THE JOURNAL OF VISUALIZATION AND COMPUTER ANIMATION, VOL. **8**: 81–96 (1997)

Interpolating Polyhedral Models Using Intrinsic Shape Parameters

YUE MAN SUN, WENPING WANG AND FRANCIS Y. L. CHIN
Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong
(email: ymsun@cs.hku.hk, wenping@cs.hku.hk, and chin@cs.hku.hk)

SUMMARY

Metamorphosis, or morphing, is the gradual transformation of one shape into another. It generally consists of two subproblems: the correspondence problem and the interpolation problem. This paper presents a solution to the interpolation problem of transforming one polyhedral model into another. It is an extension of the intrinsic shape interpolation scheme (T. W. Sederberg, P. Gao, G. Wang and H. Mu, '2-D shape blending: an intrinsic solution to the vertex path problem, *SIGGRAPH '93*, pp. 15–18.) for 2D polygons. Rather than considering a polyhedron as a set of independent points or faces, our solution treats a polyhedron as a graph representing the interrelations between faces. Intrinsic shape parameters, such as dihedral angles and edge lengths that interrelate the vertices and faces in the two graphs, are used for interpolation. This approach produces more satisfactory results than the linear or cubic curve paths would, and is translation and rotation invariant. © **1997 by John Wiley & Sons, Ltd.**

KEY WORDS: metamorphosis; computer animation; interpolation; shape transformation; object representation; planar graph

1. INTRODUCTION

Metamorphosis, or morphing, is the gradual transformation of one shape into another. Given two polyhedral models, a metamorphosis algorithm determines the in-between polyhedra to animate the process of metamorphosis. It is best illustrated in Figure 1, in which the leftmost object is morphed into the rightmost. The upper and lower sequences show the morphing in two possible manners.

This paper addresses the metamorphosis of 3-D polyhedral models. Metamorphosis usually consists of two subproblems:[2] the correspondence problem and the interpolation problem. A correspondence specifies which face, edge or vertex of one model is mapped to which face, edge, vertex of the other model. A correspondence algorithm determines how the face/edge/vertex is transformed during the animation according to the particular correspondence.

In the existing approaches to metamorphosis[2–9] the correspondence problem is the main concern and the interpolation problem is given less emphasis. Linear or Hermite cubic vertex paths are usually used. Poor visual effects, such as unnatural shrinkage and flipping inside out of the objects, often occur in these interpolation schemes. A

CCC 1049–8907/97/020081–16 $17.50
© 1997 by John Wiley & Sons, Ltd.
Received January 1996
Revised July 1996

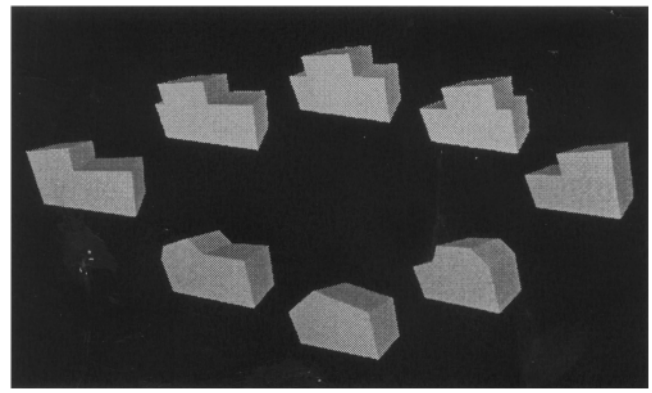

Figure 1. Morphing of two L-shapes

solution to the interpolation problem of 2-D polygons based on the intrinsic shape definition of polygons has recently been proposed.[2] It is particularly useful in animating figures of similar shapes and successfully eliminates various unnaturally formings (shrinking and kinking effects) that result from the linear vertex interpolation. Intrinsic parameters are invariant under translation and rotation, locally defined, and are therefore ideal candidates for shape interpolation. Unfortunately, the intrinsic shape interpolation method[2] works only for 2-D polygons. Intrinsic definition for polygons is simple: a polygon can be defined by a series of the intrinsic parameters, edge lengths and interior angles, enumerated in either clockwise or counterclockwise direction. However, no intrinsic definition is available for polyhedra. There are 3D intrinsic parameters but they are highly interdependent. It is not known how to define a polyhedron by enumerating a series of intrinsic parameters. Therefore, it is difficult to use these parameters in 3-D metamorphosis. Thus, morphing 3-D polyhedral models using intrinsic shape parameters becomes the natural, and challenging, next step in research.

This paper presents a solution to the interpolation problem of transforming one polyhedral model into another using intrinsic shape parameters. We will restrict our discussion to genus zero polyhedra, i.e. polyhedra with no handles. The proposed method is inspired by the intrinsic shape interpolation[10] for 2-D polygons. Instead of being treated as a collection of independent points or faces, an object is considered to be defined by a planar graph representing the interrelations between vertices; its dual graph is also used to represent the interrelations between faces. The method is not restricted to one-to-one correspondence, and experience shows that it is feature preserving and produces visually natural results.

After a brief review of previous work in Section 2, our method is outlined in Section 3. Section 4 presents the main steps of the algorithm: the interpolations of the face adjacency and the vertex adjacency graphs. Section 5 provides a complete description of the algorithm. Some properties and experimental results of the algorithm are discussed in Section 6. The paper concludes in Section 7 with a discussion of the improvements in our algorithm over previous work and directions for future research.

2. PREVIOUS WORK

Most existing techniques for solving the interpolation problem can be classified into one of the following categories: (a) linear interpolation; (b) independent interpolation of each vertex pair using a curved path; (c) intrinsic shape interpolation.

(a) Linear vertex path

In linear interpolation, each pair of corresponding points is linearly interpolated independently. Usually, only the corresponding vertices are interpolated. It is the most widely used technique both in research and commercial softwares. The super-object technique[2] and the method of minimizing the distances or corresponding face centroids and vertex pairs[4] both use the linear vertex path. It is simple, computationally inexpensive, suitable for real-time animation, and works well for morphing highly dissimilar objects.

In the super-object technique[2] and the technique of minimizing distances of corresponding centroids and vertices,[4] the interpolation algorithms consider an object as a collection of independent points. Shrinkage usually occurs for two objects differing by a rotation. In the worst case the animated model can degenerate into a zero or negative volume. For example, consider the metamorphosis of two congruent tetrahedra, one of which is 180° opposite the other. In this case, an in-between tetrahedron can flip totally inside out, resulting in a negative volume. Figure 2 shows morphing one tetrahedron into another similar one at time $t = 0,0.2,0.4,0.6,0.8,1$, from right to left. The instance at $t = 0.4$ totally flips inside out.

From a theoretical point of view, we can consider that the Minkowski sum technique[5] and the alpha-shape geometric morphing[3] use linear interpolation in E^3 and E^4, respectively. The Minkowski sum technique has proved suitable for metamorphosis in which at least one object is convex. However, for metamorphosis between two non-convex objects, features common to both objects are not preserved. For example, when morphing from a dog to a cow, the mixed animal may have more than four legs! The reason can be best seen in Figure 3 in transforming a 'C' shape to itself. Furthermore, even when the two input objects are identical but concave, the in-between object is not the same as the input objects. Thus, the Minkowski sum approach is not *identity preserving*.

The alpha shape morphing method[3] has the drawback that the intermediate object generated may not be a valid polyhedron or consists of a collection of disjoint polyhedra, even though the two input objects are connected polyhedra.

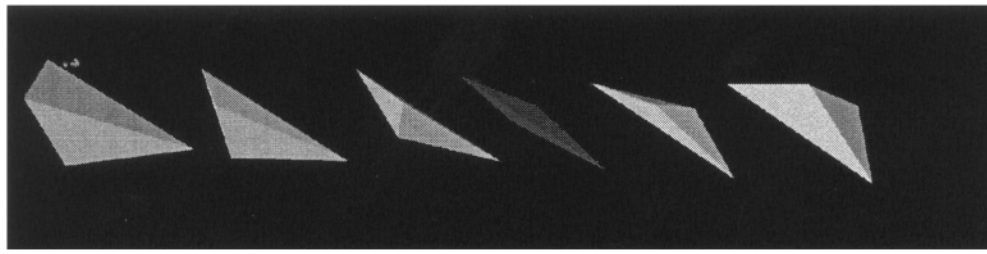

Figure 2. Interpolation between two tetrahedra by the linear vertex path

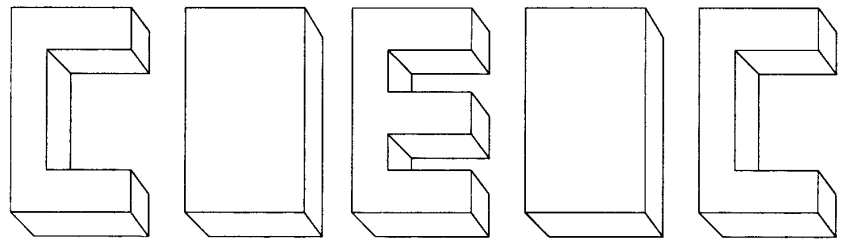

Figure 3. Minkowski sum for two 'C' shapes

(b) Independent interpolation of each vertex pair using a curved path

A Hermite cubic path with end tangents set equal to the normals at the vertices interpolated[6,7] and a quadratic Bézier vertex path[10] have also been proposed. Like using the linear vertex path, these approaches do not exploit relationships between vertices. These methods may work well for highly dissimilar objects but fail to preserve the features common to the two input objects.

(c) Intrinsic shape interpolation

The intrinsic shape interpolation for 2-D polygons has proved to be successful. It uses the intrinsic relation between the vertices of a polygon, described by edge lengths and vertex angles. It is shown[1] that interpolating the edge lengths and vertex angles generally produces more satisfactory in-between shapes than the linear vertex path would. However, this method works only for 2-D polygons with a one-to-one correspondence.

There are still some other special techniques. For example, the shape blending using the star-skeleton representation for 2-D polygons[11] solves many problems[11] encountered in the intrinsic shape interpolation[1] but it seems more difficult to generalize this method to blending of 3-D polyhedra.

3. THE PROPOSED SOLUTION

Let A and B be two input polyhedral models. In the metamorphosis between A and B, A is transformed to B continuously from $t = 0.$ to $t = 1$ under a prescribed correspondence between the vertices of the input polyhedra. In this paper, it is assumed that the correspondence is given as an input to the interpolation algorithm. Let $A \otimes_t^{\varphi} B$ be the in-between polyhedral model of A and B at time t under a correspondence φ. When there is no danger of confusion, we denote $M(t) = A \otimes_t^{\varphi} B$.

3.1. Basic criteria

The natural morphing process between two objects is, in general, not unique. For instance, there is no consensus on which of the morphing paths shown in Figure 1 is more natural. However, we believe that there are some basic criteria that a morphing algorithm should satisfy. In the following, we propose four such basic criteria, which have guided us in the search for a 3-D interpolation algorithm. We

will see that our interpolation algorithm satisfies the four criteria. Let $A_{\mathbf{x}}$ denote the translation of object A by vector $\mathbf{x}$. Let $\mathbf{R}_N(\alpha)$ denote the rotation about axis N by an angle α.

1. *Identity preserving*: $\forall t \in [0, 1]$, $A \otimes_t^{\varphi} A = A$.
2. *Translation invariant*: $\forall t \in [0, 1]$, $A_{\mathbf{x}} \otimes_t^{\varphi} B_{\mathbf{y}} = (A \otimes_t^{\varphi} B)_{\mathbf{z}}$ for some $\mathbf{z}$. Here $\mathbf{z}$ is a point on the line segment connecting $\mathbf{x}$ and $\mathbf{y}$. If the animation speed of the morphing process is uniform, $\mathbf{z} = (1 - t)\mathbf{x} + t\mathbf{y}$.
3. *Rotation invariant*: $\forall t \in [0, 1]$, $(\mathbf{R}_a(\alpha)A) \otimes_t^{\varphi} (\mathbf{R}_b(\beta)B) = \mathbf{R}_c(\gamma)(A \otimes_t^{\varphi} B)$ for some γ and $\mathbf{c}$.
4. *Feature preserving*: if there are features common to both input objects, the features should be preserved during the metamorphosis. One example is preserving the legs or the head of two different animals during morphing.

3.2. Basic ideas

In the interpolation algorithm, rather than being treated as a collection of independent points, a polyhedral model is considered as an *oriented plane graph*, called the *vertex adjacency graph (VAG)*. The vertex adjacency graph is composed of the vertices and edges of the polyhedron.[2] The *face adjacency graph (FAG)*, which can be thought of as the dual of the vertex adjacency graph, is used to represent the interrelations between faces. The nodes of the face adjacency graph and vertex adjacency graph are interrelated by intrinsic parameters, such as edge lengths, dihedral angles and interior angles. These parameters are invariant under rotation and translation, and locally defined. We will use the intrinsic parameters to interpolate the face adjacency graph and the vertex adjacency graph.

Let FAG^A, FAG^B, and $FAG^{M(t)}$ be the face adjacency graphs of A, B, and $M(t)$, respectively. Let VAG^A, VAG^B, and $VAG^{M(t)}$ be the vertex adjacency graphs of A, B, and $M(t)$, respectively.

It is possible to just interpolate the FAG^A and FAG^B to produce the in-between polyhedral model for animation and the algorithm is translation and rotation invariant, and also identity-preserving when the correspondence is the identity one. However, experimental results shown undesirable visual defects. We developed an alogrithm to solve the problem and achieve the same properties (please see Reference 12 for further details). The algorithm consists of two phases: (1) interpolating FAG^A and FAG^B to obtain $FAG^{M(t)}$, discussed in Section 4.1; (2) interpolating VAG^A and VAG^B using the information in $FAG^{M(t)}$ to obtain $VAG^{M(t)}$, discussed in Section 4.2. The in-between polyhedral model $M(t)$ is then uniquely determined by and generated from $VAG^{M(t)}$.

3.3. Representations of polyhedral models

A polyhedron can be represented as a planar graph. Since a planar graph can have more than one embedding on the plane, we will restrict ourselves to the particular embedding that represents the topology of the polyhedron. An embedded planar graph is called a *plane graph*. In order to distinguish a plane graph from its mirror-image graph, we assign an orientation[13] to the plane graph. This results in an *oriented plane graph*

Definition 3.1

An oriented plane graph is a plane graph in which each circuit bounding a region is a cyclically ordered set of links that bound the region counterclockwise.

Definition 3.2

Let P be a polyhedron. The vertex adjacent graph (VAG) of P is an oriented plane graph associated with P. Each node of the VAG represents a vertex of P and there is a link between two nodes if the corresponding vertices are linked by an edge of P. Each region of VAG corresponds to a face of P. The order of the links around the region of VAG is the same as that of the corresponding edges around the corresponding face in P. Each node contains the vertex coordinates.

Definition 3.3

Let P be a polyhedron. The face adjacency graph (FAG) of P is an oriented plane graph associated with P. Each node of the FAG represents a face of P and there is a link between two nodes if the corresponding faces share a common edge. A node contains the outward unit normal vector of the corresponding face. The link connecting two nodes contains a flag of value +(positive), –(negative), or =(coplanar), indicating that the corresponding faces form a convex dihedral angle, concave dihedral angle, or the two faces are coplanar, respectively.

We use the VAG as the primary representation of a polyhedral model since it represents a polyhedron uniquely. Assume that the flag in FAG takes only + or − for simplicity. Each region in the FAG corresponds to a vertex of the polyhedron. Note that a FAG cannot uniquely define a polyhedron. Additional geometric information is needed to make the FAG a complete representation of a polyhedron.[12] As plane graphs, FAG and VAG are the dual of each other, and the FAG of a polyhedron can be constructed if the VAG of the polyhedron is given. We say that two VAGs or two FAGs are *topologically equivalent* if the vertices, edges and faces of the underlying graphs can be put in a one-to-one correspondence that keeps the ordering of the bounding links surrounding every pair of corresponding regions.

Suppose polyhedron P has n faces and m vertices. The FAG and VAG of P will be denoted by *FAG* and *VAG*, respectively. A notation with the symbol of a particular polyhedron, such as A, B, and $M(t)$, as superscript, will refer to the element for that polyhedron. When without superscript, the notation defaults to the general genus zero polyhedron.

4. INTERPOLATION OF INTRINSIC PARAMETERS

A one-to-one correspondence between VAG^A and VAG^B is first assumed for simplicity; interpolation under the general correspondence is discussed in Section 5.2. Under a one-to-one correspondence, VAG^A, VAG^B, and $\text{VAG}^{M(t)}$ are topologically equivalent, so are FAG^A, FAG^B, and $\text{FAG}^{M(t)}$. In this case, $n^A = n^B = n^{M(t)}$ and $m^A = m^B = m^{M(t)}$, denoted as $\bar{n}$ and $\bar{\text{m}}$. We first discuss the computation of outward normals and vertex coordinates by intrinsic relations in Subsections 4.1 and 4.2. A method of stabilizing the algorithm is proposed in Section 4.3.

4.1. Interpolation of face adjacency graphs

Let $f_i^{M(t)}$ be the ith face of the in-between polyhedral model $M(t)$, and $N_i^{M(t)}$ be the outward unit normal of $f_i^{M(t)}$. By interpolating FAG^A and FAG^B, we mean to compute the normal vector $N_i^{M(t)}$ at time t, for $i = 0,1,\ldots,\bar{n}-1$. We will start with two initial faces, $f_0^{M(t)}$ and $f_1^{M(t)}$, and then use the intrinsic relation between faces to compute $N_i^{M(t)}$ for $i = 2,3,\ldots,\bar{n}-1$, by propagation.

Computing the first two normals

To proceed, we introduce the *outface normal* of an edge. Let f_b and f_a be two adjacent faces sharing an edge in a polyhedron. Let $T_{b,a}$ be the unit vector parallel to the edge and oriented counterclockwise with respect to the face f_b. The *outface normal* $e_{b,a}$ to the edge with respect to f_b is defined as the unit vector $T_{b,a} \times N_b$. That is, $e_{b,a}$ is parallel to face f_b and points towards the exterior of f_b. Figure 4 illustrates the definition in two different situations. Let $l_{b,a}$ be the dihedral angle between the faces f_b and f_a.

The normal vector $N_0^{M(t)}$ is interpolated using the shortest arc on the Gaussian sphere connecting N_0^A and N_0^B. $N_1^{M(t)}$ can be computed if the outface angle $l_{0,1}^{M(t)}$ and the outface normal $e_{0,1}^{M(t)}$ are known. The outface normal $e_{0,1}^{M(t)}$ is computed as follows. Since N_0^A and $e_{0,1}^A$ are orthogonal, $\mathbf{F}^A = \{N_0^A, e_{0,1}^A, N_0^A \times e_{0,1}^A\}$ is an orthogonal frame. Similarly, $\mathbf{F}^B = \{N_0^B, e_{0,1}^B, N_0^B \times e_{0,1}^B\}$ is also an orthogonal frame. Let $\mathbf{K}(t)$, t in [0,1], be the rotational motion about a fixed axis that rotates $\mathbf{F}^A$ into $\mathbf{F}^B$, i.e. $\mathbf{K}(0) = \mathbf{I}$ and $\mathbf{K}(1)\mathbf{F}^A = \mathbf{F}^B$. Then set $e_{0,1}^{M(t)} = \mathbf{K}(t)e_{0,1}^A$. The dihedral angle $l_{0,1}^{M(t)}$ is computed as $(1-t)l_{0,1}^A + e_{0,1}^{M(t)}$ by angle $l_{0,1}^{M(t)}$. To make the computation of $N_1^{M(t)}$ stable, we choose the first two faces such that $\min\{l_{0,1}^A, l_{0,1}^B\}$ is maximized.

Propagation

Suppose f_c, f_b, and f_a are three consecutive nodes around a region of the FAG. Let $\theta_{c,b,a}$ be the angle by which to rotate $e_{b,c}$ to $e_{b,a}$ about N_b, called the *outface angle*. Denote $\theta = \theta_{c,b,a}$ for short. Then N_c, N_b, and N_a are related by the following relations (see Figure 5):

$$e_{b,a} = \mathbf{R}_{N_b}(\theta)e_{b,c} \tag{1}$$

$$N_a = \mathbf{R}_{N_b \times e_{b,a}}(l_{b,a})N_b \tag{2}$$

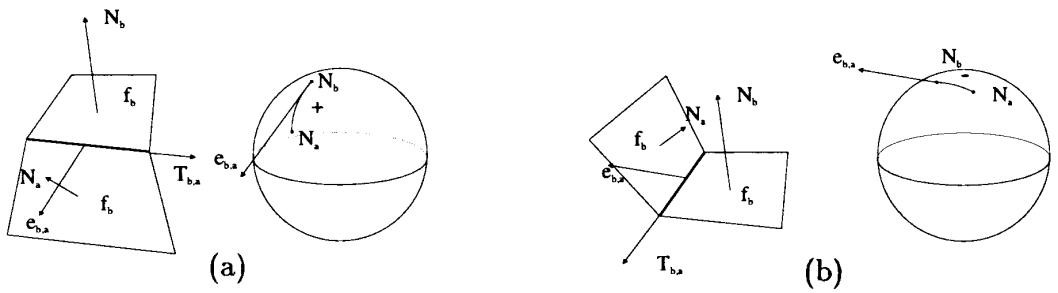

Figure 4. Definition of the outface normal $e_{b,a}$: (a) convex edge; (b) reflexive edge

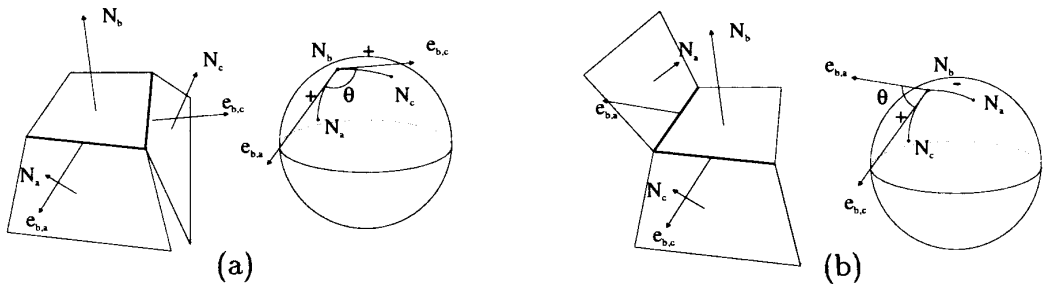

Figure 5. Relation of N_a, N_b, and N_c: (a) $f_a^{M(t)}$ and $f_b^{M(t)}$ share a convex edge; (b) $f_a^{M(t)}$ and $f_b^{M(t)}$ share a reflexive edge

Since the in-between polyhedron $M(t)$ should also satisfy these two relations, the equations

$$e_{b,a}^{M(t)} = \mathbf{R}_{N_b^{M(t)}}(\theta^{M(t)}) e_{b,c}^{M(t)} \tag{3}$$

$$N_a^{M(t)} = \mathbf{R}_{N_b^{M(t)} \times e_{b,a}^{M(t)}}(l_{b,a}^{M(t)}) N_b^{M(t)} \tag{4}$$

will be used to compute $N_a^{M(t)}$ when $N_c^{M(t)}$ are known. The terms $\theta^{M(t)}$and $l_{b,a}^{M(t)}$ used in equations (3) and (4) are computed by interpolation:

$$\theta^{M(t)} = (1-t)\theta^A + t\theta^B \tag{5}$$

$$l_{b,a}^{M(t)} = (1-t) l_{b,a}^A + t l_{b,a}^B \tag{6}$$

where θ^A, θ^B, $l_{b,a}^B$ are directly computed from FAG^A and FAG^B. Thus, $e_{b,a}^{M(t)}$ is determined by equation (3) and $N_a^{M(t)}$ is then determined by equation (4). In other words, given FAG^A and FAG^B, $N_a^{M(t)}$ can be determined if $N_b^{M(t)}$ and $N_c^{M(t)}$ are known. Consequently, $N_i^{M(t)}$, $i = 2,3,\ldots,\overline{n}-1$, can be computed once $N_0^{M(t)}$ and $N_1^{M(t)}$ are known. There are numerous choices of the order of computing $N_i^{M(t)}$, and breadth-first search is used in our implementation.

4.2. Interpolation of vertex adjacency graphs

Let $V_j^{M(t)}$ be the jth vertex of $M(t)$, and let $v_j^{M(t)}$ be the vertex co-ordinates of $V_j^{M(t)}$. Our goal is to compute the vertex co-ordinates $v_j^{M(t)}$ for $j = 0,1,\ldots,\overline{m}-1$. We first compute two starting vertices which are shared by the two starting faces in the interpolation of FAG^A and FAG^B. Then using the intrinsic relation between vertices, we compute $v_j^{M(t)}$ by propagation for $j = 2,3,\ldots,\overline{m}-1$.

Computing the first two vertices

Without loss of generality, we denote the two vertices of the edge shared by the faces f_0 and f_1 by V_0 and V_1, oriented such that V_1 follows V_0 counterclockwise around f_0. Let $l_{a,b}$ be the length of the edge $\overline{v_a v_b}$. First, set $v_0^{M(t)} = (1-t)v_0^A + tv_0^B$. Then $v_1^{M(t)}$ is computed such that the direction of $v_1^{M(t)} - v_0^{M(t)}$ is parallel to $N_0^{M(t)} \times e_{0,1}^{M(t)}$ and $\tilde{l}_{0,1}^{M(t)} = (1-t)\tilde{l}_{0,1}^A + t\tilde{l}_{0,1}^B$.

Propagation

Suppose V_c, V_b and V_a are three consecutive vertices of a face f_i. Let $\tilde{e}_{b,a}$ be the unit edge vector parallel to the direction of $v_a - v_b$ and let $\tilde{\theta}_{c,b,a}$ be the interior angle $\angle V_a V_b V_c$. When V_a, V_b, and V_c take the counterclockwise order around f_i, $\tilde{\theta}$ is negative; otherwise, it is positive. We write $\tilde{\theta} = \tilde{\theta}_{c,b,a}$ for short. Then v_c, v_b, and v_a are related by (Figure 6):

$$v_a = v_b + \tilde{l}_{b,a}\mathbf{R}_{N_i}(\tilde{\theta})\tilde{e}_{b,c} \tag{7}$$

This relation gives the equation

$$v_a^{M(t)} = v_b^{M(t)} + \tilde{l}_{b,a}^{M(t)}\,\mathbf{R}_{N_i^{M(t)}}(\tilde{\theta}^{M(t)})\tilde{e}_{b,c}^{M(t)} \tag{8}$$

to compute $v_a^{M(t)}$ when $v_b^{M(t)}$ and $v_c^{M(t)}$ are known. The terms $\tilde{\theta}^{M(t)}$ and $\tilde{l}_{b,a}^{M(t)}$ in equation (8) are computed by the interpolation

$$\tilde{\theta}^{M(t)} = (1-t)\tilde{\theta}^A + t\tilde{\theta}^B \tag{9}$$

$$\tilde{l}_{b,a}^{M(t)} = (1-t)\tilde{l}_{b,a}^A + t\tilde{l}_{b,a}^B \tag{10}$$

Thus, when $v_0^{M(t)}$ and $v_1^{M(t)}$ are known, $v_j^{M(t)}$, $j = 2,3,\ldots,\overline{m}-1$ can be determined by propagation. Again, breadth-first search is used for propagation.

As FAG and VAG are dual to each other, the interpolation of VAG should be similar to that of FAG. However, as the normal $N_b^{M(t)}$ used for computing $v_a^{M(t)}$ by equation (8) is stored in $\mathrm{FAG}^{M(t)}$, $\mathrm{FAG}^{M(t)}$ has to be computed first.

4.3. Stability

The interpolation of FAG or VAG is a one-pass algorithm. The result of the interpolation depends on the initial values and the order of computation. A small perturbation of initial values can introduce a big difference in later stages of computation. In equation (8), when $\|v_c^{M(t)} - v_b^{M(t)}\| \ll \tilde{l}_{b,a}^{M(t)}$, a perturbation $\delta v_c^{M(t)}$ to $v_c^{M(t)}$ can make a big change to $v_a^{M(t)}$. The same is true of equation (4). Therefore, if the intial values are not set properly, the in-between object can be highly distorted. We

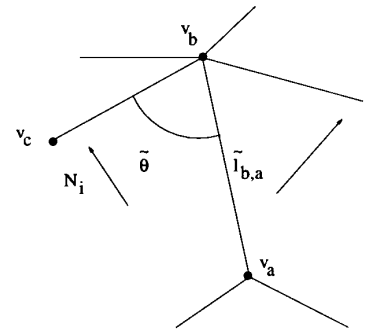

Figure 6. Relation of v_a, v_b, and v_c

circumvent this problem by using a heuristic to make the algorithm numerically more stable.

In the intermediate stage of interpolating VAG^A and VAG^B, suppose we are to compute the vector $v_a^{M(t)}$. Then, we first have to find a pair of vertices, $V_b^{M(t)}$ and $V_c^{M(t)}$, whose co-ordinates are already computed, such that $V_a^{M(t)}$, $V_b^{M(t)}$, and $V_c^{M(t)}$ are consecutive around a face of $\mathrm{VAG}^{M(t)}$. Let there be $\tilde{d}_a$ pairs of such vertices for $V_a^{M(t)}$, denoted by $V_{b_k}^{M(t)}$ and $V_{c_k}^{M(t)}$, $k = 0,1,\ldots,\tilde{d}_a - 1$ (see Figure 7). Each of these $\tilde{d}_a$ pairs is used to compute a $v_{a_k}^{M(t)}$ with a weight

$$\tilde{w}_{a_k} = \tilde{l}_{b_k,c_k}^{M(t)} / \tilde{l}_{b_k,a_k}^{M(t)} \tag{11}$$

The final $v_a^{M(t)}$ is computed as

$$\frac{\sum_{k=0}^{\tilde{d}_a-1} \tilde{w}_{a_k} v_{a_k}^{M(t)}}{\sum_{k=0}^{\tilde{d}_a-1} \tilde{w}_{a_k}}$$

The choice of the weight by equation (11) is made so that a pair $v_{c_k}^{M(t)}$, $v_{b_k}^{M(t)}$ with smaller length $\|v_{c_k}^{M(t)} - v_{b_k}^{M(t)}\|$ contributes less to the final $v_a^{M(t)}$. The same principle applies to the interpolation of FAG^A and FAG^B. That is, $N_a^{M(t)}$ is assigned the direction of $\Sigma w_{a_k} N_{a_k}^{M(t)}$ with the weight w_{a_k} given by

$$w_{a_k} = l_{b_k,c_k}^{M(t)} / l_{b_k,a_k}^{M(t)} \tag{12}$$

Experiments show that the above remedy greatly alleviates the numerical instability of $v_a^{M(t)}$ and $N_a^{M(t)}$, and leads to a more robust algorithm.

5. COMPLETE ALGORITHM

5.1. General correspondence

In general, for two input polyhedra A and B, VAG^A and VAG^B are not necessarily topologically equivalent. Then a general correspondence has to be used, in which

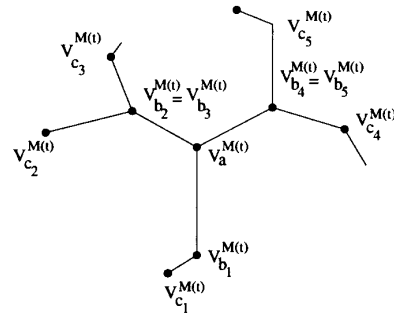

Figure 7. Computing $v_a^{M(t)}$

one vertex of A may correspond to many vertices, edges or faces of B, and vice versa. Our approach to the case of the general correspondence follows the super-object approach,[2] which also discusses the problem of establishing a correspondence.[2,12] Two concepts are needed to define a general correspondence: the *vertex correspondence mapping* and the *super-graph*.

Let M be a polyhedral model. Let $M.V$ and $A.V$ be the set of nodes of VAG^M and VAG^A, respectively. The terms *vertex* and *node* will be used interchangeably in the section. A vertex correspondence mapping $hv^A{:}M.V \rightarrow A.V$ is defined as $hv^A(x) = y$ if vertex x of VAG^M corresponds to the vertex y of VAG^A. We use $hv^A(VAG^M)$ to denote the graph with its set of nodes being the range of hv^A and its set of links being all the links mapped from those of VAG^M.

Let $M.E$ and $A.E$ be the set of links VAG^M and VAG^A, respectively. Let $M.F$ and $A.F$ be the set of faces of VAG^M and VAG^A, respectively. We call $hv^A{:}M.V \rightarrow A.V$ a *valid vertex correspondence mapping* if it satisfies the following conditions: (1) hv^A is onto; (2) If $(m_1,m_2) \in M.E$, then either $(hv^A(m_1),hv^A(m_2)) \in A.E$ or $hv^A(m_1) = hv^A(m_2) \in A.V$; (3) For any $e \in A.E$, $\exists(m_1,m_2) \in M.E$ such that $(hv^A(m_1),hv^A(m_2)) = e$; and (4) For every face $r \in M.F$, r is mapped to either a face with the same orientation, or an edge, or a vertex of A. If hv^A satisfies conditions 1, 2, and 3, then $hv^A(\mathrm{VAG}^M)$ is isomorphic to VAG^A, and the only difference between $hv^A(\mathrm{VAG}^M)$ and VAG^A, as oriented plane graphs, may be the embeddings and orientation. If hv^A further satisfies condition 4, then $hv^A(\mathrm{VAG}^M)$ is topologically equivalent to VAG^A. In this case, we call VAG^M a *super-graph* of VAG^A.

The general correspondence between A and B is defined by (1) a super-graph VAG^M of both VAG^A and VAG^B; and (2) two valid vertex correspondence mappings $hv^A{:}M.V \rightarrow A.V$ and $hv^B{:}M.V \rightarrow B.V$. In this paper, it is assumed that the correspondence is given as an input; the super-graph VAG^M is used as the underlying graph of $\mathrm{VAG}^{M(t)}$ and the mappings hv^A and hv^B are known.

5.2. Interpolation under a general correspondence

To interpolate VAG^A and VAG^B under a general correspondence, we first replace VAG^A and VAG^B by two vertex adjacency graphs which are topologically equivalent to $\mathrm{VAG}^{M(t)}$, so that the method used for the case of one-to-one correspondence can be directly applied. However, this may cause degeneracy problems in interpolating the intrinsic parameters and using the relations (3), (4), (5), (8) and (9). We will resolve it by using the intrinsic shape information in one object to approximate the degenerate part of another object.

To apply the method for one-to-one correspondence here, VAG^A is replaced by a vertex adjacency graph, denoted VAG^{M_A}, which is topologically equivalent to $\mathrm{VAG}^{M(t)}$ such that if $hv^A(V_a^M) = V_b^A, v_a^{M_A} = v_b^A$. FAG^{M_A} is constructed from VAG^{M_A}. We renumber the vertices of VAG^{M_A} and faces of FAG^{M_A} in such a way that $V_j^{M_A}$ of VAG^{M_A} corresponds to $V_j^{M(t)}$ of $\mathrm{VAG}^{M(t)}$, and $f_i^{M_A}$ of FAG^{M_A} corresponds to $f_i^{M(t)}$ of $\mathrm{FAG}^{M(t)}$. VAG^B is replaced by a similarly defined VAG^{M_B}. FAG^{M_B} is constructed similarly.

Unfortunately, the edges and faces of VAG^{M_A}, VAG^{M_B}, FAG^{M_A}, and FAG_B^M may be degenerate. If a face $f_i^{M_A}$ is mapped to an edge or a vertex of A, the normal vector $N_i^{M_A}$ is not well defined; consequently, the related outface angles and outface normals are not defined. If an edge $(V_a^{M_A}, V_b^{M_A})$ of VAG^{M_A} is mapped to a vertex of

VAG^A, the edge tangent $\tilde{e}^{M_A}_{a,b}$ is not defined and neither are the related interior angles. The same problem occurs in VAG^{M_B} and FAG^{M_B}. Thus, equations (3), (4) and (5) cannot be used for the interpolation of FAG^{M_A} and FAG^{M_B}, neither can equations (8) and (9) for the interpolation of VAG^{M_A} and VAG^{M_B}. We overcome this problem by adapting the intrinsic shape information in one object to the corresponding degenerate part of another object.

The method of approximating the undefined outward normals $N^{M_A}_a$ and $N^{M_B}_a$ is similar to the idea of interpolating FAG^A and FAG^B in Subsection 4.1. Assume that there is at least one pair of well-defined corresponding outface normals $e^{M_A}_{b,a}$ and $e^{M_B}_{b,a}$ to serve as the start point of the approximation. This condition is satisfied in most practical situations.

Suppose $f^{M_A}_a$ is mapped to a vertex or an edge of VAG^A; thus $N^{M_A}_a$ is not defined. Assume that $f^{M_B}_a$ is well defined, for otherwise, the super-graph can always be simplified so that the assumption holds. To approximate $N^{M_A}_a$, first find two pairs of corresponding faces, $f^{M_A}_c$, $f^{M_B}_c$, and $f^{M_A}_b$ and $f^{M_B}_b$, with well-defined outward normals such that $f^{M_A}_c$, $f^{M_A}_b$ and $f^{M_A}_a$, and $f^{M_B}_c$, $f^{M_B}_b$, and $f^{M_B}_a$ are consecutive around corresponding regions in FAG^{M_A} and FAG^{M_B}, respectively. Let the total number of such pairs be κ_a. Denote these pairs of faces as $f^{M_A}_{b_k}$ and $f^{M_A}_{c_k}$, $k = 0,1,\ldots,\kappa_a - 1$. Each of these pairs is used to compute one $N^{M_A}_{a_k}$. Here we simply use θ^{M_A} to denote $\theta^{M_A}_{c_k,b_k,a_k}$. To simulate the non-degenerate case, the normal $N^{M_A}_{a_k}$ can be computed by $e^{M_A}_{b_k,a_k} = \mathbf{R}_{N^{M_A}_{b_k}}(\theta^{M_A}) e^{M_A}_{b_k,c_k}$ and $N^{M_A}_{a_k} = \mathbf{R}_{N^{M_A}_{b_k} \times e^{M_A}_{b_k,a_k}}(l^{M_A}_{b_k,a_k}) N^{M_A}_{b_k}$. However, as the outface normal $e^{M_A}_{b_k,a_k}$ and the outface angle θ^{M_A} in the first equation are not defined, we replace θ^{M_A} by its counterpart in VAG^{M_B}, i.e. set $\theta^{M_A} = \theta^{M_B}$. As a result, $e^{M_A}_{b_k,a_k}$ is computed by $e^{M_A}_{b_k,a_k} = \mathbf{R}_{N^{M_A}_{b_k}}(\theta^{M_B}) e^{M_A}_{b_k,c_k}$, $k = 0,1,\ldots,\kappa_a - 1$. The outward normal $N^{M_A}_a$ is assigned the unit vector of $\Sigma_{k=0}^{k_a-1} N^{M_A}_{a_k}$. Now, the outface normals and edge tangents related to $f^{M_A}_a$ can be determined from $N^{M_A}_a$ and the related outface angles and interior angles can also easily be determined. The undefined quantities in VAG^{M_B} and FAG^{M_B} are approximated similarly.

6. RESULTS AND DISCUSSION

Property 6.1

If A and B are identical polyhedra and the given correspondence is the identity correspondence, i.e. all vertices, edges, and faces of A correspond to their counterparts of B, then $M(t) = A = B$ for all $t \in [0,1]$.

The proof is trivial and therefore omitted. By this property, our algorithm is identity preserving.

As we can see, all operations—approximating normals, computing initial normals and vertex co-ordinates, stabilized propagation of computing normals and vertex co-ordinates—involve only the interpolation of intrinsic parameters, which are invariant under rotation and translation. So, we can prove the following two properties.

Property 6.2

The intrinsic interpolation is rotation invariant.

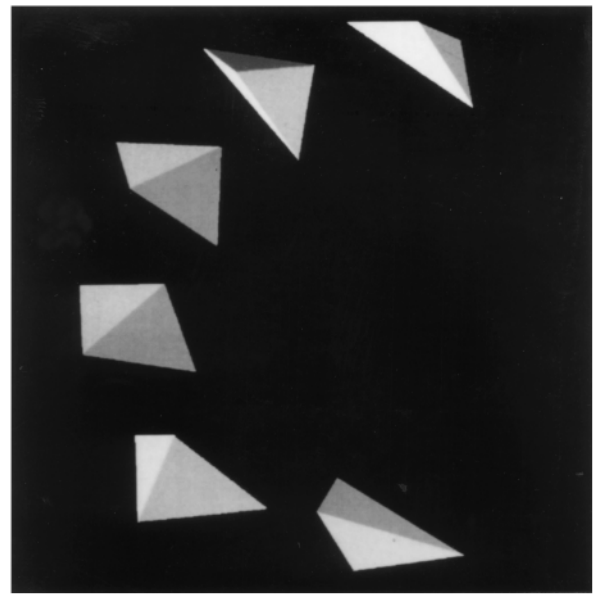

Figure 8. Interpolation between two tetrahedra

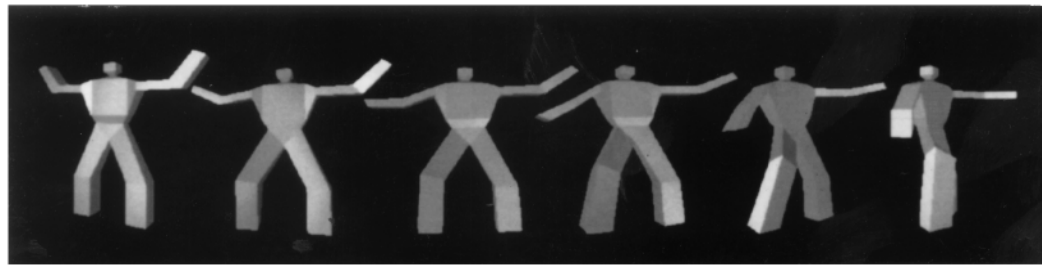

Figure 9. Interpolation between two human figures

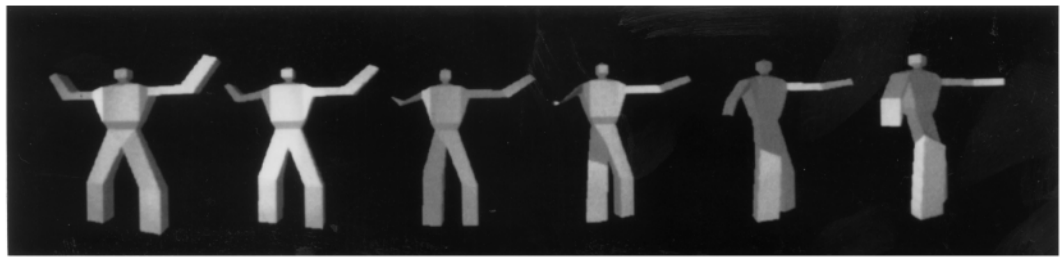

Figure 10. Interpolation between two human figures using linear vertex path

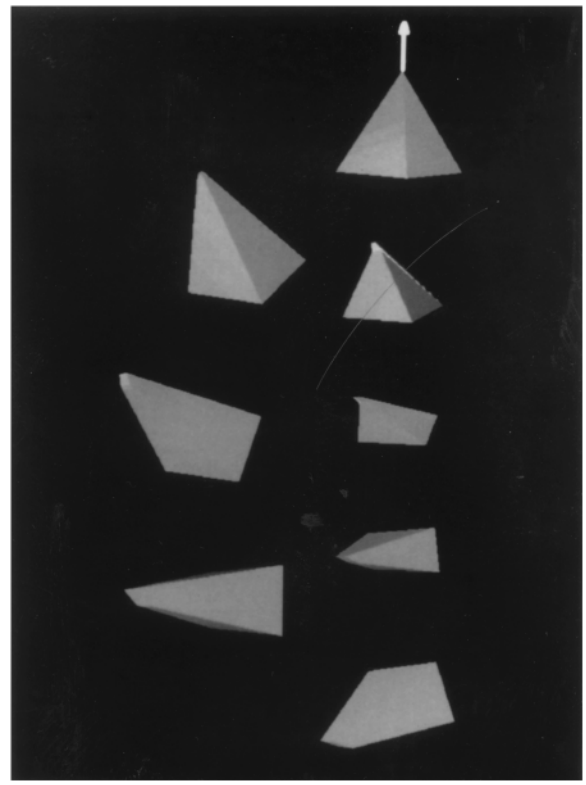

Figure 11. Morphing under a general correspondence

Property 6.3

The intrinsic interpolation is translation invariant, if the centre of gravity of $M(t)$ is put at $(1-t)G^A + tG^B$, where G^A and G^B are the centres of gravity of A and B, respectively.

Experimental results

The algorithm has been implemented and tested on SGI Indigo/XZ graphics workstations with R4000 CPU. Experiments shown that the algorithm works well for many cases where the two morphed objects have similar features, and if the correspondence is properly specified, the features are preserved during morphing. Figure 1 is an example showing different ways of morphing between two objects, in which the only difference is the correspondence. Figure 8 shows the interpolation of two tetrahedra that are used in Figure 2. The objects are placed along the curved path of motion for illustration. Notice that the in-between tetrahedron is rotating and deforming at the same time, thus avoiding the flipping-inside-out problem that appears in Figure 2 using the linear vertex path.

Figure 9 shows the interpolation of two human figures facing us. The figure is rotating his body, and meanwhile, turning his right arm. Figure 10 shows the interpolation of the same Figure as in Figure 9, using the linear vertex path. Notice

that the in-between human figures are unnaturally thin and the right arm is severely distorted. Thus, a distinct advantage of our algorithm is that if it is capable of turning the arm and the body in two different directions simultaneously.

One is often tempted to think[5] that properly tuning the orientation of one object and simply using linear vertex path or Minkowski sum can produce a desired in-between model. However, a model like the one in Figure 9 may have a lot of prominent features, e.g. body and arms, turning in different directions at the same time. In this case, the approach of tuning the relative orientation of the input objects can never produce a desired in-between object. Thus, that our intrinsic interpolation algorithm is able to turn the arm and the body in two different directions at the same time is a distinct advantage.

Figure 11 shows the morphing of two solids at the top and bottom under a general correspondence with the linear interpolation shown on the right and the intrinsic interpolation on the left. The approximated normal is shown as the white arrow. The flipping inside out problem on the right at $t = 0.2$ is avoided by the intrinsic interpolation. Figure 12 shows how the degenerated normals are approximated at a convex edge and a concave edge in morphing with the linear interpolation shown on the right and the intrinsic interpolation on the left. Shrinkage and flip-around are avoided in the interpolation.

7. CONCLUSIONS AND FUTURE RESEARCH

While the criteria for morphing can be artistic, this paper proposes four basic criteria a morphing algorithm should satisfy, and presents an interpolation algorithm for 3-D morphing of polyhedral models. The interpolation algorithm is translation and

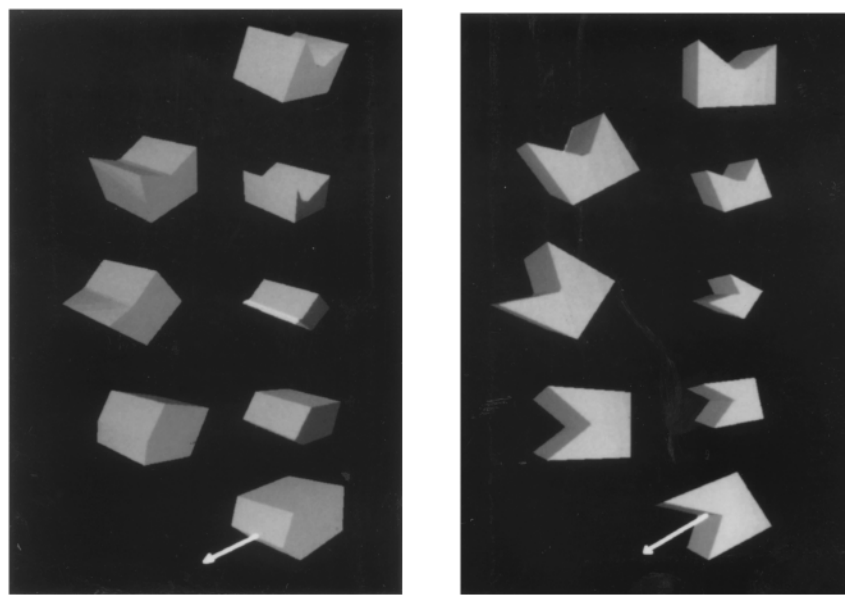

Figure 12. Approximating normal: (a) convex edge; (b) reflexive edge

rotation invariant, and is identity preserving if the correspondence is the identity correspondence. It is also demonstrated in practice that the algorithm preserves the characteristic features common to both input objects and avoids the shrinkage problem with the linear vertex interpolation.

Several aspects of the interpolation problem are still under investigation. First, what are the other possible representations for the intrinsic relations? Second, can we eliminate the dependency on the order of searching uncomputed nodes in the interpolation of FAG or VAG? Third, there is no guarantee that the in-between object produced by our algorithm does not have self-intersection. Up to now, no known published work on the interpolation problem can effectively solve this problem.

Another problem that is currently under investigation is the design of a correspondence algorithm that can work well together with our intrinsic interpolation algorithm.

REFERENCES

1. T. W. Sederberg, P. Gao, G. Wang and H. Mu, '2-D shape blending: an intrinsic solution to the vertex path problem', *SIGGRAPH '93*, pp. 15–18.
2. E. W. Bethel and S. P. Uselton, 'Shape distortion in computer-assisted keyframe animation', in N. Magnenat-Thalmann, and D. Thalmann (eds), *State of the Art in Computer Animation*, Springer-Verlag, NY, 1989, pp. 215–224.
3. H. Edelsbrunner and P. Fu, Geometric morphing with alpha shapes', *Private communication*.
4. T. M. Hong, N. Magnenat-Thalmann and D. Thalmann, 'A general algorithm for 3-D shape interpolation in a facet-based representation', *Proceedings of Graphics Interface '88*, pp. 229–235.
5. A. Kaul and J. Rossignac, 'Solid-interpolating deformations: construction and animation of PIPs', in F. H. Post and W. Barth (eds), *Proc. Eurographics '91*, Elsevier Science Publishers B.V, 1991, pp. 493–505.
6. J. R. Kent, R. E. Parent and W. E Carlson, 'Establishing correspondences by topological merging: a new approach to 3-D shape transformation', *Proceedings of Graphics Interface '91*, Calgary, Alberta, June 1991, pp. 271–278.
7. J. R. Kent, W. E. Carlson and R. E. Parent, 'Shape transformation for polyhedral objects', *SIGGRAPH '92*, pp. 47–54.
8. Apostolos Lerios, Chase D. Garfinkle and Marc Levoy, 'Feature-based volume metamorphosis', *SIGGRAPH '95*, pp. 449–456.
9. T. W. Sederberg and E. Greenwood, 'A physically-based approach to 2-D shape blending', *SIGGRAPH '92*, pp. 47–54.
10. G. Slinker, 'Inbetweening using a physically based model and nonlinear path interpolation', *Master's Thesis*, Brigham Young University, Dept. of Comp. Sci., 1992.
11. M. Shapira and A. Rappoport, 'Shape blending using the star-skeleton representation', *IEEE Computer Graphics and Applications*, **15**(2), 44–50 (1995).
12. Y. M. Sun, 'Metamorphosis of polyhedral models using intrinsic shape parameters', *Master's Thesis*, Comp. Sci. Dept., The Universtiy of Hong Kong, 1995.
13. L. Weinberg, 'A simple and efficient algorithm for determining isomorphism of planar triply connected graphs', *IEEE Trans. Circuit Theory*, **13**(2), 142–148 (1966).
14. B. K. P. Horn, 'Extended Gaussian images', *Proceedings of IEEE*, **72**, (12), 1671–1686 (1984).