Three-Dimensional Metamorphosis using Multiplanar Representation

Mahesh Ramasubramanian
Program of Computer Graphics
Cornell University
Ithaca NY 14853
mahesh@graphics.cornell.edu

Anurag Mittal
Department of Computer Science
Cornell University
Ithaca NY 14853
anurag@cs.cornell.edu

Abstract

We introduce a novel method for three-dimensional metamorphosis between two polyhedral objects with different topologies. The 3D objects are represented as multiple 2D images and well-understood 2D morphing techniques are used to morph between their 2D representations. The resulting 2D morphs are used to reconstruct the intermediate 3D objects. The user controls the metamorphosis by matching features during the 2D morphing stage. Our method is simpler and more efficient than existing 3D morphing techniques which handle general topologies since the interaction and morphing is essentially done in 2-space.

1. Introduction

Metamorphosis, generally called *morphing*, is the construction of a sequence of intermediate objects depicting the gradual transition from one object to another. 2D and 3D morphing find a number of applications in entertainment, computer animation, scientific visualization, and education. 2D morphing of images has been widely studied and many effective algorithms have been developed [9, 8]. However, in recent years, more research has been directed towards 3D morphing, which operates between objects in 3-space and results in more realistic transformations. The intermediate 3D morphs also benefit from being independent of the viewing and lighting parameters. Hence, we can create the morph sequence once, and then experiment with various camera angles and lighting conditions for optimal rendering.

Lazarus and Verroust [6] present an excellent survey of the various 3D approaches to date. They classify 3D morphing algorithms as volume-based approaches or boundary-based approaches depending on the object representation. Volume-based approaches are known to work on general topologies but need to handle enormous amounts of data and are very slow [3, 7]. Boundary-based approaches which deal with polyhedral objects are in general very restrictive about the type of topologies supported. In recent years, Gregory *et al.* [5] proposed a method to deal with objects of equivalent topologies and DeCarlo and Gallier [4] proposed a method to deal with objects of different topologies. These two methods suffered from heavy user interaction, difficult correspondence input, and complex algorithms.

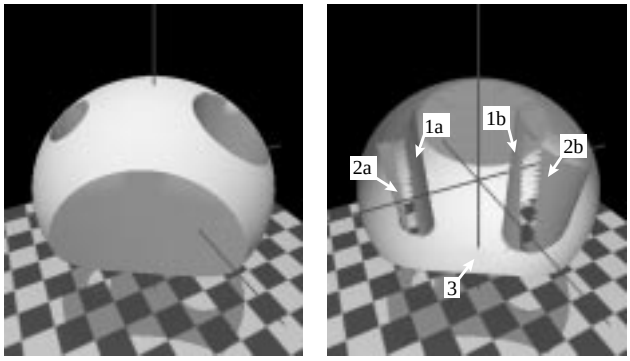

Figure 1. Unwrapping the 3D object about the vertical axis to transform it to its multiplanar representation. The object's cross-section is shown on the right with the individual surfaces slightly separated. The multiplanar representation of this object will be similar to Figure 3, with the surfaces 1a and 1b, 2a and 2b, and 3 forming the three planes.

The method we propose for polyhedral object morphing was motivated by two primary factors. The first and foremost criterion was to be able to morph between objects of both equivalent and different topologies. The second criterion was to provide a controlled, feature-based metamorphosis where the user could more easily identify and indicate correspondences. We extended Chen *et al.*'s [2] method of morphing 3D objects in a 2D parametric space to allow general topologies. An overview of our approach is presented in the next section.

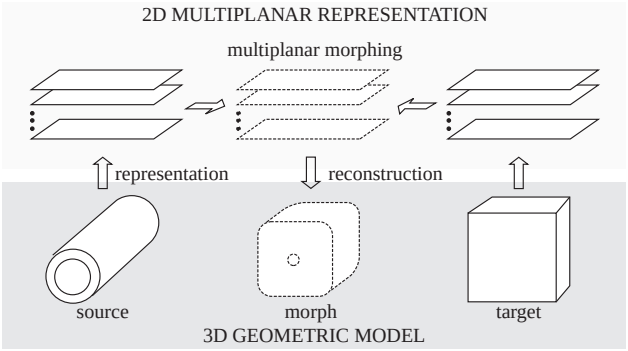

Figure 2. Basic Algorithm. The source and target 3D objects are transformed to their 2D multiplanar representation and these are morphed to generate intermediate multiplanes. The intermediate 3D objects are are reconstructed from these multiplanes. This way, the final morph sequence is in 3-space, even though the morphing takes place in 2-space.

2. Overview

Figure 2 shows an overview of the basic framework of our algorithm. Multiplanar representation of a general 3D object is central to this framework. This representation is achieved by transforming the 3D object to a 2D parametric space. The parametric space we have implemented is that of cylindrical coordinates (w.r.t. an axis) although any useful parametric space such as spherical (w.r.t. a point), or planar (w.r.t. a plane) can also be used. In cylindrical coordinates, we transform a 3D object into an image where θ and h determine the (x, y) pixel location on the planar representation and the "radius" for a particular θ and h is stored at that pixel location. The 2D image is thus a "radius" image and not an "intensity" image. Since there may be, in the general case, more than one surface encountered on a radial sweep outward from the selected morphing axis for the object, there may be more than one radius value for a particular pixel location. We therefore may have multiple planes rather than a single plane in our 2D representation. The distinct surfaces of the object are separated into different planes which are morphed individually to corresponding planes in the other object. Then, we use existing well-known techniques for morphing the 2D planar images, extending them to support multiple planes. The final step is to reconstruct 3D objects from the multiplanar representations obtained as a result of morphing. Object attributes including color, texture, and normal fields can be morphed by storing them alongside the radius values in the multiplanar representation and morphing them too. These stages are described in greater detail in the following sections.

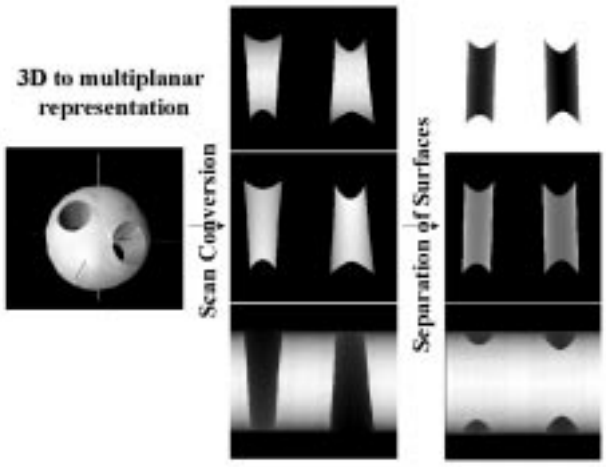

Figure 3. 3D to multiplanar representation. The cylindrical holes were aligned parallel to the vertical axis when computing the multiplanes, the object has been tilted here to show the holes more clearly. The object is first transformed to a multiple "radius" image and then the surfaces are separated to form the multiplane. After separation the top two planes hold the inner and outer surfaces of the cylindrical holes, while the lowest plane is the outermost surface of the object.

3. 3D to multiplanar representation

The 3D object is described by geometric primitives, which we assume to be a vertex/edge/face network. Every vertex on the surface of the 3D object can be described in cartesian coordinates (x, y, z) from which the it can be mapped to cylindrical, spherical or planar coordinates as desired.

3.1. Scan conversion

In scan conversion, we have to map the surfaces of the 3D object onto the "radius" image. To do this, the vertices of every face are mapped to the (θ, h, $radius$) coordinate system and then the faces are scan-converted. Depending on the position and orientation of the axis, and the complexity of the object, many points on the surface of the 3D object may get mapped to the same pixel in the radius image. This would be the case whenever a semi-infinite ray originating from the axis intersects the 3D object at more than one point. The next step is to separate these multiple radii into different planes so as to represent the different surfaces as different images. This will provide the multiplanar representation which is required for the morphing stage.

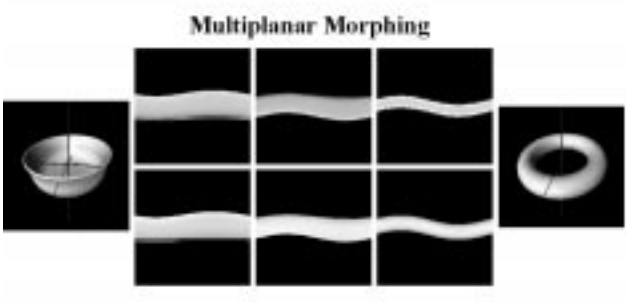

Figure 4. Multiplanar morphing. The left and right pair of images are the multiplanes of the bowl and torus objects w.r.t. the vertical axis. The top plane holds the inner surface of the objects and the bottom plane the outer surface. The corresponding planes from the two multiplanes are morphed to generate the intermediate multiplane in the middle.

3.2. Separation of surfaces

As shown in Figure 3, the multiplanar representation must be extracted from the radius image with multiple values by separating the different surfaces into different planes. To do this, we use modifications of a simple seed-fill algorithm. In the simple seed-fill algorithm, points on a surface are defined recursively as follows: (1) The initial seed is part of the surface. (2) Any point close to a point on the surface (both in their location and "radius" value) is also on the surface. This algorithm can be implemented using recursion, but a faster method is to use stacks to keep track of all the seed points yet to be tested. Once a surface has been separated, all these points are removed from the set of points yet to select and the algorithm is repeated till there are no points left.

4. Multiplanar morphing

After we have separated the different surfaces to obtain multiple 2D radius images ("multiplanes") of the objects, we select corresponding 2D images and morph them together using modified 2D morphing techniques. This will give us multiple 2D images for each particular stage in the morph, from which we will reconstruct the 3D structure of the object.The case when the number of planes is different in the two objects will be described in Section 6.

4.1. 2D morphing

We can use any good 2D morph technique to morph two 2D radius images from the two objects. In common 2D image morphing, image warping is coupled with color (in our case *radius*) interpolation. Image warping applies 2D geometric transformations to the images to retain geometric alignment between their features, while color interpolation blends their colors. There are however, constraints on the techniques which should be used since the images are *radius* images (example, the boundaries during dissolve should match to keep the object stitched together).

We have found the technique given by Beier and Neely [1] to be particularly well-suited to our morphs. Beier and Neely's algorithm uses line pairs to specify correspondences between the source and target images. Direct correspondence is provided for all points along the lines, while the mapping of points in the vicinity of the line is determined by their distance from the line. The displacement of any point in the source image is a weighted sum of its mappings due to each correspondence line pair, with the weights attributed to distance and line length.

4.2. Extending the Algorithm for Multiplanar morphing

For multiplanar morphing, we must first identify which of the multiple 2D radius images for one object is to be morphed to which radius image from the other object. Then, we have to specify the line segment correspondences for all these morphings. One constraint is that surfaces which are connected in the original 3D object, but have been separated in two planar images (because of the surface folding over itself as seen from the axis), should have boundary radius values which move together in the two images. This is required so that the reconstructed 3D object remains connected. This is accomplished by automatically generating line pairs along the boundaries in the two images, forcing them to move together.

5. Reconstruction

Intermediate 3D objects are reconstructed from the morphed multiplanes. The reconstruction algorithm takes advantage of the continuity of the surface of the 3D object to "stitch" together the multiplanes. In each of the planes, every pixel together with its neighbors forms a "face" of the 3D object. These pixels are transformed to the vertices of the 3D object by mapping them back to cartesian space. Pixels at the boundaries in a plane form a "face" with the closest adjacent pixels from among its neighboring planes. This is because all the surfaces in a 3D object are "closed". This condition of continuity "stitches" together the different planes and "closes" the 3D object, ensuring that there are no gaps on its surface. The robustness of this reconstruction depends on the morphs generated in the previous stage.

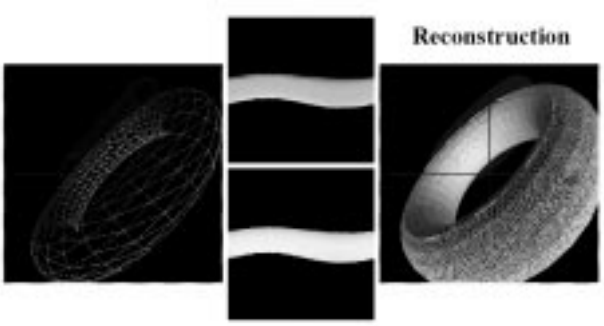

Figure 5. Reconstruction. A torus is reconstructed from its unmodified multiplanar representation. Adjacent pixels from the interiors of the individual planes form faces of the 3D object, and at the boundaries, pixels from adjacent planes form faces - this "stitches" together the various surfaces to form one continuous 3D object. Note the large increase in the vertex/edge/face network density after reconstruction.

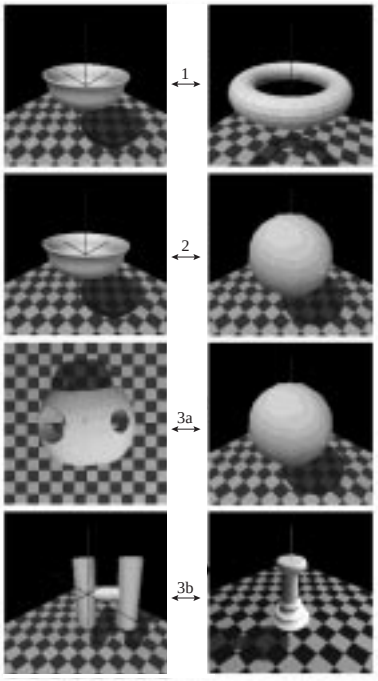

Figure 6. Different morphing scenarios. (1) The two multiplanes perfectly matching, (2) One unmatched plane, (3) A pair of unmatched planes in one of the multiplanes due to (a) the presence of an unmatched "hole" in one of the objects, or (b) the presence of an unmatched "extrusion". All other cases can be formed by combining the above.

Boundaries in the initial multiplanes must be morphed together so that at all intermediate morphs, boundaries on one plane can be connected to the boundaries on an adjacent plane. A reconstructed torus is shown in Figure 5.

6. Morphing scenarios

The previous sections describing the morphing process assume that there are corresponding planes between the two objects to be morphed. Matching planes from the two object multiplanes is not always possible, as in the case when the number of planes in the two object multiplanes is different. Three distinct cases may arise as illustrated in Figure 6 :

1. *Corresponding planes.* In the simplest case there are corresponding planes between two object multiplanes. The two planes are morphed to each other as described in the previous section. For example, in Figure 6.1, the outer surface of the saucer is morphed to the outer surface of the torus.

2. *A single plane having no correspondence.* In this case there is a plane present in one of the multiplanes with no corresponding plane in the other. This plane must be "adjacent" to the axis and is morphed either to a "0-radius" image (the axis) or to an adjacent plane. In Figure 6.2, the inner surface of the bowl morphs gradually to the outer surface or to the axis.

3. *Dual planes having no correspondence.* This arises when there is a "hole" or "extrusion" in one object without any corresponding feature in the other object. In this case, the two extra planes are morphed to an adjacent plane from the other object's multiplane. This has an effect of forming the "hole" or "extrusion" out of the adjacent surface. In Figure 6.3a, the holes collapse into the outer surface of the sphere and in Figure 6.3b, one of the arms of the "H" takes form out of the other arm.

7. User interaction

The user is required to interact with the system in two of the stages: in the stage linking the 3D object to a multiplanar representation and in the multiplanar morphing stage. In the first of the these stages, the user positions the axis about which the object is to be converted to its multiplanar representation. The position of this axis affects the number of planes generated and the shape of the surfaces in the planes. The user may have to experiment a little to minimize the number of planes generated and to position the

surfaces in the planes as desired. In the next stage, the user specifies correspondences between the two multiplanar representations by matching features on the planes using line pairs. This way the user controls the morphing by mapping certain features on one object to features on the other object. There are cases when some planes do not have corresponding planes (as described in Section 6) and the user can decide which adjacent planes to merge them with. This operation can be automated by randomly selecting one of the adjacent planes. Most of the user interaction is in 2-space so it is relatively easy for the user to map features.

8. Results

The following examples were rendered using faceted shading and neutral colors to better illustrate the shape of the intermediate objects. Animations described in this section are available at http://www.graphics.cornell.edu/∼mahesh/3dmorph.html.

Figure 7 shows multiplanar morphing where both the objects are represented with the same number of planes and there is simple correspondence between the planes. The multiplanes of the two objects are illustrated in Figure 4. Note that the saucer and the torus are of a different topological genus.

Figure 8 shows morphing from a weird-ball (a closed surface with two cylindrical fully penetrating holes) to a bust model of Beethoven. This example illustrates a difficult morphing case where the two objects are of a different topological genus, the number of planes in their multiplanar representation is different, and there is no easy correspondence between the planes. The multiplanar representation of the weird-ball is similar to Figure 3 and the Beethoven model has only a single plane in its radius image. The outer surface (the furthest plane) of weird-ball morphs to the outer surface (the single plane) of Beethoven; the outer and inner surfaces of the holes in weird-ball (the other two planes in Figure 3) morph to Beethoven's outer surface. This causes the two holes in the weird-ball to move towards and gradually merge into the shape of Beethoven's outer surface. The shadows in the figure show the changing shape of the holes during the morph sequence.

9. Conclusion

We have shown how 3D object morphing can be reduced to multiple 2D image morphing by using the multiplanar representation of the 3D objects. Our method inherits all the advantages of 3D morphing over 2D morphing while avoiding the complexity of morphing in 3D, since the morphing itself takes place in 2-space. There are no fundamental restrictions on the topology of the objects being morphed and it is easier for the user to feed in correspondence information among 2D planes.

This technique could be extended in a number of ways. The user could be given finer control over the generation of the multiplanar representation by allowing the controlling axis to curve. The system could be automated to position the axis such that minimum number of planes are created. In some cases it may be more intuitive for the user to match features in 3-space, and the system could be extended to automatically match the corresponding features in their multiplanar representations.

Acknowledgements

Many thanks to Jonathan Corson-Rikert for proofreading part of this paper. We gratefully acknowledge generous support of the Intel Corporation, on whose workstations the images in this work were computed.

References

[1] T. Beier and S. Neely. Feature-based image metamorphosis. In *Computer Graphics (ACM Siggraph '92 Conference Proceedings)*, volume 26(2), pages 37–42, July 1992.

[2] D. T. Chen, A. State, and D. Banks. Interactive shape metamorphosis. In *Proceedings of 1995 ACM Symposium on Interactive 3D Graphics*, pages 43–44, April 1995.

[3] D. Cohen-Or, D. Levin, and A. Solomovici. Three-dimensional distance field metamorphosis. *ACM Transactions on Graphics*, 17:116–141, 1998.

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

[5] A. Gregory, A. State, M. C. Lin, D. Manocha, and M. A. Livingston. Feature-based surface decomposition for correspondence and morphing between polyhedra. In *Computer Animation'98*, 1998.

[6] F. Lazarus and A. Verroust. Three-dimensional metamorphosis: a survey. *The Visual Computer*, 14(8/9):373–389, 1998.

[7] A. Lerios, C. D. Garfinkle, and M. Levoy. Feature based volume metamorphosis. In *Computer Graphics (ACM Siggraph '95 Conference Proceedings)*, pages 449–456, August 1995.

[8] G. Wolberg. *Digital Image Warping.* IEEE Computer Society Press, Los Alamitos, CA, 1990.

[9] G. Wolberg. Image morphing: a survey. *The Visual Computer*, 14(8/9):360–372, 1998.

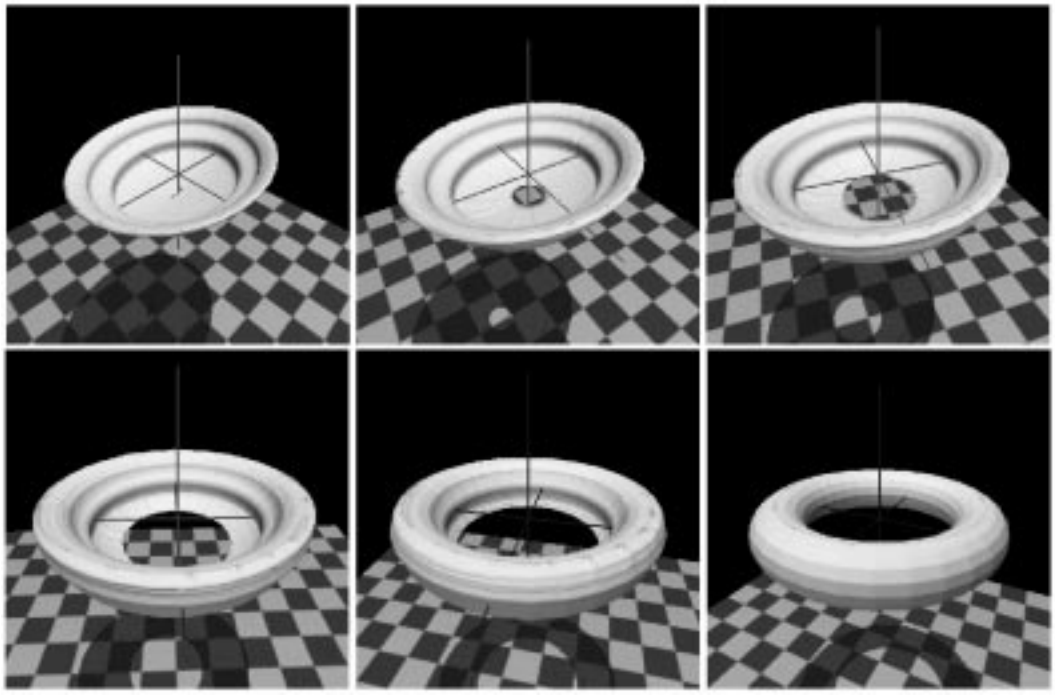

Figure 7. Saucer to torus morph sequence.

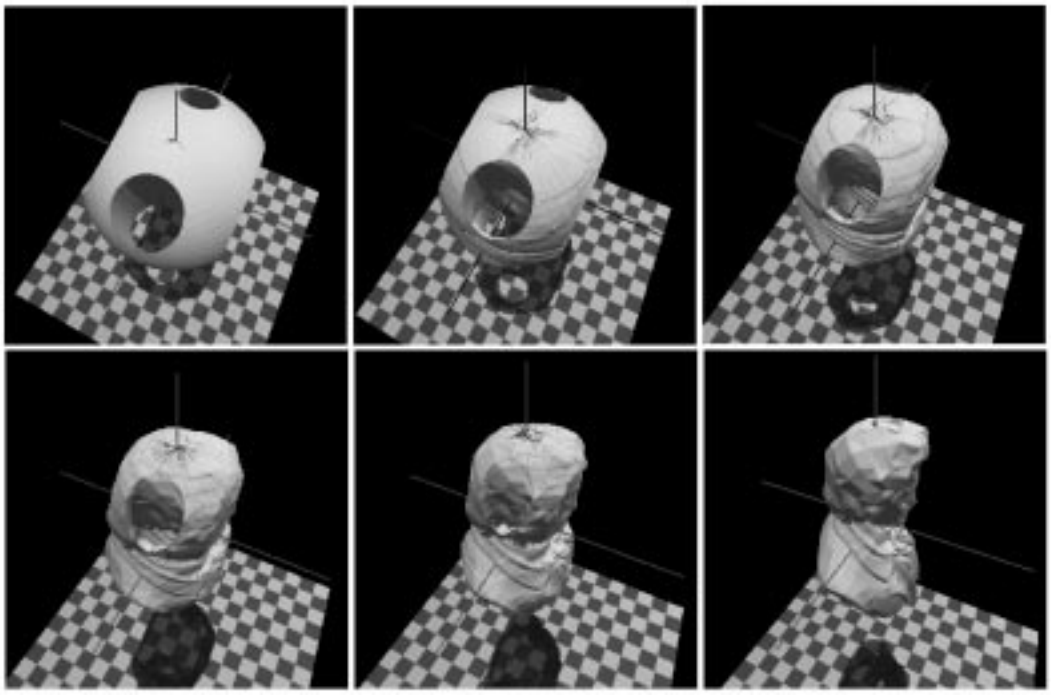

Figure 8. Weird-ball to Beethoven morph sequence. Note that the cylindrical hole gradually takes the shape of the outer surface of Beethoven; the shadow acts as a visual cue.