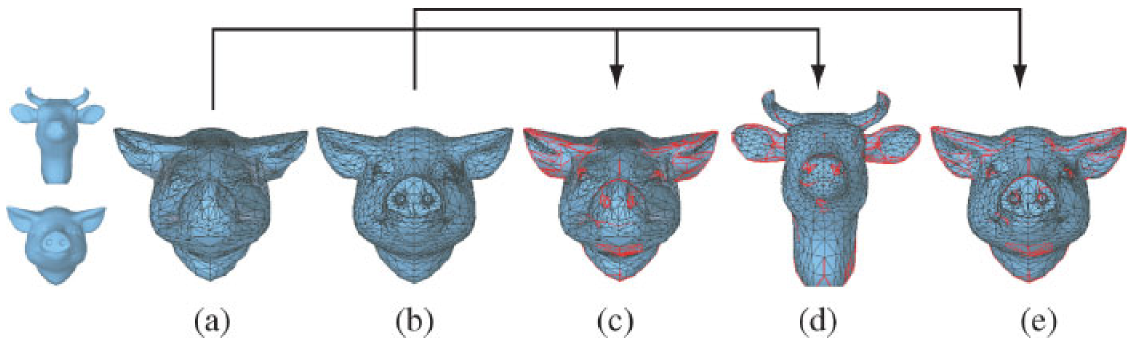


Figure 5. The semi-overlay of a cow's head M_0 and a pig's head M_1. Both M'_0 and M_1 are shown in (a) and (b), respectively. The semi-overlay results for M'_0, M_0, and M_1 are shown in (c), (d), (e), respectively.

is a smooth visual transition between two successive meshes $M(t)$ and $M(t+\Delta t)$, i.e., $M(t)\xrightarrow{\text{VRO,VSO}}M(t+\Delta t)$, where Δt is a time step. In this paper, we propose a new geomorph approach to reduce the popping effect while the connectivity of spherical embeddings is changing. The primary idea is to generate the geomorph by $M(t)\xrightarrow{\text{VRO,VSO}}M(t+\Delta t\times\delta)$, where δ is the number of successive frames to execute a VRO or VSO operation. In other words, we attempt to make the popping effect caused by each operation dispersed among the successive δ frames. Using such a geomorph approach, a smoother transition can be obtained and the popping effects in a morphing sequence can be further reduced. Figure 6 demonstrates an example of a geomorph zoom-in viewed from a morph between a cows head and a pig's head. In this example, we demonstrate how a VRO operation on an apex vertex v is handled by the geomorph. Without the geomorph, this apex vertex v is removed at t as VRO is scheduled to remove it. In contrast, with the geomorph, we do not remove it until $t+\Delta t\times\delta$ and linearly interpolate its position before its removal. In this example, the popping effects are dispersed on average among four successive frames (i.e., $\delta=4$). As a result, a smoother transition is obtained as shown in Figure 6(b). A VSO on a vertex v' can be handled in a similar manner. Take Figure 6 as an example again but in inverse in terms of t. Without the geomorph, v' is inserted and splitting triangles, and then v' becomes v at t. With the geomorph, v' inserted and splitting triangles at t, and v' and v is linearly interpolated. Finally, at $t+\Delta t\times\delta$, v' becomes v. The visual effect is similar to Figure 6(b) but in inverse order. We can expect vertex v' to be gradually pumped out and becomes v.

In addition to VRO and VSO, the geomorph can also be applied to edge insertion. In Reference [7], they show that an edge can be inserted into an embedding using a sequence of valid edge swaps. However, executing a sequence of edge swaps on a local mesh region could cause the popping effects. There are two possible approaches to solving this problem. The first approach is to distribute the sequence of edge swaps among several successive frames (as shown in Figure 7(a)). The second approach is that a sequence of edge swaps is replaced by a sequence of vertex split operations and a sequence of vertex removal operations (as shown in Figure 7(b)). These substitution operations (i.e., VRO and VSO) are then handled by a geomorph. A transition generated using the second approach could be potentially smoother than those generated using the first approach, because the first approach must execute a discontinuous transformation, i.e., edge swap. However, the drawback of the second approach is that there too many VRO and

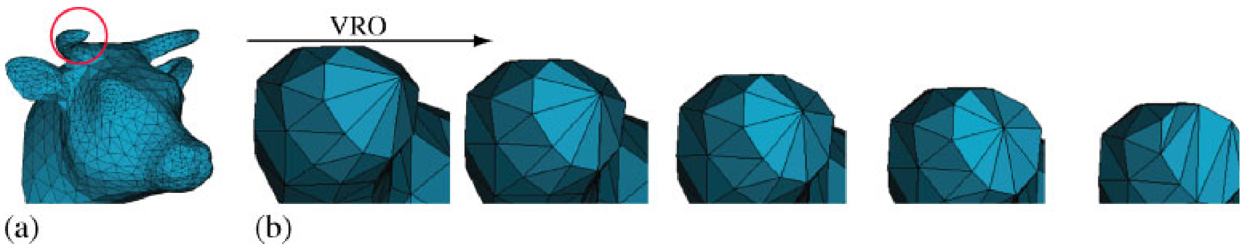


Figure 6. The geomorph for the vertex removal. A smoother visual transition is shown in (b); (b) is the enlargement of the red circle shown in (a).

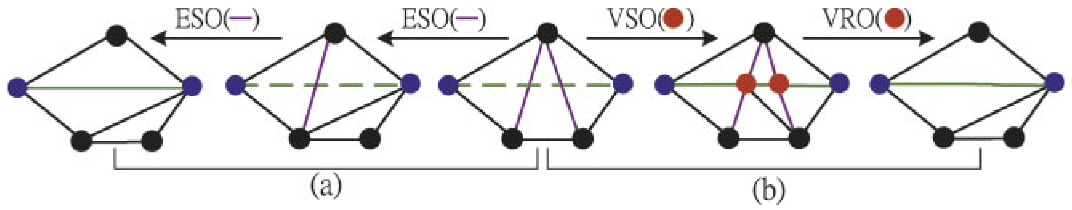


Figure 7. Two approaches for executing an edge creation.

VSO operations are required. Therefore, the first approach seems more efficient than the second approach. Furthermore, the number of vertices in the intermediate meshes generated using the first approach is fewer than those generated using the second approach. Therefore, our geomorph adopts the first approach to insert an edge.

The *Connectivity_Transformation()* algorithm shows the pseudo code for the proposed techniques including the semi-overlay and connectivity transformation with geomorph. In the beginning of this algorithm, rough vertex matching is performed in the *VertexMatching()* procedure. The semi-overlaying described in Section 'Semi-Overlaying the Embeddings' is performed in the *Semi-Overlaying()* procedure. Each vertex and each edge priority in both embeddings are then calculated in the *PriorityCalculation()* procedure. In Reference [7], the two vertices on two embeddings that are closet to one another are termed as a matched pair in the Vertex-Matching() procedure. The vertex priority is defined as the vertex curvature multiplies by the face areas that surround the vertex. The edge priority is defined as the normal variations of the neighbor faces of this edge multiply by the areas of these faces. The priorities are sorted and then normalized (i.e., various from 0 to 1) to relate the priority to the frame time t, $0 \le t \le 1$.

```
Algorithm Connectivity_Transformation($M_0$, $M_1$)
{
  VertexMatching();
  Semi-Overlaying();
  PriorityCalculation(); /*Calculate the priority of each vertex and each edge */
  For each matched vertices $v_i$
    $v_i$.priority ← ∞; /*Assign an initial value to each matched vertices */
  For $t \leftarrow 0$ to 1 {
    Foreach $v_i \in V(t)$ /*The intermediate mesh $M(t) = (V(t), K(t))$*/
      If ($v_i$.priority $\le t - \Delta t * \delta$)
      /*$\Delta t$ is time step; $\delta$ is the number of frames to perform geomorph*/
        VRO ($v_i$); /*Vertex removal operation from $M(t)$*/
  For each $v_i \in V_1$ /*The target mesh $M_1 = (V_1, K_1)$*/
      If($v_i$.priority $\le t$)
        VSO ($v_i$);
        /* Vertex split operation: add vertex from $V_i$ to $V(t)$*/
  EdgeInsertionWithGeomorph();
  /* Edge insertion with geomorph: insert edge from $K_1$ to $K(t)$*/
  InterpolationWithGeomorph ($M(t)$);
  /* Generate the intermediate mesh $M(t)$ using
    linear interpolation with geomorph* /
  MeshRefinement ($M(t)$);
  /* Refine the intermediate mesh$M(t)$*/
  Output ($M(t)$);
  /* Output the intermediate mesh$M(t)$*/
  $M(t + \Delta t) = M(t); t = t + \Delta t$;
  }
}
```

Therefore, in *Connectivity_Transformation()*, we can schedule the VRO, VSO, and edge creation execution orders in term of t, i.e., vertex or edge priority.

In *Connectivity_Transformation()*, we employ the proposed geomorph technique to all VRO, VSO, and edge insertions. Geomorph is performed during several successive frames to obtain a smooth visual transition. In the *Connectivity_Transformation()* algorithm, the

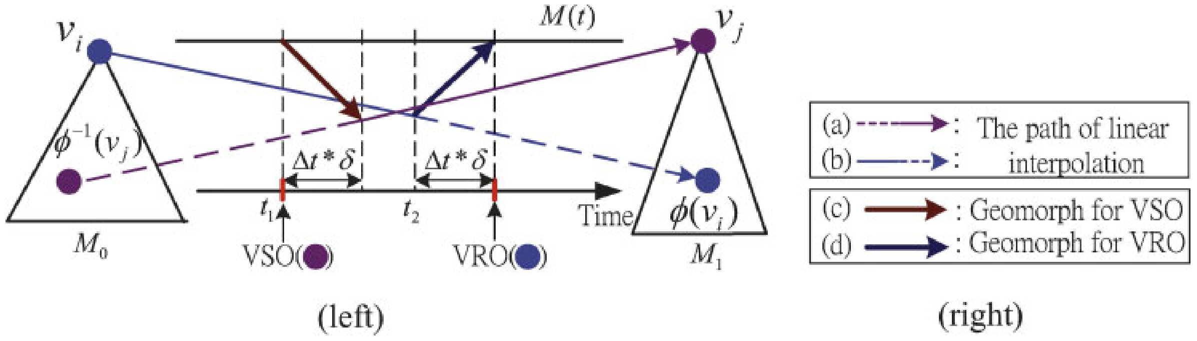


Figure 8. A sketch of the geomorph for VSO and VRO. The paths (a) and (b) show the linear interpolation between two correspondence vertices. The path (c) shows the VSO geomorph. The path (d) shows the VRO geomorph.

vertex and edge priorities are critical to determining when the geomorph is started and ended for each operation. Initially, the priority is set to ∞ (i.e., infinite) for all matched vertices produced by both the *VertexMatching()* and the *Semi-Overlaying()* procedures. These matched vertices will not necessarily perform VSO or VRO in the whole morphing time. In other words, the matched vertices will not necessarily be interpolated with the geomorph. These matched vertices are only linearly interpolated. Only the unmatched vertices require the geomorph. Once these unmatched vertices become matched and finish the corresponding geomorph interval (i.e., $\{t \sim t + \Delta t \times \delta\}$), they are linearly interpolated until the end of the morphing sequence like those matched vertices using *VertexMatching()* and the *Semi-Overlaying()* procedures.

In the connectivity transformation, if the priority of the unmatched vertex is equal to the time frame t, the VRO or VSO will be performed with the geomorph for this unmatched vertex. The VRO or VSO geomorph is handled in the InterpolationWithGeomorph() procedure. For the VRO, a vertex $v_i \in V(t)$ is removed until the time frame equal to $v_i.priority + \Delta t \times \delta$. In the $\{t \sim t + \Delta t \times \delta\}$ time interval, a geomorph for the VRO is performed (as shown in Figure 8(d)). This unmatched vertex v_i is linearly interpolated with its corresponding vertex v_{pos} belonging to the intermediate mesh $M(t = (v_i.priority + \Delta t \times \delta))$. For the VSO, a vertex $v_i \in V_1$ is inserted into $M(t)$ and set $v_i.match = true$ when the time frame t is equal to $v_i.priority$. In the $\{t \sim t + \Delta t \times \delta\}$ time interval, a geomorph for VSO is performed (as shown in Figure 8(c)). This inserted vertex is linearly interpolated with its corresponding vertex v_{pos} belonging to the intermediated mesh $M(t = (v_i.priority + \Delta t \times \delta))$.

```
Procedure InterpolationWithGeomorph($M(t)$)
{
For each $v_i \in V(t)$ {
  /* For matched vertices */
  If $v_i.match = true$ {
    /* Geomorph for VSO */
    If ($v_i.priority < t$) and ($v_i.priority \geq t - \Delta t * \delta$)
      $v_i \leftarrow (v_i.priority + \Delta t * \delta - t) * v_0 + (t - v_i.priority) * v_{pos}$
      /* Where $v_0 \in M(t = (v_i.priority))$ $v_{pos} \in M(t = (v_i.priority + \Delta t * \delta))$
        and $v_i$ correspond to $v_0$ and $v_{pos}$ */
    Else
      $v_i \leftarrow (1 - t) * v_i^0 + t * v_j^1$;
      /* Where $v_i^0 \in V_0, v_j^1 \in V_1$ and $v_i^0$ match with $v_j^1$ (linear interpolation) */
  } Else { /* For unmatched vertices */
    /* Geomorph for VRO */
    If ($v_i.priority \leq t$) and ($v_i.priority > t - \Delta t * \delta$)
      $v_i \leftarrow (v_i.priority + \Delta t * \delta - t) * v_0 + (t - v_i.priority) * v_{pos}$
    Else
    /* Where $v_i^0 \in V_0$ and $\phi(v_i^0) \in M_1$ */
      $v_i \leftarrow (1 - t) * v_i^0 + t * \phi(v_i^0)$;
    }
  }
}
```

In the *EdgeInsertionWithGeomorph()* procedure, an edge $e_i \in K_1$ is inserted into the intermediate mesh $K(t)$ if the priority for this edge is smaller than or equal to the time frame t. The edge insertion consists of a sequence of valid edge swaps. For performing an edge insertion geomorph, only one edge swap is performed per frame at the $\{t \sim t + \Delta t \times p\}$ time interval, where p is the number of edge swap for inserting a new edge. This procedure is very similar to *FirstPassEdgeSwap()*[7] except that we disperse the edges swaps among p frames. Therefore, the pseudo code is omitted here.

Experimental Results

We have implemented our system on a Pentium 4 2.2 G PC with 512 M memory. Many pleasing morphing sequences have been generated using the proposed morphing scheme. These examples are shown in Figures 1, 11–13. In this paper, the warping function proposed by Reference [4] is used to align feature vertices and the proposed method is used to solve the foldover during the warping process. With edge swaps, the feature vertices, in general, are much better coincident than Reference [4]. The feature coincidence is important for the morphing applications. If the feature vertices cannot be well aligned, a strange and unexpected morphing result could be generated. For example of the Figure 9(a), the hoofs of the input models are not well aligned and the morphing sequence is strange. A better result generated by the proposed warping method is shown in Figure 9(b). In addition, we find the warping with edgeswap perform much faster than the original method[4] as shown in Table 1.

In Figure 10, we show different thresholds on the parameter E of Equation (3) and obtain different number of vertices in both models. We do not want to increase too many vertices and triangles in our experiments. Therefore, we always select a threshold with an increase of the number of vertices below 10% of the original input models. For example, in Figure 10, we will select cases between (a) and (b).

We demonstrated several pleasing morphs and the connectivity transformations in Figures 11–13. To demonstrate the robustness of the proposed morphing scheme, several examples of the input models with large different shapes or data sizes are selected. In addition, we add many features up to 93 to control morphs in these examples. In the example of the Figure 12, the input model is very different in shapes. For example, the horse feet are very noticeable but rabbit does not. The example of Figure 13 is another extreme case. Both the input models have large data sizes. The number of vertices in the source and target models is up to 44 955 and 50 002, respectively. In the examples of the Figures 11–13, not only the shape and connectivity transition is generated, the color material is also be generated by linear color blending. The example of the Figure 12 is similar to References [7,13]. Lee *et al.*[13] and Lin *et al.*[7] define 60 and 63 feature pairs, respectively, on this example. In their video clips, the pumping out the horse feet from the rabbit looks too narrow at the beginning. In this paper, our warping method is foldover free and we can add more number of feature pairs up to 93. Therefore, unlike their results, we obtain a more elaborate morph in particular for legs. Finally, Table 1 shows some statistics in terms of vertices and triangles for the intermediate meshes. Using the proposed scheme, the intermediate meshes are much simpler than those

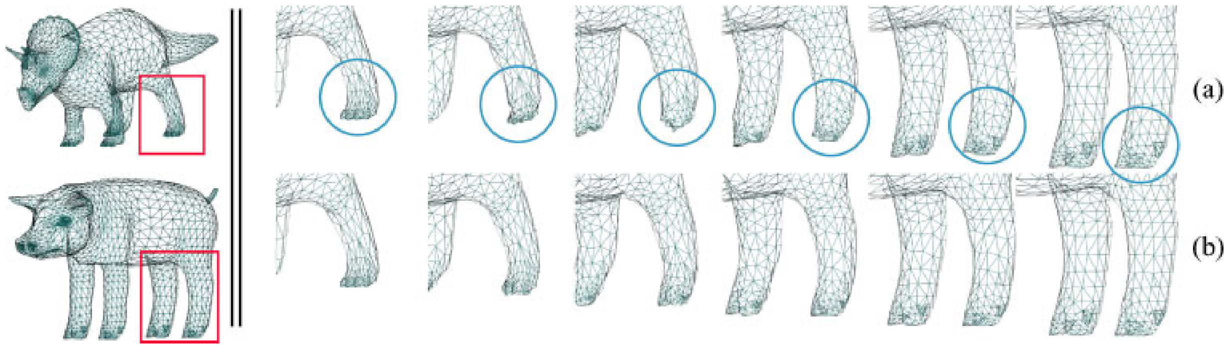


Figure 9. Comparison of the feature alignment between the warping without edge swaps (a) and the warping with edge swaps (b). The morph sequences (right) are enlarged from the red quadrangles (left).

	Source/target models (Number of vertices)/ (Number of faces)	Number of features vertices	Methods	Number of edge swap	Maximum error	Warping Time time (seconds)	Shape and connectivity transformation time (seconds)	Number of vertex ranges Number of face ranges in $M(t)$
Fig. 1	(5471, 10938)/	35	(a)	—	0.04534	589.922	35.32	(3899–5534)/
	(3755, 7506)		(b)	28	0.00544	58.219		(7794–11064)
Fig. 11	(2982, 5960)/	52	(a)	—	0.00099	366.047	20.24	(3191–3227)/
	(2995, 5986)		(b)	1	0.00099	45.25		(6378–6450)
Fig. 12	(10002, 20000)/	93	(a)	—	0.01970	658.703	101.32	(3134–10083)/
	(2982, 5960)		(b)	2	0.00397	70.87		(6264–20162)
Fig. 13	(44955, 89906)/	41	(a)	—	0.00097	381.625	1298.95	(44955–50002)/
	(50002, 100000)		(b)	0	0.00097	381.625		(89906–100000)

Table 1. The comparison of the warping method between Reference [4]'s method (a) and the proposed method (b) The 3rd column shows the number of feature vertices. The 5th column shows the number of edge swap for solving foldover. The 6th column shows the maximum distance of the correspondence feature vertices on the unit sphere. The 7th column shows the warping time for these two methods. The 8th column shows the connectivity transformation time to generate 50 frames and the 9th column shows a range of the number of vertices and triangles for the intermediate meshes.

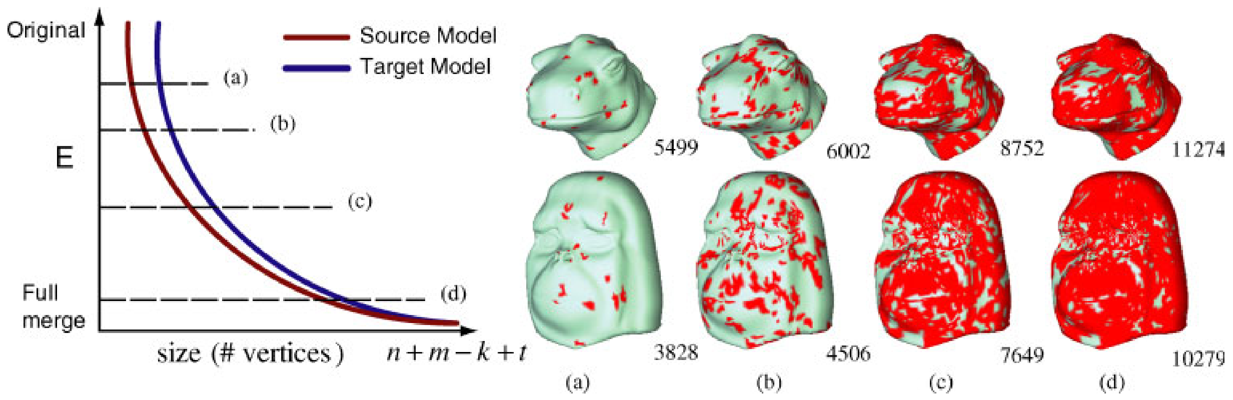


Figure 10. The number of vertices after semi-overlaying; left part: the number of vertices v.s. the parameter E on Equation (3); right part: the semi-overlaid vertices and edges are visualized in red color; full merging will create $n + m - k + t$ vertices, where t is the number of intersections after semi-overlaying.

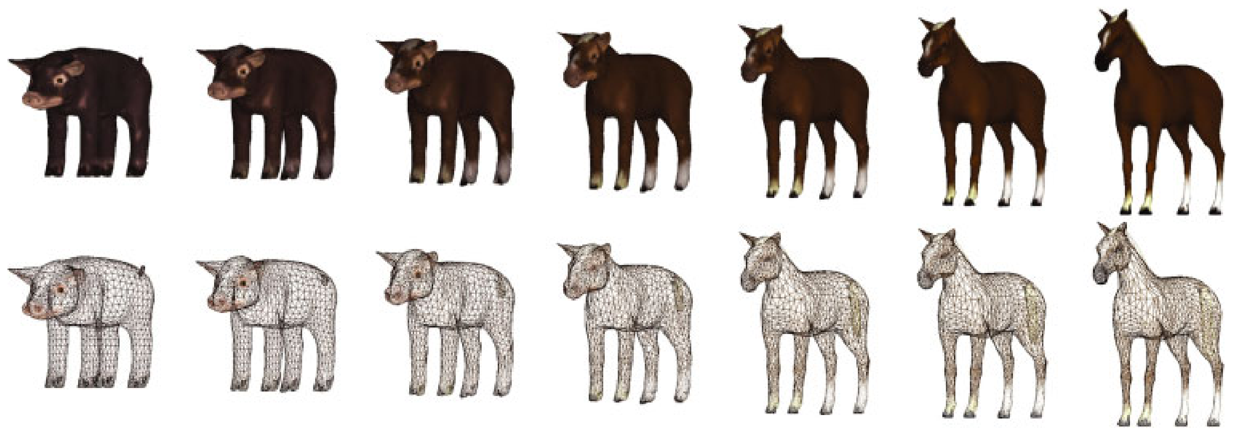


Figure 11. Morphs between the models of a pig and a horse.

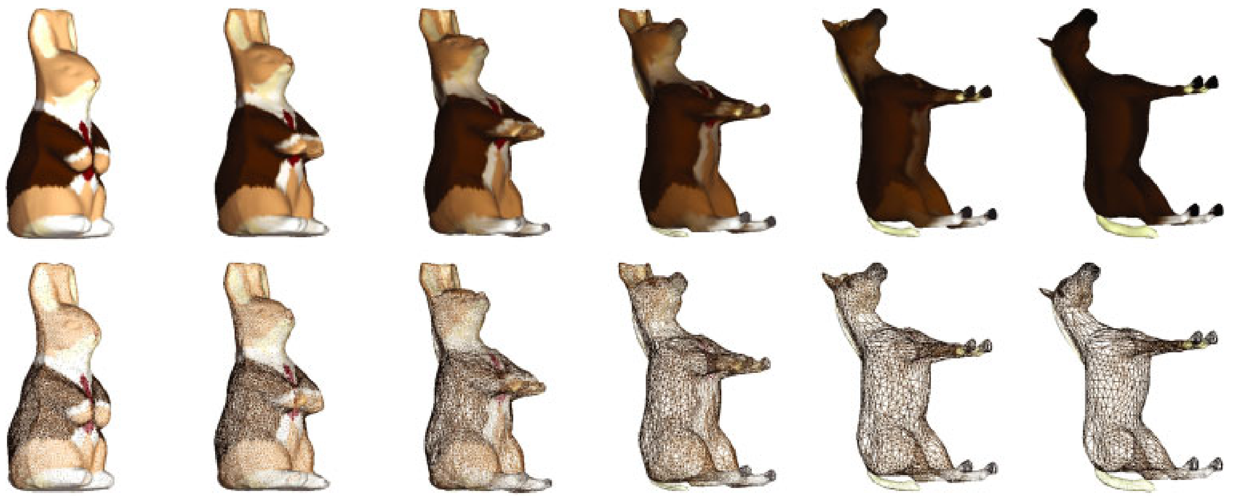


Figure 12. Morphs between the models of a rabbit and a horse.

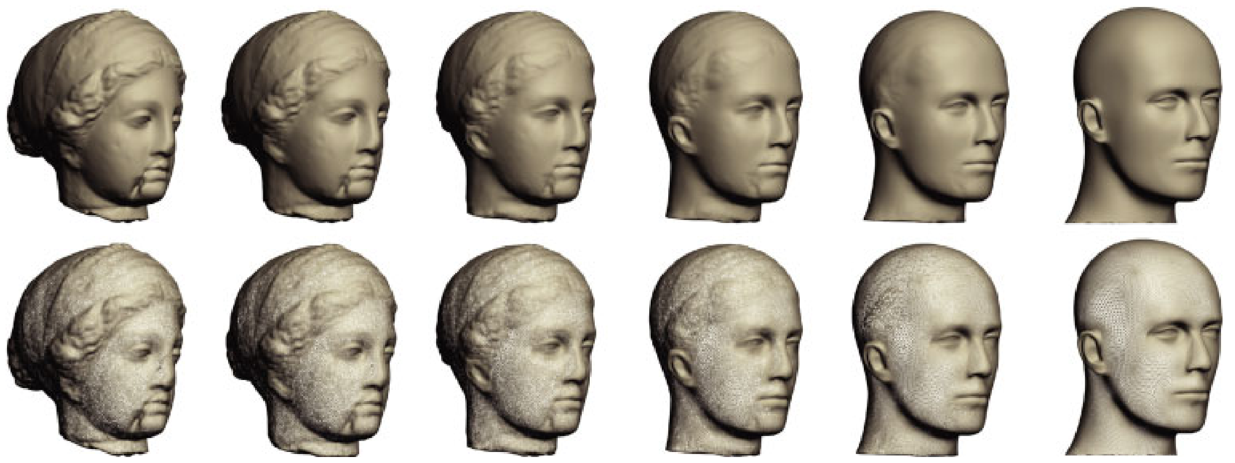


Figure 13. Morphs between the models of a venus's head to a shaven head.

generated by previous approaches. Either merging or re-meshing approach usually generates about several times of input models for intermediate meshes. For more examples, please see our accompany video.

Conclusion and Further Works

A novel integrated scheme for metamorphosis between two closed manifold genus-0 polyhedral models is presented. The proposed method adopts a foldover-free warping approach to align the spherical embeddings with several user-defined feature pairs. A new semi-overlaying scheme and a geomorph approach are used to reduce the popping effects caused by the connectivity transformation. In contrast to other previous works, not only the intermediate meshes generated by the proposed method have reasonable number of vertices and a smooth morphing is also obtained. Several future works can be done and are described as follows. To perform 3D morphs with texture is interesting and challenging future work. Simply interpolating texture attributes such as our examples can only work for simple examples. We are planning to design a better approach. Although the morphing sequences are smooth and the popping effects are reduced, the connectivity transformation is still discontinuous. A continuous connectivity transformation is required for a smooth morphing and for solving the popping problem. The optimization of the number of edge swap operations can be a very difficult but interesting problem. It relates to the problem in computational geometry of edge-swapping distance of triangulations. However, the continuous connectivity transformation must be solved first, since the discontinuous connectivity transformation is due to the edge swap operation. In addition, we are planning to exploit NPR technique[14] to visualize our 3D metamorphosis method in near future. For colour images in this paper, please see them at http://couger.csie.ncku.edu.tw/~vr/CAV85/color_figures.htm.

References

1. Gregory A, State A, Lin M, Manocha D, Livingston M. Interactive surface decomposition for polyhedra morphing. *The Visual Computer* 1999; **15**(9): 453–470.
2. Kanai T, Suzuki H, Kimura F. Metamorphosis of arbitrary triangular meshes. *IEEE Computer Graphics and Application* 2000; **20**(2): 62–75.
3. Zockler M, Stalling D, Hege H-C. Fast and intuitive generation of geometric shape transitions. *The Visual Computer* 2000; **16**(5): 241–253.
4. Alexa M. Merging polyhedral shapes with scattered features. *The Visual Computer* 2000; **16**(1): 26–37.
5. Shlafman S, Tal A, Katz SS. Metamorphosis of polyhedral surfaces using decomposition. In *EUROGRAPHICS Conference Proceedings*, 2002; pp. 219–228.
6. Lee T-Y, Huang P-H. Fast and intuitive metamorphosis of 3d polyhedral models using smcc mesh merging scheme. *IEEE Transactions on Visualization and Computer Graphics* 2003; **9**(1): 85–98.
7. Lin C-H, Lee T-Y. Metamorphosis of 3d polyhedral models using progressive connectivity transformations. *IEEE Transactions on Visualization and Computer Graphics* 2005; **11**(1): 2–12.
8. Fujimura K, Makarov M. Foldover-free image warping. *Graphics Models and Image Processing* 1998; **60**(2): 100–111.
9. Praun E, Sweldens W, Schröder P. Consistent mesh parameterizations. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, U.S.A., 2001; pp. 179–184.
10. Michikawa T, Kanai T, Fujita M, Chiyokura H. Multiresolution interpolation meshes. In *PG '01: Proceedings of the 9th Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, Washington, DC, U.S.A., 2001; p. 60.
11. Lee T-Y, Huang C-C. Dynamic and adaptive morphing of threedimensional mesh using control maps. *IEICE Transactions on Information and Systems* 2005; **e88-d**(3): 645–651.
12. Hoppe P. Progressive meshes. In *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press, New York, NY, U.S.A., 1996; pp. 99–108.
13. Lee AWF, Dobkin D, Sweldens W, Schröder P. Multiresolution mesh morphing. In *SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, U.S.A. 1999, pp. 343–360.
14. Chi M-T, Lee T-Y. Stylized and abstract painterly rendering system using a multi-scale segmented sphere hierarchy. *IEEE Transactions on Visualization and Computer Graphics*, 2006. Accepted.

Authors' biographies:

Chao-Hung Lin received his B.S. degree in Computer Science/Engineering from Fu-Jen University, Taipei, Taiwan, in 1997. He received his M.S. degree and his Ph.D. in Computer Engineering from National Cheng-Kung University, Tainan, Taiwan, in 1998 and 2004, respectively. He is now serving in the Taiwan army. His research interests include computer graphics and image processing.

Tong-Yee Lee was born in Tainan county, Taiwan, Republic of China, in 1966. He received his B.S. in Computer Engineering from Tatung Institute of Technology in Taipei, Taiwan, in 1988, his M.S. in Computer Engineering from National Taiwan University in 1990, and his Ph.D. in Computer Engineering from Washington State University, Pullman, in May 1995. Now, 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, non-photorealistic rendering, image-based rendering, visualization, virtual reality, surgical simulation, distributed and collaborative virtual environment. He leads a Computer Graphics Group/Visual System Lab at National Cheng-Kung University (http://couger.csie.ncku.edu.tw/~vr). He is a member of the IEEE.

Hung-Kuo Chu received his B.S. degree in Computer Science/Engineering from National Cheng-Kung University, Tainan, Taiwan, in 2003. Now, he is pursuing his Ph.D. at the Department of Computer Science and Information Engineering, National Cheng-Kung University. His research interest is computer graphics.

Chih-Yuan Yao was born in Tainan, Taiwan in 1980. He received his B.S. and M.S. degree in Computer Engineering from National Cheng-Kung University, Taiwan, in 2002 and 2003, respectively. He is currently working toward his Ph.D. at the Department of Computer Science and Information Engineering, National Cheng-Kung University. His research interests is computer graphics.