3D Geometric Metamorphosis based on Harmonic Map

Takashi KANAI Hiromasa SUZUKI Fumihiko KIMURA
Department of Precision Machinery Engineering
The University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo 113, Japan
{kanai,suzuki,kimura}@cim.pe.u-tokyo.ac.jp

Abstract

Recently, animations with deforming objects are frequently used in various Computer Graphics applications. Metamorphosis (or morphing) of three dimensional objects is one of techniques which realize a shape transformation between existing two or more objects. We present a new algorithm for 3D geometric metamorphosis between two objects based on Harmonic Map. Our algorithm is applicable for arbitrary polyhedra that are homeomorphic to the three dimensional sphere or the two dimensional disk. In our algorithm, each of two 3D objects is first embedded to the circular disk on the plane. This embedded model has the same graph structure as its 3D objects. By overlapping those two embedded models, we can establish correspondence between the two objects. Using this correspondence, intermediate objects between two objects are easily generated. The user only specifies a boundary loop on an object and a vertex on that boundary, and which control the interpolation.

1 Introduction

Image metamorphosis, or image morphing, is a popular technique for creating a smooth transition between two images by interpolating both color information and geometric shapes approximately [1, 9, 15]. These techniques have a number of advantages, but only in the case that underlying both objects in images don't move, or eye positions are not changed when two metamorphosis images are created by rendering.

Therefore, 3D metamorphosis (morphing), direct geometric interpolation of two 3D objects, has become one of active research themes in recent years.

Lerios et al. [10] extended Beier's feature-based image metamorphosis method [1] to volume representation. Two objects are translated into volumes, and each corresponding pair of volume values for volumes is interpolated according to "feature element pairs", which must be defined by the user. This method has also been presented for altering the topology of the surface mesh during transformation. However, the intermediate shape is represented as a volume, extraction to isosurface by such as Marching Cube method [11] is required for getting polygonal surfaces. Then, the extracted objects don't have smooth surfaces.

Kent et al. [8] offer an algorithm for the metamorphosis of polyhedral objects topologically equivalent to a sphere. First, two objects are mapped into a sphere, and are merged by clipping one sphere to another. Then a new shape which has connectivity information (vertices, edges, and faces) of two objects is created. However this method is applicable only for star-shaped, swept or revolutionary objects.

Parent [13] presents a recursive algorithm which automatically finds correspondences of surfaces between two objects of equivalent topology. This algorithm uses several *sheets* for covering the whole shape of the object. Those two objects must have the equal number of sheets. The boundary of sheets is composed of edges of the object. Objects of genus G are automatically subdivided into $2(G+1)$ sheets. Each sheet is recursively subdivided up to a face level. Vertex-to-vertex interpolations and thus deformations between two objects are completely established.

Delingette et al. [4] use a *simplex mesh*, and propose basic mesh operations to alter the shape topology. The interpolation was performed using a physically-based deformation approach, using a method derived from a data fitting process.

Decarlo et al. [3] describe another framework for the metamorphosis between two objects which have different topologies, for example, from a sphere to a torus. The user controls the transformation by specifying a sparse *control mesh* on each surface and by associating each face in one control mesh with that in the other. In this method, the user has to create control meshes by considering how to transform from one object to another. Therefore, the more complicated shape is, the harder users create control meshes.

In this paper, a new 3D metamorphosis algorithm based

on Harmonic Map is presented. The main contributions of our algorithm are as follows:

- Metamorphosis between two arbitrary objects, which have the same topology as a sphere or a disk, is available.
- User only has to specify a boundary in each object.
- Numerically stable and fast computation of correspondence are presented.

This paper is partitioned into the following sections. Section 2 describes an overview of our method for 3D metamorphosis based on Harmonic Map. In this section we overviews the algorithm, and explains mathematical background of harmonic map as an internal structure. Section 3 shows how to realize 3D metamorphosis. In this section we presents an algorithm for creating a new embeddings by merging two 2D embeddings generated from 3D objects, and shows a method for interpolation between two objects. Section 4 describes graphical user interface for an efficient implementation of our method, and discusses our results. We conclude with suggestions for future work and applications in Section 5.

2 Overview and Harmonic Map

2.1 Overview

In our research, we handle two 2-manifold objects which are topologically equivalent to a sphere, or to a disk. These objects are represented as a triangular mesh, or a polyhedron with triangular faces. We also assume that these two objects have the same topology. Hereafter we call them *the source object* $\mathcal{M}_1$ and *the target object* $\mathcal{M}_2$.

An approach to transforming from $\mathcal{M}_1$ to $\mathcal{M}_2$ is usually divided the problem into two steps. The first step is to establish a correspondence from each point of $\mathcal{M}_1$ to a point of $\mathcal{M}_2$. Once this correspondence is established, in the next step a series of intermediate objects are created by interpolating corresponding points from their original position to the target. These steps are called *correspondence problem* and *interpolation problem* respectively. The algorithm we suggest is referred to mainly the former issue.

Our algorithm for finding correspondence begins to develop $\mathcal{M}_1$ and $\mathcal{M}_2$ to a two dimensional unit disk D^2 and create embeddings by Harmonic Map. These 2D embeddings, called $\mathcal{H}_1$ and $\mathcal{H}_2$, have the same connectivity[1] as $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively, i.e. adjacent relations of vertices, edges, and faces are preserved. Figure 1. shows an

[1] we use a word *connectivity* to denote a vertex/edge/face graph of a polygon.

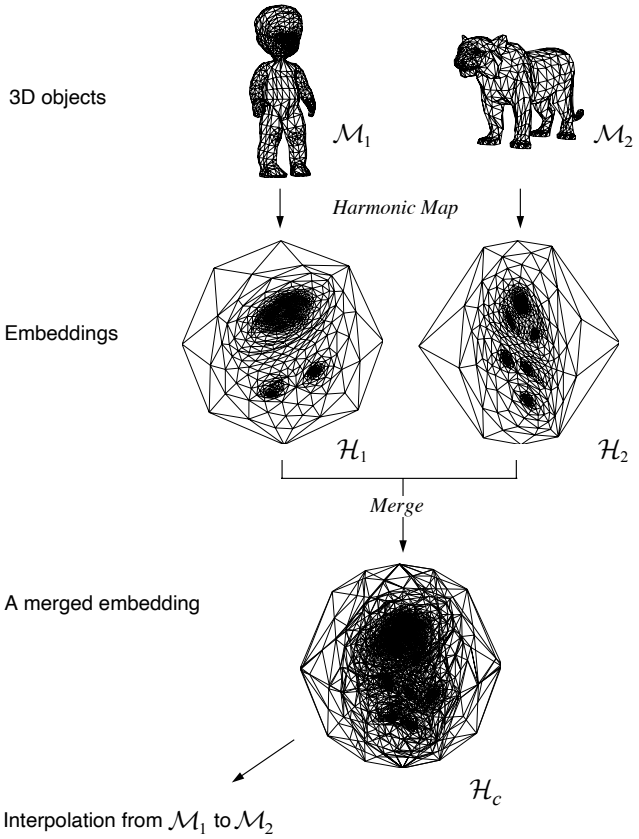


Figure 1. A overview of our method for establishing 3D geometric metamorphosis.

overview for establishing correspondence between two objects. For given two objects $\mathcal{M}_1$ and $\mathcal{M}_2$, 2D objects, respectively $\mathcal{H}_1$ and $\mathcal{H}_2$, are created by mapping into a unit disk using Harmonic Map.

Next, these two embeddings are merged and a new object $\mathcal{H}_c$ is created which has connectivity inherited from both two objects and defines one-to-one correspondence between each position of $\mathcal{M}_1$ to that of $\mathcal{M}_2$. Since $\mathcal{H}_c$ has not only the connectivity of $\mathcal{H}_1(\mathcal{M}_1)$, but also that of $\mathcal{H}_2(\mathcal{M}_2)$. The correspondence between $\mathcal{H}_1$ and $\mathcal{H}_2$ (and thus between $\mathcal{M}_1$ and $\mathcal{M}_2$) is established.

In the same way as most of previously addressed works, our method for solving the correspondence problem will be based on *surgery of connectivity*. That is, to create a new object which has a blended connectivity of both two objects (or those projects) , it is nessessary to subdivide those original faces, and to create new vertices/edges/faces, then mesh connectivities are changed. In our notation, a new object corresponds to $\mathcal{H}_c$. These operations will be implemented geometically, that causes to some numerical unstabilities. Our method can avoid such unstabilities when creating cor-

respondences as described later (Section.3.1).

By using $\mathcal{H}_c$, continuous intermediate objects are created by interpolation. To establish an interpolation between $\mathcal{M}_1$ and $\mathcal{M}_2$, we use a simple linear interpolation technique.

Note that these two embeddings are used for internal data structure, whose related calculation is implicitly processed. The user specifies only correspondence information on the boundary of Harmonic Map as described later.

2.2 Harmonic Map

In this subsection, we discuss about the mathematical background and a method for mapping from an object $\mathcal{M}$ in $\mathbf{R}^3$ to a unit circle $\mathcal{H}$ in $\mathbf{R}^2$.

Harmonic Map, $h : \mathcal{M} \rightarrow \mathcal{H}$ is one of the mapping that realizes embedding from a topological disk to a planar graph, which has a property to minimize metric dispersion [2] (see [2] about other mathematical terminologies).

To construct this embedding, we adopted a method proposed by Eck et al. [5]. Strictly speaking, Eck's mapping method is a piecewise linear approximation method for realizing embedding from $\mathcal{M}$ to $\mathcal{H}$. A similar method is proposed by Marillot et al. [12]. The reason why we adopt Eck's embedding method is that stable and fast embedding can be established.

Eck's embedding method is established in the following steps: First, n vertices, which make up a boundary segment of H, are positioned on the disk (the boundary of D^2) in $\mathbf{R}^2$ whose center point is the origin, so that angles between two boundary vertices are proportional to edge lengths connected to those vertices. The positions of the rest of the vertices are calculated so that the total energy E_{harm} is minimized. E_{harm} can be represented as a sum of the energy of a configuration of springs with one spring placed along each edge of $\mathcal{M}$:

$$E_{harm}(\boldsymbol{v}) = \frac{1}{2} \sum_{\{i,j\} \in Edges(\mathcal{H})} \kappa_{i,j} \{||\boldsymbol{v}_i - \boldsymbol{v}_j||\}^2, \quad (1)$$

where i, j denote vertices, $\boldsymbol{v}_i, \boldsymbol{v}_j$ denote their positions in $\mathcal{H}$ and $\boldsymbol{v}$ is the set of $\boldsymbol{v}_i$. $Edges(\mathcal{H})$ denotes the set of edges in $\mathcal{H}$ and $\{i, j\}$ is an edge connecting vertices i and j. $\kappa_{i,j}$ denotes a spring constant for edge $\{i, j\}$, and is calculated as follows. For each edge $\{i, j\}$, let $l_{i,j}$ denote its length as measured in $\mathcal{M}$. For each face $\{i, j, k\}$, let $A_{i,j,k}$ denote its area, again as measured in $\mathcal{M}$. For each interior edge $\{i, j\}$ incident to two faces $\{i, j, k_1\}, \{i, j, k_2\}$:

$$\kappa_{i,j} = \frac{{l_{i,k_1}}^2 + {l_{j,k_1}}^2 - {l_{i,j}}^2}{A_{i,j,k_1}} + \frac{{l_{i,k_2}}^2 + {l_{j,k_2}}^2 - {l_{i,j}}^2}{A_{i,j,k_2}}. \quad (2)$$

[2] Metric dispersion means a measure of distortion, i.e. the extent to which a map stretches regions of small diameter in D^2.

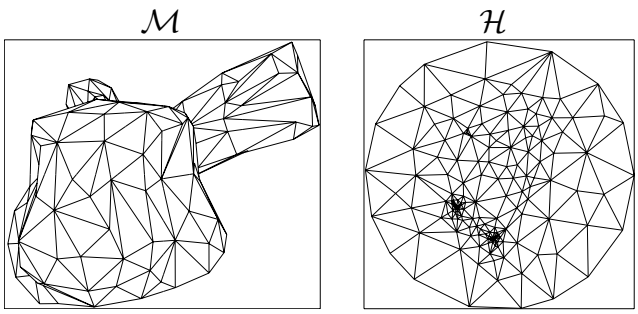


Figure 2. Harmonic Map: (a) a bunny model. (b) Harmonic Map.

A unique solution that minimizes E_{harm} can be found by solving a sparse linear system $\nabla E_{harm} = \mathbf{0}$, where ∇E_{harm} is the gradient of E_{harm} over $\boldsymbol{v}$, because E_{harm} is a positive quadratic function over vertex positions in H. To solve the gradient of E_{harm}, x, y components of $\boldsymbol{v}$ are listed in order and a $2N$ dimensional vector $\mathbf{V}$ is defined as follows:

$$\mathbf{V} \equiv (v_{1x}, v_{1y}, v_{2x}, v_{2y}, ..., v_{Nx}, v_{Ny}), \quad (3)$$

where N denotes the number of vertices. E_{harm} is quadratic form over every component in $\mathbf{V}$, it can be represented as a form $E_{harm} = \mathbf{V}^T\mathbf{H}\mathbf{V}$. Then the gradient of E_{harm} can be expressed as $\nabla E_{harm} = \partial E_{harm}/\partial \mathbf{V}$.

Moreover vertices on the boundary are fixed. To solve a linear system, we first divide a variable vector $\mathbf{V}$ into two parts, a free part $\mathbf{V}^\alpha$ and a fixed part $\mathbf{V}^\beta$, and the constant matrix $\mathbf{H}$ is also divided accordingly. Thus, an energy function E_{harm} is rewritten as follows:

$$E_{harm} = \begin{bmatrix} \mathbf{V}^{\alpha T} & \mathbf{V}^{\beta T} \end{bmatrix} \begin{bmatrix} \mathbf{H}^{\alpha\alpha} & \mathbf{H}^{\alpha\beta} \\ \mathbf{H}^{\beta\alpha} & \mathbf{H}^{\beta\beta} \end{bmatrix} \begin{bmatrix} \mathbf{V}^{\alpha} \\ \mathbf{V}^{\beta} \end{bmatrix}. \quad (4)$$

As this energy function is constant for fixed parts, we only solve a linear equation from the gradient ∇E_{harm} over free parts $\mathbf{V}^\alpha$ to minimize E_{harm}:

$$\nabla E_{harm} = \frac{\partial E_{harm}}{\partial \mathbf{V}^\alpha} = 2\mathbf{H}^{\alpha\alpha}\mathbf{V}^\alpha + 2\mathbf{H}^{\alpha\beta}\mathbf{V}^\beta = \mathbf{0}. \quad (5)$$

Figure 2. shows an example of this embedding method. An original 3D "bunny" model (Figure 2. (a)) has a closed boundary segment along its neck. 17 vertices consisting of a boundary segment are positioned on a unit circle in $\mathbf{R}^2$. Faces of the original object are developed into the disk without self-intersection, no matter whether an object has convex region or concave region. For example, the result (Figure 2. (b)) shows that bunny's ears which include concave regions are successfully developed into the disk.

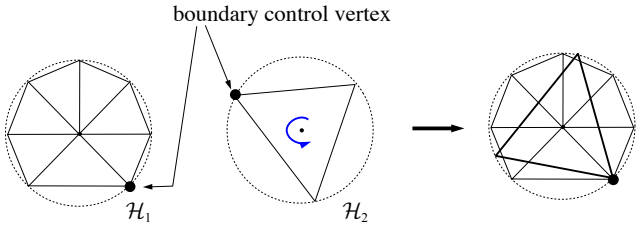


Figure 3. Registration of Harmonic Maps.

3 Metamorphosis Based on Harmonic Map

3.1 Find Surface Correspondences

In the preceding section, 2D embeddings $\mathcal{H}_1, \mathcal{H}_2$ from 3D objects $\mathcal{M}_1, \mathcal{M}_2$ were generated respectively. Each of two embeddings has the same connectivity as its 3D object. This subsection describes how to create new embedding $\mathcal{H}_c$ by merging $\mathcal{H}_1$ and $\mathcal{H}_2$. $\mathcal{H}_c$ has a combined connectivity of those two embeddings $\mathcal{H}_1, \mathcal{H}_2$, and as a result, $\mathcal{M}_1, \mathcal{M}_2$. Moreover, correspondences from arbitrary points of to the source object to those to the target object are also established in $\mathcal{H}_c$.

Kent et al. [8] has proposed a similar method. The difference between our method to their method is that we take into account *coincidence issues*. Their method assumes that no embedded vertices of the two objects are coincident and that no embedded vertex of the source object lies on a projected edge of the target object. Our method can handle these coincident issues. This is important for avoiding numerical unstability which frequently occurs from these coincidence cases especially when objects have a number of faces.

The construction of $\mathcal{H}_c$ consists of four steps.

The first step rotates $\mathcal{H}_1$ (or $\mathcal{H}_2$) around the center of D^2 so that an embedded vertex of $\mathcal{H}_1$ selected by a user (hereafter we call it *boundary control vertex*) is coincident with that in $\mathcal{H}_2$ (Figure 3).

The second step calculates corresponding 3D positions in $\mathcal{M}_1$ and $\mathcal{M}_2$ of each vertex at $\mathcal{H}_1$ and $\mathcal{H}_2$. First, to calculate 3D position at $\mathcal{M}_2$ for each vertex $\boldsymbol{v}_m^1$ in $\mathcal{H}_1$, we search a face at $\mathcal{H}_2$ that $\boldsymbol{v}_m^1$ is included. Let $\mathbf{v}_m^1, \mathbf{v}_m^2$ be corresponding 3D positions of $\boldsymbol{v}_m^1$ at $\mathcal{M}_1, \mathcal{M}_2$ respectively. When $\boldsymbol{v}_m^1$ is included in a face $f = \{i, j, k\}$ of $\mathcal{H}_2$, a barycentric coordinate (u, v, w) of $\boldsymbol{v}_m^1$ for f is calculated as shown in Figure 4. Using this coordinate, a corresponding 3D position $\mathbf{v}_m^2$ of a vertex $\boldsymbol{v}_m^1$ over $\mathcal{M}_2$ is calculated as follows:

$$\mathbf{v}_m^2 = u\mathbf{v}_i^2 + v\mathbf{v}_j^2 + w\mathbf{v}_k^2, \quad u + v + w = 1. \tag{6}$$

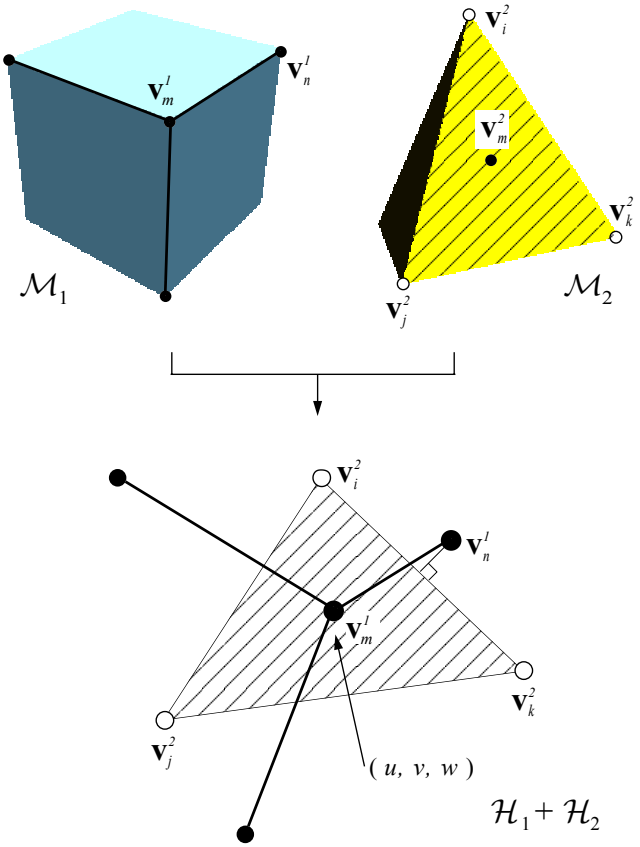


Figure 4. Mapping vertices in $\mathcal{H}_1$ to $\mathcal{H}_2$.

In a similar way, for each vertex in $\mathcal{H}_2$ its corresponding 3D position at $\mathcal{M}_1$ is calculated. To speed up the searching of a face including a vertex, we use a spatial partitioning data structure such as the quad-tree data structure [6], then the search is performed in $O(n \log n)$ time.

Especially vertices near the boundary are not included any faces can exist (such as $\boldsymbol{v}_j^n$ in Figure 4). For these vertices, we calculate a 3D position by using barycentric coordinates of the face which has the nearest distance from its edges.

One of the reason why a calculation failure by numerical errors occurs is a coincidence issue. There are cases in which a vertex in $\mathcal{H}_1$ and a vertex in $\mathcal{H}_2$ are as being located nearly at the same position, or in which a vertex of one embedding is as being nearly on an edge of the other embedding (Figure 5) .

To solve this issue, we first consider two vertices of $\mathcal{H}_1$ and $\mathcal{H}_2$ within a threshold distance to be coincident. After the search of all coincident vertices, we re-calculate $\mathcal{H}_1, \mathcal{H}_2$ again with coincident vertices fixed. Each position of coincident vertices is set to an average of their incident vertices. When $\mathcal{H}_1$ and $\mathcal{H}_2$ are re-calculated, we can again use the energy function (Equation (4)). In Equation (4), the number

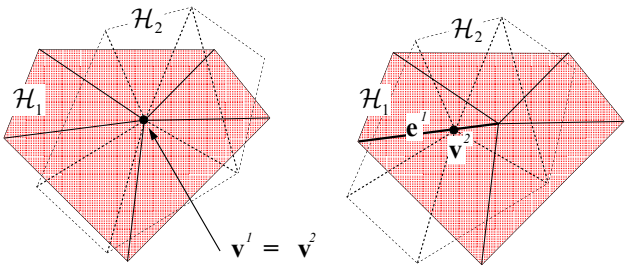


Figure 5. Coincidence issues: (a) two vertices in $\mathcal{H}_1$ and $\mathcal{H}_2$ have same position. (b) a vertex in $\mathcal{H}_1$ is on a edge in $\mathcal{H}_2$ (an opposite case can be done).

of rows of $\mathbf{V}^\beta$ is increased. These operations are repeated and terminated if no coincident vertices are generated. The process dealing with the case in which a vertex of one embedding is on an edge of the other embedding is also realized by similar substeps. In this case, the boundary vertices of an associated edge are fixed. Moreover, the relative position of a vertex to the edge is preserved as a parametric form, and a new fixed position is calculated using this parametric value.

The third step investigates edge-to-edge intersection between edges of $\mathcal{H}_1$ and those of $\mathcal{H}_2$. If an intersection is found, both edges are divided each other at this intersection point. This process is also established in $O(n \log n)$ time by using the spatial partitioning data structure.

$\mathcal{H}_c$ generated by the above step is composed of vertices and edges, but has no triangular faces. The last step generates triangular faces in $\mathcal{H}_c$ (Figure 6). First at every vertex, edges incident to the vertex are sorted in the counter-clockwise order. A new face $f^c = \{i, j, k\}$ is generated by using two continuous edges $e_i = \{i, k\}$ and $e_j = \{j, k\}$. If there is no edge between j and k, a new edge $\{j, k\}$ is created simultaneously and is added to the edge-list of vertices j and k. This operation continues until all of edges in $\mathcal{H}_c$ have a face on both sides. This greedy-like operation executes in $O(n)$ time (A simple proof that single time search of vertices which have a list of neighboring edges in clockwise or counter-clockwise order makes covering all of faces in $\mathcal{H}_c$ is described in [2]).

For objects which are topologically equivalent to a sphere, two embeddings are generated over each object. In this case, we can apply an above approach to each of two pairs of embeddings $\mathcal{H}_1$ and $\mathcal{H}_2$ respectively, then two $\mathcal{H}_c$ are generated.

3.2 Interpolation

An interpolation between the source object $\mathcal{M}_1$ to the target object $\mathcal{M}_2$ is realized by applying a simple linear

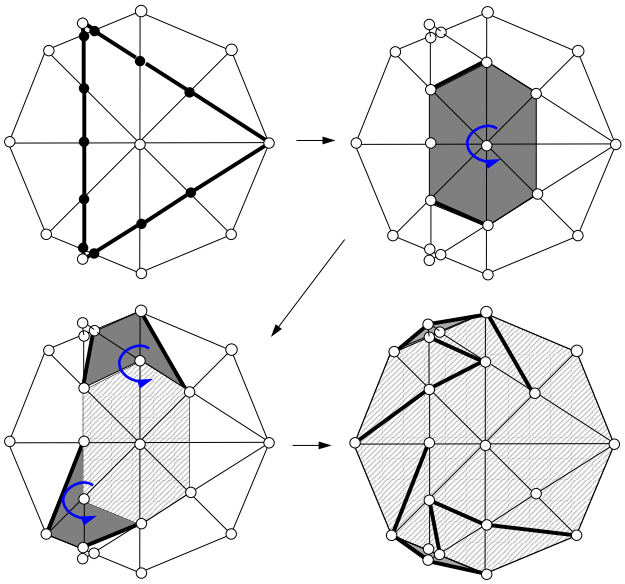


Figure 6. A method for creating a new object $\mathcal{H}_c$ from $\mathcal{H}_1$ and $\mathcal{H}_2$

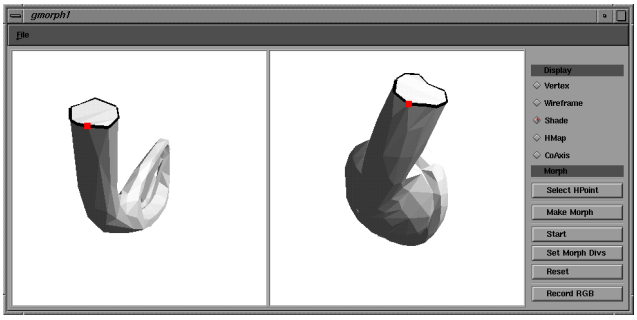


Figure 7. Graphical User Interface

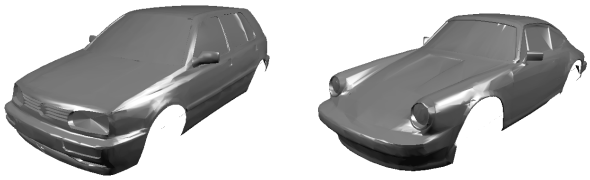
method. Let $\mathbf{v}^c_m (m : 1 \ldots N^c)$ be a 3D position of a vertex $\boldsymbol{v}^c_m$ in $\mathcal{H}_c$, where N^c denotes the number of vertices in $\mathcal{H}_c$. For any time $t \in [0, 1]$, $\mathbf{v}^c_m$ are calculated using $\mathbf{v}^1_m$ and $\mathbf{v}^2_m$ as follows:

$$\mathbf{v}^c_m = (1 - t)\mathbf{v}^1_m + t\mathbf{v}^2_m. \tag{7}$$

4 Results

4.1 User Interface

Based on the methods described in the preceding sections, we developed a prototype system on a Sillicon Graphics workstation. Figure 7. shows the layout of the user-interface. This user-interface has two windows. The left window displays the source object, and the right displays



(a) (b)

Figure 8. (a) "Volkswagen Golf" model. (b) "Porsche 911" model.

the target object. Users can interactively operate transformations, rotations, scalings of objects by a mouse. Of course, these objects are located on the same coordinate system. Moreover, users can check a transition from the source object to the target object by this system.

To establish the 3D metamorphosis method described above, the following operations must be done by users:

Boundary loop specification To calculate embedding described as a subsection 2.2, users must specify a boundary loop on both objects. Each boundary loop is composed of edges of the objects, and must contain more than three vertices. Directions of two loops must be same directions.

A boundary control vertex specification A vertex on each boundary loop must be specified. Because these two vertices define the relative position of the two boundary loops. These vertices are used in the subsection 3.1.

In both original objects in Figure 7., bold lines denote boundary loops and filled rectangle marks denote boundary control vertices.

4.2 Examples

We first demonstrate two metamorphosis examples between a golf club model (#vertices: 149, #faces: 287) and an ear-phone model (#vertices: 245, #faces: 477). Both two models are topologically equivalent to the sphere, and are generated by Hoppe's surface reconstruction and simplification method [7] from range images. Each of both objects has a unique center point (an average of center of balances of faces) which is located near to the origin of the world coordinate system.

Figure 9. (a) shows a sequence of metamorphosis using user-specified boundary loops and boundary control vertices illustrated in Figure 7. Figure 9. (b) shows another sequence of metamorphosis. In this metamorphosis, only a boundary control vertex of ear-phone model is changed (to be that opposite side of the boundary loop). In both examples, each of $\mathcal{H}_c$ was obtained within a second (MIPS R4400 250MHz). The calculation is so fast that we can interactively create more fine results by some changes of transformations, rotations or scaling of only one object of the two.

The other metamorphosis is made between a "Volkswagen Golf" model (Figure 8. (a), #vertices: 2978, #faces: 5798) and "Porsche 911" model (Figure 8. (b),#vertices: 1955, #faces: 3765). Both two models are topologically equivalent to a disk. Figure 10. shows result of metamorphosis. $\mathcal{H}_c$ (#vertices: 16909, #faces: 32101) was obtained in about 46.0 seconds.

One problem that arises during the interpolation is that an intermediate object can have self-intersections. It seems quite difficult to avoid such self-intersections. Because, for example, we consider metamorphosis between certain two objects without such intersection. However if one of these objects is transformed, there is no guarantee that intermediate shapes during the interpolation between two objects that are at new locations are not self-intersected. That is, locations of the object have an effect on the self-intersections of intermediate objects. Thus whether these self-intersections are permitted or not depends on the application.

5 Conclusion and Future Work

We described a method for three dimensional geometric metamorphosis based on Harmonic Map for any two objects which are topologically equivalent to a sphere or to a disk. Interpolation can be controlled by specifing only a boundary loop and a vertex on the boundary. This method is executed so fast that finer control of the interpolation can also be achieved.

We think that the following three extensions are effective for interactive control of the interpolation:

- The number of boundary control vertices is limited. For more complicated objects, an extension for some local shape control have to be needed. For example, in metamorphosis between one person to the other person, we may wish local controls so that the right arm of one person is transformed to that of the other person.
- All of vertices at a correspondence object are transformed simultaneously from $t = 0$ (object A) to $t = 1$ (object B). It is desired that only a part of a object is transformed, while the rest of it is not changed.
- The number of faces of $\mathcal{H}_c$ is greatly increased. Therefore, it is quite slow to display animation during metamorphosis. To establish real-time animation of metamorphosis, a method for decreasing number of faces without losing surface features of the original objects are needed. Many approaches such as [7] address surface simplification.

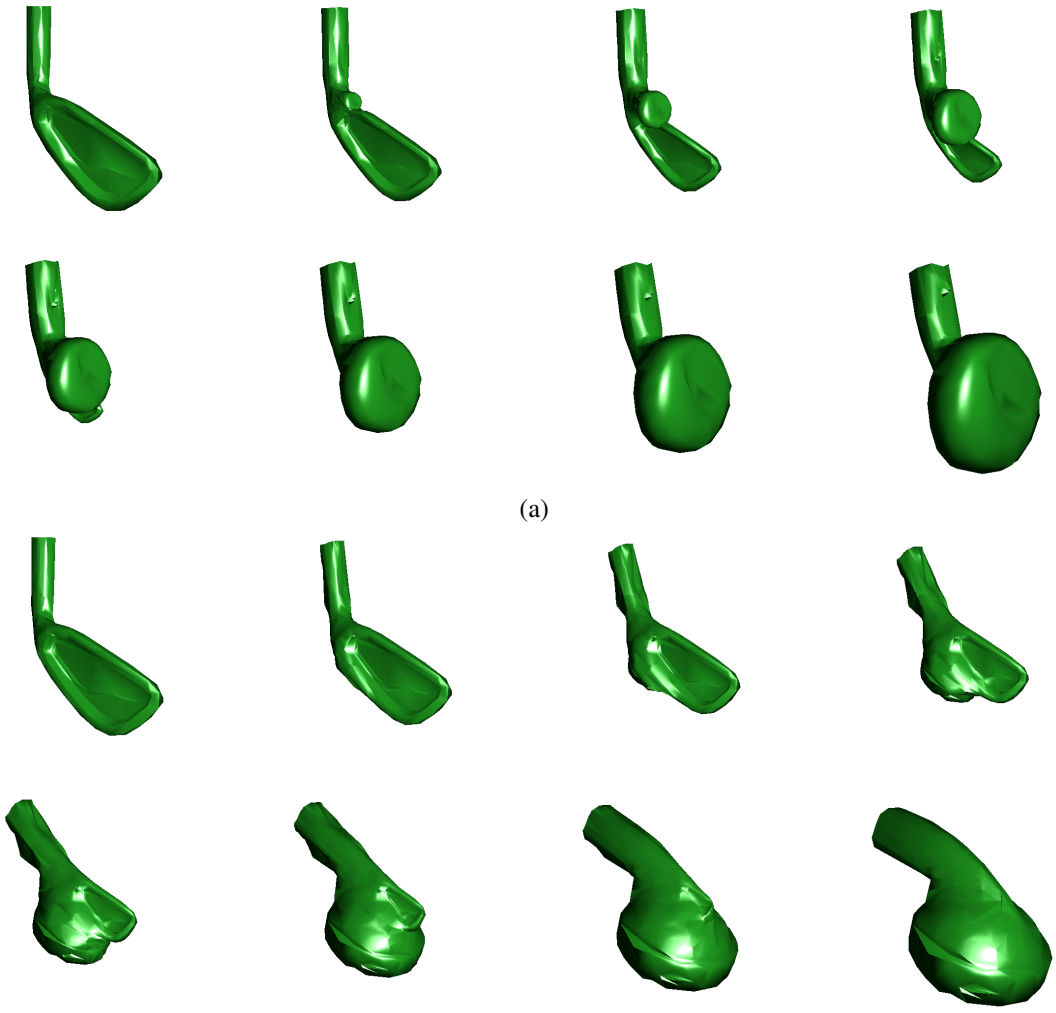


Figure 9. (a) "golf club" to "phone" metamorphosis. (b) "golf club" to "phone" metamorphosis with a user-specified vertex on a loop of ear-phone model is changed.

- Our method can be referred only the correspondence problem. We use a linear method for the interpolation problem, and we know our method can cause some problems (self intersection, shape distortion, and so on), and we know that some approaches address this problem ([14] for 2D polygon case, [16] for 3D polyhedral polygon case). Unfortunatelly, these approaches are useful only for the case that two models are similar shape. We are going to find a interpolation algorithm that can work well with our correspondence algorithm.

6 Acknowledgements

We thank all the referees for making us a aware of references and giving us a lot of useful comments.

References

[1] T. Beier and S. Neely. Feature-based image metamorphosis. In *Proceedings of ACM SIGGRAPH'92*, pages 35–42. ACM Press, 1992.

[2] G. Chartrand and L. Lesniak. *Graphs & Digraphs*. The Wadsworth & Brooks/Cole, 2nd edition, 1986.

[3] D. DeCarlo and J. Gallier. Topological evolution of surfaces. In *Graphics Interface'96*, pages 194–203. CHCCS, 1996.

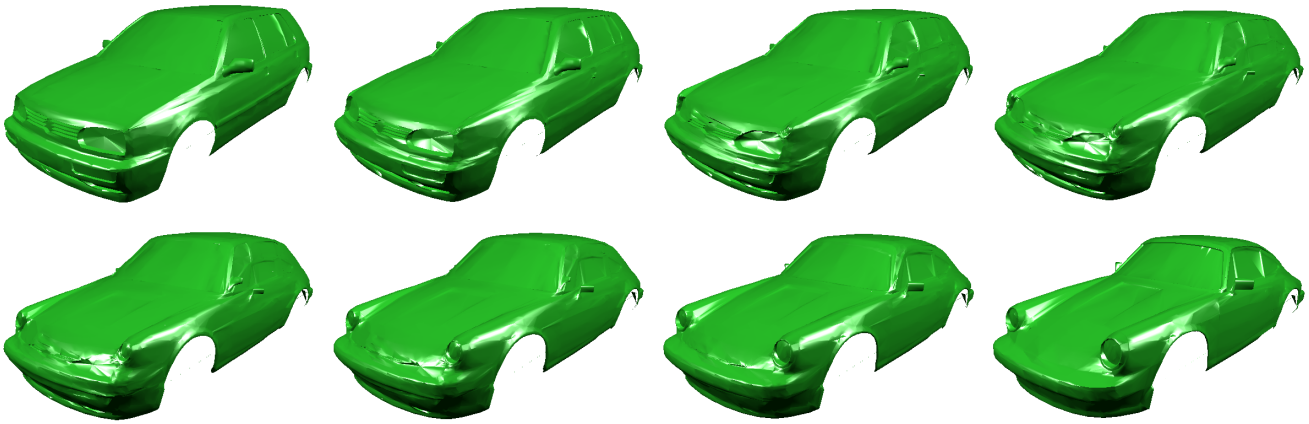


Figure 10. “volkswagen” to “porsche” metamorphosis.

[4] H. Delingette, Y. Watanabe, and Y. Suenaga. Simplex based animation. In N. M. Thalmann and D. Thalmann, editors, *Models and Techniques in Computer Animation*. Springer-Verlag, 1993.

[5] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of ACM SIGGRAPH'95*, pages 173–182. ACM Press, 1995.

[6] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics - Principles and Practice -*. Addison-Wesley Publishing Company, 2nd edition, 1990.

[7] H. Hoppe. *Surface reconstruction from unorganized points*. PhD thesis, Department of Computer Science and Engineering, University of Washington, 1994.

[8] J. R. Kent, W. E. Carlson, and R. E. Parent. Shape transformation for polyhedral objects. In *Proceedings of ACM SIGGRAPH'92*, pages 47–54. ACM Press, 1992.

[9] S.-Y. Lee, K.-Y. Chwa, and S. Y. Shin. Image metamorphosis using snakes and free-form deformations. In *Proceedings of ACM SIGGRAPH'95*, pages 439–448. ACM Press, 1995.

[10] A. Lerios, C. D. Garfinkle, and M. Levoy. Feature-based volume metamorphosis. In *Proceedings of ACM SIGGRAPH'95*, pages 449–456. ACM Press, 1995.

[11] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of ACM SIGGRAPH'87*, pages 163–169. ACM Press, 1987.

[12] J. Maillot, H. Yahia, and A. Verroust. Interactive texture mapping. In *Proceedings of ACM SIGGRAPH'93*, pages 27–34. ACM Press, 1993.

[13] R. E. Parent. Shape transformation by boundary representation interpolation: a recursive approach to establishing face correspondences. In *Journals of Visualization and Computer Animation*, volume 3, pages 219–239, 1992.

[14] T. W. Sederberg, P. Gao, G. Wang, and H. Mu. 2-D shape blending: An instrinsic solution to the vertex path problem. In *Proceedings of ACM SIGGRAPH'93*, pages 15–18. ACM Press, 1993.

[15] T. W. Sederberg and E. Greenwood. A physically based approach to 2D shape blending. In *Proceedings of ACM SIGGRAPH'92*, pages 25–34. ACM Press, 1992.

[16] Y. M. Sun, W. Wang, and F. Y. L. Chin. Interpolating polyhedral models using intrinsic shape parameters. In *Pacific Graphics '95*, Aug. 1995.