Establishing Correspondences by Topological Merging: A New Approach to 3-D Shape Transformation

James R. Kent
Richard E. Parent
Wayne E. Carlson
Department of Computer and Information Science
Advanced Computing Center for the Arts and Design
The Ohio State University
Columbus, Ohio 43210

Abstract

This paper presents a new technique for computing transformations between two polyhedral models. Unlike previous techniques, the algorithm described herein employs both topological and geometric data from the original models in establishing vertex, edge, and face correspondences between the objects. This results in transformations which maintain their connectivity at intermediate steps of the transformation and which display far less distortion than those obtained using previous algorithms.

Keywords: Computer Animation, Computer-Aided Geometric Design, Interpolation, Shape Transformation.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling; I.3.7 - [Computer Graphics]: Three-Dimensional Graphics and Realism

1.0 Introduction

The ability to algorithmically transform one shape into another is quite useful in animation and design. Traditional animation makes extensive use of shape transformation as a special effect. Growth processes, such as a tadpole turning into a frog, can be modeled by smoothly interpolating between models of the object at various stages of maturity. In industrial design, Chen [Chen89] has used shape transformations to explore new forms, (e.g. by interpolating a teardrop and a car body to achieve a more stream-lined effect), and to extract trends in design (e.g. by blending objects with similar purposes to obtain the shape of an average object).

This paper presents a new algorithm which, given two 3-D models, generates two new models which have the same shape as the original ones, but which allow transformations from one to another to be easily computed. Currently, the models are limited to star-shaped[1] polyhedral solids with no holes, but the basic concepts involved are applicable for arbitrary polyhedral models.

2.0 Previous Work

Most of the literature on shape modification in computer graphics refers to the manipulation of a single model. These algorithms fall into two categories, methods which rely on a user-specified distortion of a grid of controlling vertices [Burt76] [Pare77] [Sede86] [Coqu90], and methods which employ physically-based models [Barr84] [Weil86] [Terz87] [Haum88] [Terz88] [Plat88]. These techniques do not address the problem being considered in this paper (i.e. that of smoothly transforming one model into another).

Reeves [Reev81] describes a technique for smoothly transforming one curve into another over time. The two curves can be viewed as the boundaries of a Coons' patch defined over a u-v rectangular parametric space (i.e. the two curves are the isoparametric curves, v=0 and v=1). The v direction of the patch is taken to be time, and the isoparametric curves, v=constant, are used as the in-between shapes. A key notion introduced in this paper is that of establishing point correspondences between two objects, which in this case are user-specified. Reeves' technique works well for single curves, but does not easily extend to surfaces or solids.

Hong [Hong88] matches the centroids of the faces of two objects to establish correspondences between the objects. These correspondences are then used to transform one face into another. When one object has more faces than the other, the extra faces are paired with the closest vertex in the other model. For carefully selected objects, this technique is effective, but for arbitrary objects, severely distorted shapes occur during the inter-

1. A *star-shaped* polyhedral object is one for which some interior point, p, exists such that any semi-infinite ray originating at p intersects the surface of the object at exactly one point. Note that the set of convex polyhedra is a proper subset of the set of star-shaped objects

polation. Additionally, the faces of the in-between objects may break apart and appear to fly randomly from their initial position on one object to their final position on the other object during the transformation.

Terzides [Terz89] uses a very primitive shape transformation technique to investigate the use of interpolated shapes in architectural design. The technique associates the vertices of the two objects based on their index in the vertex coordinate arrays of the objects. This is clearly a very limited technique which will only provide useful results if explicit knowledge of the transformation process is used when modeling the objects.

Chen [Chen89] approaches the problem of transforming shapes by first creating coaxial 2-D slices through each object. The number of slices is user-specified. Corresponding slices from the two objects are transformed to have a common center point. New points are inserted in each slice at the point where the ray through each vertex of one slice intersects the other slice. Each of these pairs of points are interpolated to create intermediate slices, which are combined to create in-between objects using a lofting technique. No actual examples of the 3D algorithm are given and it is not clear how the original slices are obtained for arbitrary objects. Also, this method does not work for most non-convex objects, since a slice through a non-convex object may result in multiple rings of edges.

Parent [Pare91] describes a solution to the shape transformation problem for arbitrary 3-D polygonal models[2]. His method involves first breaking the surface of the two objects into equal numbers of sheets of connected faces. New vertices are added along the boundaries of these sheets until there is a one-to-one correspondence of the points along the boundary of each pair of sheets. The corresponding pairs of sheets are subdivided by first finding a path of edges on one sheet which connects two vertices on the boundary of the sheet. A path of edges which connects the corresponding two vertices of the other sheet is then found, and vertices are added to the newly constructed paths so that a one-to-one point correspondence is obtained. These paths split each original pair of sheets into two new pairs of sheets with one-to-one correspondences between the points on their boundaries. The subdivision process continues recursively until each sheet contains a single face. The subdivided, single-face sheets are then recombined, resulting in two objects which have the same shape as the original objects, but which share a common vertex-edge-face structure. The transformation between the two shapes is computed by interpolating the corresponding vertex positions of these two objects. The main drawback of this algorithm is that small regions of one model often map to large regions of the other, which results in severely distorted in-between shapes.

---

2. The models must be valid in the sense of obeying Euler's formula (*Euler validity*):

(# of Vertices) - (# of Edges) + (# of Faces) =
2 - 2 * (Genus of Object)

A comparison of the four 3D shape transformation algorithms discussed above (i.e. [Hong88], [Terz89], [Chen89], and [Pare91]) with the one described in this paper is made in Section 8.0.

3.0 Geometry and Topology

The models being discussed in this paper are three-dimensional Euler-valid solids which are bounded by planar faces (i.e. polyhedral solids). Such models are defined by giving the relative coordinates of each vertex of the model and describing the connectivity of the vertices in terms of edges and faces. In this paper, the term *topology* is used when referring only to the connectivity information. The term *geometry* is used when referring to a specific instance of the topology for which the relative vertex coordinates have been specified.

This use of the term topology was established in the graphics literature by Weiler [Weil84]. It is quite different from the traditional meaning of topology used in mathematics when referring to properties of surface manifolds.

The distinction between changes in the topology of a model (altering the number of vertices and/or the connections between them) and changes in the geometry of a model (altering the relative locations of the vertices) is important. For example, rotations and translations do not change either the topology or the geometry of the object. Scaling and shear operations change the geometry, but leave the topology unaltered. Boolean operations alter the topology and geometry of objects.

4.0 The Shape Transformation Problem

The general problem of transforming one polyhedral shape into another can be viewed as having two parts that will be referred to as the *correspondence problem* and the *interpolation problem*. The correspondence problem is that of establishing which surface elements of one object will be transformed into which surface elements of the other object. The interpolation problem is concerned with the actual transformation of corresponding surface elements. The two problems are interrelated since the method used to solve the interpolation problem is dependent upon the manner in which the surface element correspondences are established.

Solutions to the shape transformation problem should be judged by the following criteria:

1) Is face connectivity maintained for all intermediate shapes?
2) Are the intermediate shapes overly distorted?
3) What restrictions on the original models exist?

The first criteria is essential for most applications. Of the four 3D transformation algorithms discussed in Section 2.0, only [Chen89] and [Pare91] satisfy the first criteria.

These two algorithms, as well as the one presented in this paper, use the same overall approach to solve the correspondence and interpolation problems. The correspondence problem is solved

by adjusting the topology of each original model until the vertex-edge-face interconnection networks of the two objects are identical (i.e. there are one-to-one vertex, edge, and face correspondences). Viewed another way, the solution to the correspondence problem generates two new objects which have the same shape as the original objects, but which share the same topology.

Solving the correspondence problem in this manner leads to a simple solution to the interpolation problem. Since the vertices of the modified objects match one-to-one, any smooth interpolation scheme between corresponding vertices (linear, spline curve, ease-in/ease-out, etc.) will cause the objects to smoothly transform from one to the other. Thus, [Chen89], [Pare91] and this paper focus on different techniques for solving the correspondence problem.

Two potential problems may arise during the interpolation process. First, for faces with more than three edges, interpolating vertices from one position to another will not guarantee that the faces maintain planarity. This may or may not be a problem depending on the rendering mechanism. If it is a problem, a pre-interpolation processing step can be used to triangulate the objects. Second, an object may penetrate itself during the interpolation. This usually occurs if one or both objects are extremely concave. No general solution to this problem is known, but providing the ability to control the rate at which each vertex moves from its initial to its final position can often cause interpenetrations to be avoided.

5.0 An Algorithm for Establishing Correspondences by Topological Merging

The algorithm for solving the correspondence problem is first briefly described, followed by a more detailed description of the key steps. Throughout the description, the original polyhedral models are referred to as M1 and M2. M1 (M2) has V1 (V2) vertices, E1 (E2) edges, and F1 (F2) faces. Except where noted, the algorithm will work for arbitrary polyhedral solids with no holes. Pseudocode for the correspondence algorithm is shown in Figure 1.

The algorithm begins by projecting each model onto the surface of a unit sphere. These projected models are referred to as PM1 and PM2, with vertices PV1 and PV2, respectively. A modified Weiler polygon clipping algorithm is then used which clips each face of PM1 to the faces of PM2. The clipped faces of PM1 form a complete covering of the unit sphere with vertices PV' = (PV1 $\cup$ PV2 $\cup$ arc-arc intersection points). These clipped faces tile the original faces of each model. Taking the union of these faces yields a common topology which has the required one-to-one vertex, edge, and face correspondences. The final step involves projecting this merged topology back onto the surface of each original model.

As can be seen from Figure 1, the algorithm runs in O(E1*E2) time, as each arc of PM1 is tested for intersection with each arc of PM2. It is therefore important to the performance of the algorithm to include tests which can quickly determine when two arcs do not intersect. In the current implementation, each projected vertex is labeled with a tag as to which octant it lies in, and these tags are used to quickly identify cases in which two arcs can not intersect.

```
Read in the Topology and Geometry of Each Model

For Each Edge of Each Model
   Project the Edge onto the Unit Sphere

(Modified Weiler Clipping Algorithm)
For Each Projected Edge (Arc) of Model 1
   For Each Projected Edge (Arc) of Model 2
      Calculate Arc/Arc Intersection Point
      If Arcs Intersect
         Subdivide Arcs at Intersection Point
         Adjust Edge Topology

For each Projected Vertex, PV
   If PV is not an Original Vertex of Model 1
      Map PV to Model 1's Surface
   If PV is not an Original Vertex of Model 2
      Map PV to Model 2's Surface
```

Figure 1 - Pseudocode for the Correspondence Algorithm

Figure 2 shows two polyhedral models, a cube and a soccer ball, before and after the correspondence algorithm is run.

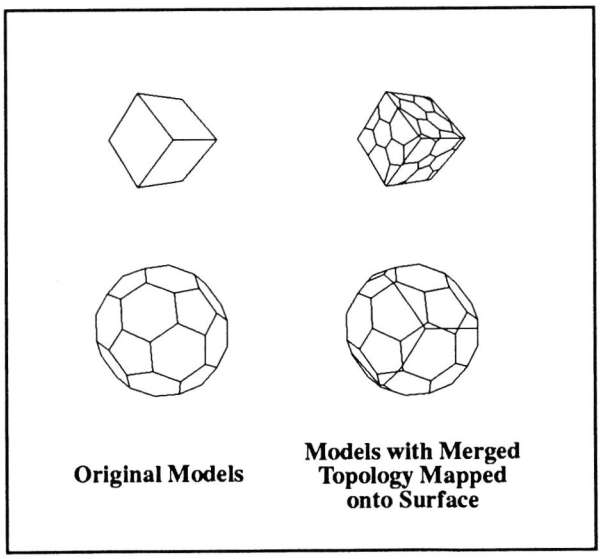

Figure 2 - An Example of Topological Merging

5.1 Projecting Onto the Unit Sphere

The first step in the algorithm is to project the topology of each model onto the surface of the unit sphere. This projection must be done in such a manner that no edges of the projected model intersect. In addition, the projected faces of each model must completely tile the surface of the sphere. For star-shaped polyhedra without holes, a simple projection exists, and is described below. A suitable projection technique has not been found for

general concave polyhedra without holes or for polyhedra with holes. Section 7.0 discusses this issue further.

For star-shaped polyhedra, at least one interior point exists from which all the vertices are visible. (For convex polyhedra, all vertices are visible from all interior points). Such a point is chosen as the center of the object. For each vertex, a ray is constructed which originates at this center point and passes through the vertex. The vertex is moved along this ray in a positive or negative direction until it is at unit distance from the center. Performing this action on all vertices of the model generates a new model whose vertices lie on the surface of the unit sphere. The edges of each model project to arcs on the surface of the sphere (i.e. the arc which is obtained by projecting any point on the edge outward from the center point until it is at unit distance). The projected faces are arc-bounded patches which completely tile the surface of the sphere.

Any point may be selected as the center point as long as it is strictly in the interior of the model and all vertices of the model are visible from it. For convex polyhedra, finding the centroid by averaging all vertex coordinates will produce a suitable center. For concave star-shaped polyhedra, the center point must be specified either implicitly or explicitly, and checked for validity by ray-casting. In either case, sllowing the user control over the location of the center point is desirable, since selecting different center points will produce different transformations, as can be observed by comparing Figures 6 and 9.

5.2 Merging the Topologies (A Modified Weiler Polygon Clipping Algorithm)

In [Weil77] and [Weil80], an algorithm is described which clips one polygon against another polygon. For our purposes, we desire a version of this algorithm which clips arc-bounded patches on the surface of a sphere to other arc-bounded patches. By clipping each patch of one projected object against all the patches of the other object, a set of patches which completely tiles the original patches as well as the surface of the sphere is formed. The union of these patches forms the desired merged topology for the transformation.

The first step in performing the clipping involves finding all arc-arc intersections between the arcs of PM1 and the arcs of PM2. Consideration of the geometry of the projected objects provides a straightforward technique for computing these intersections. Each edge of the original models projects onto an arc of the unit sphere which lies in the plane containing the center point of the sphere and the two endpoints of the arc. This is illustrated in Figure 3, where O is the origin of the unit sphere, A and B are the endpoints of an edge, and PA and PB are the projected endpoints of that edge.[3] Thus, any projected edge (i.e. an arc) of model 1 is contained in a plane P = PA t + PB u, where PA and PB are the projected endpoints of that edge. Similarly, some plane Q = PC v + PD w, contains the projected edge of model 2 with projected endpoints PC and PD.

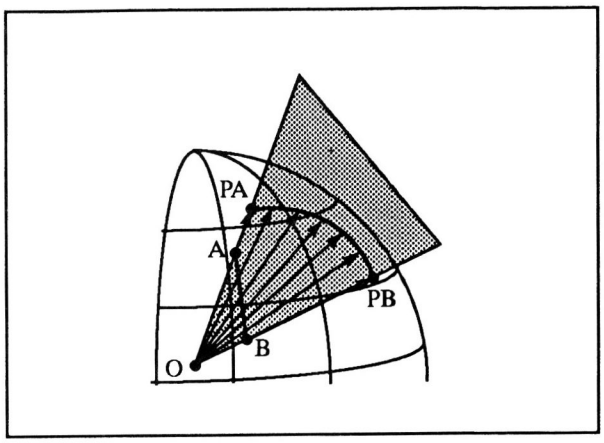

Figure 3 - Projecting an Edge onto the Unit Sphere

If **P** and **Q** are not coplanar, then their intersection is a line. One point on this line is the center of the unit sphere, **O**. A second point, **X**, on the line can be found by solving for a point where the plane equations are equal. Setting **P** = **Q**, yields the following three equations:

$$\mathbf{PA}_x\, t + \mathbf{PB}_x\, u = \mathbf{PC}_x\, v + \mathbf{PD}_x\, w$$
$$\mathbf{PA}_y\, t + \mathbf{PB}_y\, u = \mathbf{PC}_y\, v + \mathbf{PD}_y\, w$$
$$\mathbf{PA}_z\, t + \mathbf{PB}_z\, u = \mathbf{PC}_z\, v + \mathbf{PD}_z\, w$$

Since there are 3 equations and four unknowns (t, u, v, and w), t is set to 1.0,[4] and the other three are computed using an equation solving technique such as Kramer's rule. This establishes point **X** = **PA** t + **PB** u.

If the arcs intersect, their point of intersection, **I**, must lie on the ray from **O** to **X** at unit distance from **O** (i.e. on the surface of the unit sphere). Once point **I** is determined, tests must be made to ensure that it does indeed lie on each of the two arcs. (It is possible that point **I** lies on the great circles of the unit sphere which contain arcs **PA-PB** and **PC-PD**, but does not actually lie on the arcs).[5] This test is easily performed by determining whether or not the ray from **O** to point **I** intersects the chords **PA-PB** and **PC-PD**. If the ray intersects both chords, then point **I** is an arc-arc intersection point. If not then the arcs do not intersect. Figure 4 illustrates the case where the arcs intersect.

If P and Q are coplanar, the two edges project to arcs of the same great circle of the unit sphere. In this case, it is easy to determine whether the two arcs abut, overlap, or are disjoint

.The remainder of the modified Weiler algorithm involves clipping each face of PM1 against each face of PM2. This process

3. Vector quantities are indicated by a **bold** letter.

4. Choosing t = 1.0 ensures that the point found lies in the same hemisphere as **A** and **B**.

5. A great circle is the circle formed by intersecting a plane through the origin of a sphere with the sphere.

involves constructing contours for the pieces of each face of PM1 which lie inside each face of PM2. Once the projected edges (i.e. the arcs) have been divided into segments as above, this construction is exactly like the contour construction phase of the Weiler polygon clipping algorithm. For details of this construction, refer to [Weil80] or [Fole90].

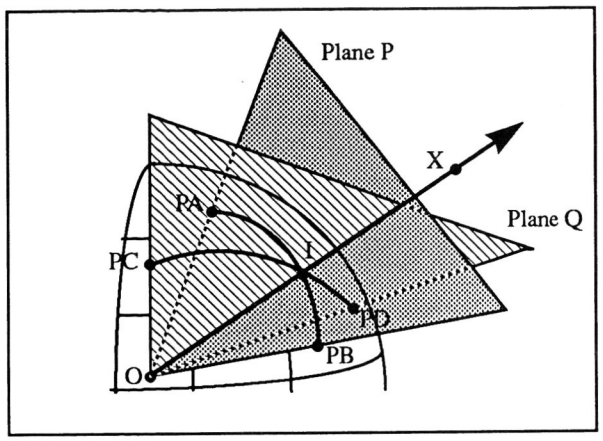

Figure 4 - Arc-Arc Intersection on the Unit Sphere

5.3 Determining the Locations of Added Vertices

The remaining step in the correspondence algorithm is to determine where the vertices added to each object's topology map onto the face of the original models. For star-shaped polyhedra, this can be done by casting rays from the center point to each vertex. The intersection of these rays with the faces of the original models determine the location of the added vertices, as shown in Figure 5. For general concave polyhedra, this step will depend on the method used for the initial projection onto the unit sphere.

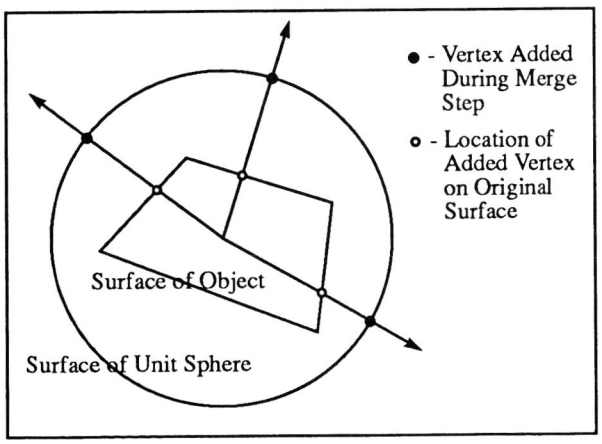

Figure 5 - Determining the Location of Added Vertices by Ray Casting

6.0 Results

Figures 6 to 9 show sequences of still frames from animations of the shape transformations between various pairs of objects. In each example, pay particular attention to the lack of distortion in the in-between objects. Figure 6 illustrates a cube changing into a soccer ball. Notice how smoothly the soccer ball emerges from the surface of the cube. Figure 7 shows a soccer ball changing into a flat, 5-pointed star. This demonstrates the capability to transform between convex and concave star-shaped polyhedra. Figure 8 shows a pair of concave star-shaped objects with very dissimilar shapes transforming from one to the other. Figure 9 shows the cube to soccer ball transformation again, but with the center of the cube selected to be near one of its vertices. Notice how the transformation is very different from that in Figure 6, but still quite smooth. The ability to alter the center point gives the designer/animator a tool to tune the exact transformation obtained.

7.0 Future Research

Further research needs to be done to develop projections for general concave polyhedra and for polyhedra with holes. For general concave polyhedra, a projection is needed which pushes out the concavities of the objects while maintaining certain geometric properties of the objects, such as ratios of edge lengths. An algorithm which recursively reduces the number of concavities of an object is currently being investigated. For objects with holes, the desired projection of the objects and subsequent polygon clipping could be carried out on the surface of a representative surface manifold with the same number of holes (e.g. a torus for an object with one hole).

Other open issues include providing the designer/animator the capability of specifying that certain regions on one object be mapped to specific regions on the other object, investigating the effects obtained by projecting the objects onto and clipping on the surface of an object other than the unit sphere, such as a cube or an ellipsoid, and providing the capability to specify that only the points in certain regions of the objects be transformed. This last capability would allow the generation of new objects which combine features of each original model.

8.0 Conclusions

Most previous approaches to the 3-D shape transformation problem either ignored key topological information ([Hong88] and [Terz89]) resulting in models whose faces fly apart during the transformation, or key geometric information ([Pare90]), resulting in uneven, distorted transformations (i.e. large portions of one object mapping to small portions of the other). [Chen89] employs both topological and geometric information from the two models, but lacks generality, due to necessity of creating contour slices of the models.

The approach in this paper employs geometric information in the projection step and topological information in the clipping step to establish correspondences between the vertices, edges,

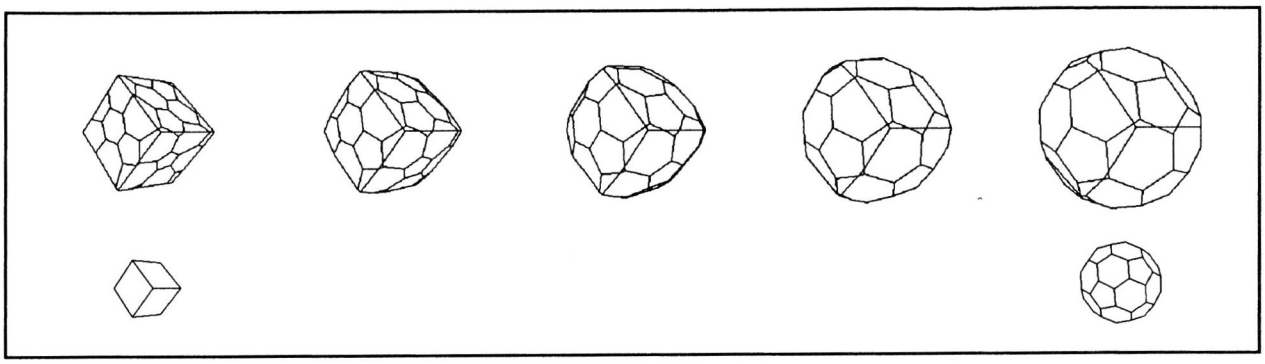

Figure 6 - Cube to Soccer Ball Transformation

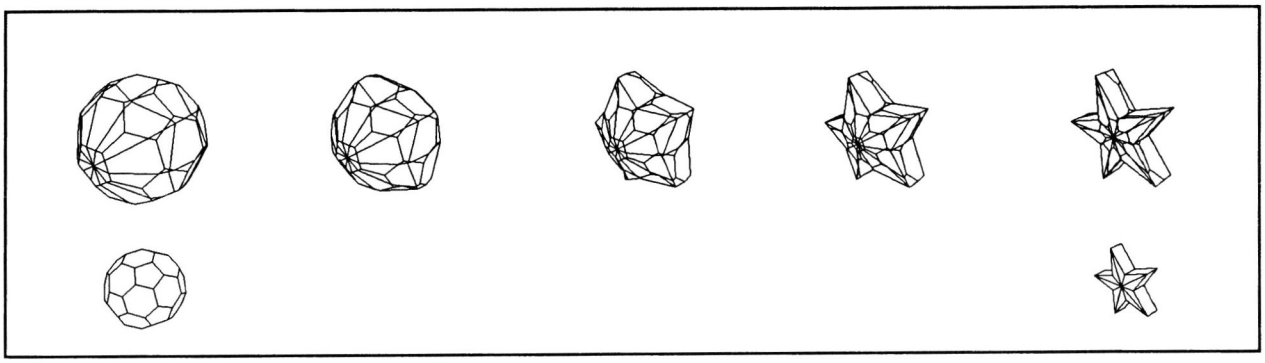

Figure 7 - Soccer Ball to Star Transformation

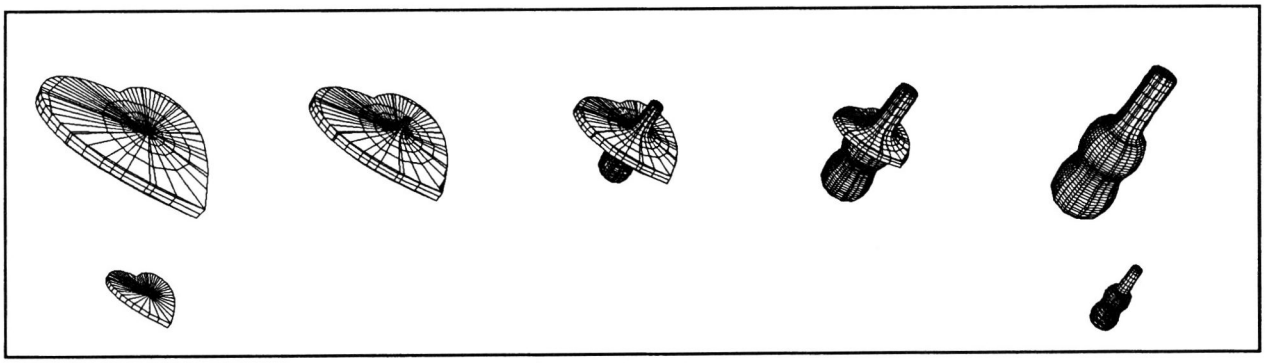

Figure 8 - Transformation between two complex objects

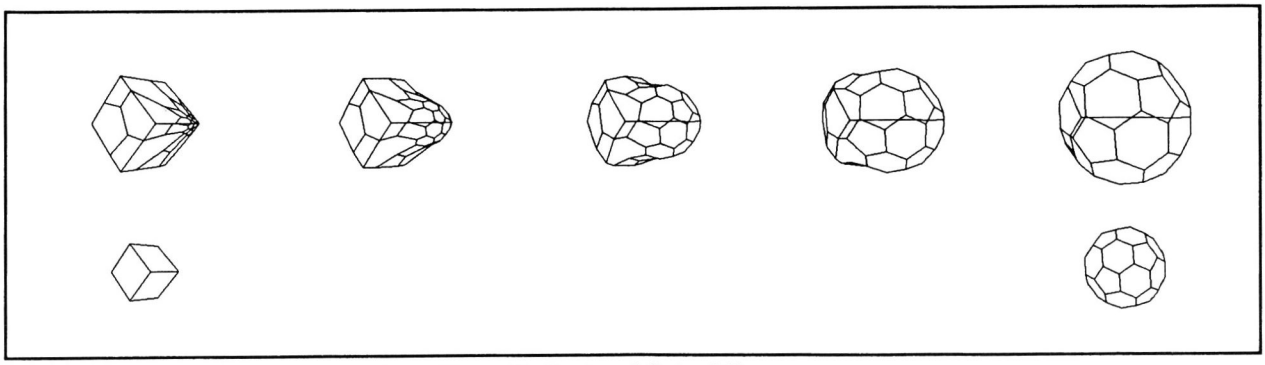

Figure 9 - Cube to Soccer Ball Transformation with Center of Cube Offset

and faces of two polyhedral models. Since both types of information are used, the intermediate objects obtained maintain their integrity (in an Euler sense), and have an appealing, intuitive appearance.

Summarizing, any shape transformation algorithm whose in-between shapes do not maintain face connectivity is of little practical use. [Hong88] and [Terz89] fall into this category. Of the other two algorithms, [Chen89], [Pare91], and the one described in this paper, only [Pare91] works for arbitrary models. [Chen89] has little hope of being extended due to the fact that slices through non-convex objects may result in multiple rings of edges. While the current implementation of our algorithm is limited to star-shaped polyhedra (of which convex polyhedra are a subset), the general approach of projecting both objects onto a unit sphere and merging the topologies is not. Wider classes of objects can be handled by determining suitable projections of the objects to the unit sphere.

The advantage of our method over [Pare91] is that the in-between objects generated by our algorithm are much less distorted than those generated by [Pare91]. This is due to the fact that [Pare91] largely ignores the geometric information contained in the original models which causes small regions of one model to map to large regions of the other.

Acknowledgments

We wish to thank the Department of Computer and Information Science and the Advanced Center for Computer Art and Design for the use of their facilities, Hewlett-Packard and AT & T for equipment grants which make this research possible, and Ken Supowit and Kevin Rodgers for useful ideas and criticism.

Bibliography

[Barr84] Barr, A., "Global and Local Deformations of Solid Primitives", *Computer Graphics,* Vol. 18., No. 3, July 1984, pp. 21-30.

[Burt76] Burtnyk, N. and M. Wein, "Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation", *Communications of the ACM,* Vol. 19, NO. 10, October,1976, pp. 564-569.

[Chen89] Chen, E., and R. Parent, "Shape Averaging and Its Applications to Industrial Design", *IEEE Computer Graphics and Applications,* Vol. 9, No. 11, January 1989, pp. 47-54.

[Coqu90] Coquillart, S., "Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modelling", *Computer Graphics,* Vol. 24, No. 4, August, 1990, pp. 187-196.

[Fole90] Foley, J., A. van Dam, S. Feiner and J. Hughes, *Computer Graphics - Principles and Practice,* Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1990.

[Haum88] Haumann, D. and R. Parent, "The Behavioral Test-Bed: Obtaining Complex Behavior from Simple Rules", *Visual Computer,* Vol. 4, No. 6, December 1988, pp. 332-347.

[Hong88] Hong, T., N. Magnenat-Thalmann and D. Thalmann, "A General Algorithm for 3-D Shape Interpolation in a Facet-Based Representation", *Proceedings of Graphics Interface '88,* June 1988, pp. 229-235.

[Pare77] Parent, R., "A System for Sculpting 3-D Data", *Computer Graphics,* Vol. 11, No. 2, Summer 1977, pp. 138-147.

[Pare91] Parent, R., "Shape Transformation by Boundary Representation Interpolation: A Recursive Approach to Establishing Face Correspondences", Technical Report OSU-CISRC-2/91-TR7, Computer and Information Science Research Center, The Ohio State University, 1991.

[Plat88] Platt, J. and A. Barr, "Constraint Methods for Flexible Models", *Computer Graphics,* Vol. 22, No. 4, August 1988, pp. 279-288.

[Reev81] Reeves, W., "Inbetweening for Computer Animation Utilizing Moving Point Constraints", *Computer Graphics,* Vol. 15, No. 3, August 1981, pp. 263-269.

[Sede86] Sederberg, T. and S. Perry, "Free Form Deformations of Solid Geometric Models', *Computer Graphics,* Vol. 20, No. 4, August 1986, pp. 151-160.

[Terz87] Terzopoulos, D., J. Platt, A. Barr and K. Fleischer, "Elastically Deformable Models", *Computer Graphics,* Vol. 21, No. 4, July 1987, pp. 205-214.

[Terz88] Terzopoulos, D. and K. Fleischer, "Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture", *Computer Graphics,* Vol. 22, No. 4, August 1988, pp. 269-278.

[Terz89] Terzides, C., "Transformational Design", *Knowledge Aided Architectural Problem Solving and Design,* NSF Project #DMC-8609893, Final Report, June 1989.

[Weil86] Weil, J., "The Synthesis of Cloth Objects for Computer Graphics", *Computer Graphics,* Vol. 20, No.4, August 1986, pp. 49-54.

[Weil77] Weiler, K. and P. Atherton, "Hidden Surface Removal Using Polygon Area Sorting", *Computer Graphics,* Vol. 11, No. 2, Summer 1977, pp. 214-222.

[Weil80] Weiler, K., "Polygon Comparison Using a Graph Representation", *Computer Graphics,* Vol. 14, No. 3, August 1980, pp. 10-18.

[Weil84] Weiler, K., "Topology as a Framework for Solid Modeling", *Proceedings of Graphics Interface '84,* May, 1984.