Feature-based Surface Decomposition for Polyhedral Morphing

Arthur D. Gregory
Andrei State, Ming C. Lin, Dinesh Manocha, Mark A. Livingston

Department of Computer Science
University of North Carolina at Chapel Hill
{gregory,andrei,lin,dm,livingst}@cs.unc.edu
http://www.cs.unc.edu/~geom/3Dmorphing

Abstract: We demonstrate a new approach for establishing correspondence for morphing between two homeomorphic polyhedral models. The user can specify corresponding feature pairs on the polyhedra with a simple and intuitive interface. Based on these features, our algorithm decomposes the boundary of each polyhedron into the same number of morphing patches. A 2D mapping for each morphing patch is computed in order to merge the topologies of the polyhedra one patch at a time. We create a morph by defining morphing trajectories between the feature pairs and by interpolating them across the merged polyhedron. The user interface provides high-level control as well as local refinement to improve the morph. The implementation has been applied to several polyhedra composed of thousands of polygons. We also demonstrate the performance of our system two non-simple polyhedra.

1 Introduction

Image and object morphing techniques have gained increasing importance in the last few years. Given two objects, metamorphosis involves producing a sequence of intermediate objects that gradually evolve from one object to another. The techniques have been used in a number of applications, including scientific visualization, education, entertainment, and computer animation. Morphing, whether in 2D or 3D, consists of two basic phases, establishing a correspondence between the images or objects and interpolating between them, in conjunction with blending their colors or textures.

We present a new approach for establishing correspondence for morphing between two homeomorphic polyhedra. Initially the user selects some corresponding elements called *feature pairs*. Our algorithm includes a simple and intuitive user interface for feature specification and automatically generates a *feature net*. Based on the feature nets, the algorithm decomposes the boundary of the polyhedra into morphing patches, computes a mapping for each morphing patch to a 2D polygon, merges them, and constructs a *merged polyhedron* whose topological connectivity contains both of the input polyhedra. In order to create a morph, the merged polyhedron has a *morphing trajectory* for each vertex to move along from one input polyhedron to the other. The overall complexity of the algorithm is O(K(m+n)), where K is a user-defined constant and m,n correspond to the number of vertices in the two input polyhedra.

SCG'99 Miami Beach Florida

2 Overview

Our algorithm decomposes the problem of morphing two polyhedra into morphing corresponding pairs of surface patches. Given the user's specification, the algorithm automatically partitions each polyhedron into a series of morphing patches, each of which is homeomorphic to a closed disk. We compute a mapping for each patch and eventually merge the topologies of the polyhedra. Moreover, we use algorithms for computing arrangements of lines, triangulations of polygons and planar straight-line graphs, and determining point locations in planar subdivisions. An overview of our approach is given in figure 1. Given the user input, the algorithm consists of two phases: establishing a correspondence between the two polyhedra and interpolating between corresponding vertex locations.

2.1 Correspondence

- **Feature Net Specification:** The user specifies a network of corresponding chains on the surfaces of the two input polyhedra by specifying the vertices of their endpoints. The interior edges of the chains are then computed as the shortest path between the specified endpoints. The feature net is a sub-graph of the vertex/edge connectivity graph of each polyhedron.
- **Decomposition into Morphing Patches:** Based on the feature nets, the algorithm decomposes the surface of each polyhedron into the same number of

morphing patches, each being homeomorphic to a closed disk.

- **Mapping:** A pair of corresponding morphing patches are mapped to a 2D polygon.
- **Merging:** The algorithm merges the topological connectivity of morphing patches in the 2D polygon as a PSLG.
- **Reconstruction:** Using the results from merging, the algorithm reconstructs the facets for the new morphing patch and generates a merged polyhedron with the combined topologies of the original two.
- **Local Refinement:** The user can make local changes to the feature net, such as splitting chains, moving extremal vertices, deleting chains or extremal vertices, or adding new ones, and then re-compute the merged polyhedron.

2.2 Interpolation

- **Trajectory Specification:** The user specifies the trajectories for the vertices of the feature net to follow during the morph. The morphing trajectories for the remaining vertices of the merged polyhedron are computed from these.
- **Morph Generation:** The algorithm makes use of the trajectories and interpolates the surface attributes to generate a morph.
- **Local Control:** The user can modify the trajectories and generate a new morph. This step does not involve re-computation of the merged polyhedron as shown in the shaded "feedback loop" of figure 1.

3 Implementation and Performance

We have implemented the algorithm in C++ using the OpenGL library and it runs on SGI workstations. It uses various data structures to represent the models and their topology, implements geometric and graph algorithms to compute the correspondences, and features a graphical user interface for specifying features and trajectories and for refining the morph.

The input polyhedra are specified in a shared vertex representation. The algorithm computes various data structures to store the geometric information as well as the adjacency graph for each polyhedron. Furthermore, the system ensures that each polyhedron has valid topology and that it satisfies the *Euler-Poincaré* formula. When the user specifies the extremal vertices of a chain in the feature net, the system computes the path connecting them using Dijkstra's *shortest path* algorithm. It starts with one of the extremal vertices as the start vertex and incrementally computes shortest paths to other vertices of the polyhedron. It stops when it has computed the path to the other extremal vertex. Since the endpoints of chains are typically close and the shortest path consists of a few edges, the system can compute these paths fast enough for interactive response.

Our implementation also utilizes a number of geometric algorithms for triangulating planar straight-line graphs, edge intersections and for point location. The system has been applied to a number of complex polyhedral models, as shown in the video. More details are given in [1].

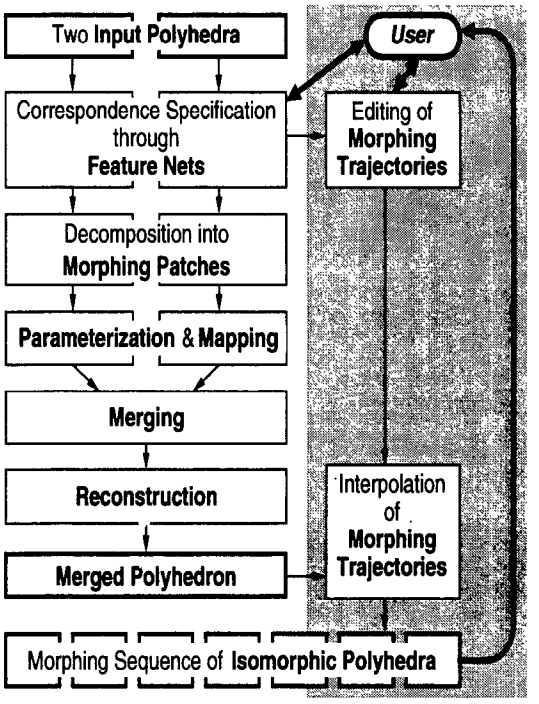

Figure 1. Overview of the polyhedral morphing algorithm.

4 Acknowledgement

This research has been supported in part by Sloan Fellowship, ARO Contract DAAH04-96-1-0257, NSF Career Award CCR-9625217, ONR Young Investigator Award (N00014-97-1-0631) and Intel.

5 References

[1] A. Gregory, A. State, M. Lin, D. Manocha and M. Livingston, ``Feature-based Surface Decomposition for Polyhedral Morphing", Proc. Of Computer Animation, 1998.