# Smooth Blending of Polygonal Slices with Different Topologies

Tatiana Surazhsky* Vitaly Surazhsky† Gill Barequet† Ayellet Tal‡

Morphing is a gradual transformation from one shape into another. Morphing has numerous practical applications in various areas, such as computer graphics, animation, scientific visualization, and modeling. Usually, to achieve spectacular, impressive and accurate results, the morphing process requires manual assistance. A major research challenge is thus to develop techniques that can automate this process as much as possible. Morphing has been investigated for various shapes, including polygons and polylines [4, 6, 8–11], and freeform curves [7].

Morphing requires the solutions to two problems. The first is to find a correspondence between the elements, or the features, of the two shapes. The second problem is to find the trajectories that the corresponding elements traverse during the morphing process. Most of the research done for solving the morphing-related trajectory problem has focused on the elimination of self-intersections and on the preservation of the geometrical properties of the intermediate shapes. To the best of our knowledge, no currently-available method guarantees that the resulting morph is self-intersection free.

Another interesting and challenging problem is the reconstruction of a solid object from a series of parallel planar cross-sections [1–3, 5]. It is important in areas such as medical imaging, topography, and solid modeling. A common approach for solving this problem is to subdivide it into subproblems of surface reconstruction between pairs of successive slices. The latter problem can be defined as follows: Given a pair of parallel planar slices, each consisting of a set of mutually non-intersecting, but possibly nested, simple polygons, find the boundary of a polyhedral solid object, whose cross-sections coincide with the input slices. Clearly, the problem has no unique solution. One of the common requirements is that the output surface should not have self-intersections, as expected from real-life objects.

In this work we present a morphing algorithm based on a reconstruction algorithm. We review the reconstruction algorithm which we use, explain the problems that occur while we adopting it for morphing, and discuss our solution.

A reconstructed surface can be viewed as the locus of a morphing process between two slices. Reconstructed surfaces can be guaranteed to not intersect themselves, and thus the corresponding morph is also free of self-intersections. Moreover, a reconstructed surface often handles non 0-genus polygons and different topologies. These are invaluable properties for morphing. The problem, however, is that reconstructed surfaces, when viewed as morphing sequences, produce abrupt changes in the topologies. While this is acceptable for reconstruction, it is not acceptable for morphing where smooth blending is a must.

Our method is based on the algorithm of [2]. This algorithm is simple and robust; it handles slices with multiple nested contours and does not rely on any resemblance between the input slices. Also, it does not introduce any intermediate slices or artificial bridges that may conflict

*Faculty of Applied Mathematics, The Technion—IIT, Haifa 32000, Israel. E-mail: tess@cs.technion.ac.il
†Faculty of Computer Science, The Technion—IIT, Haifa 32000, Israel.
E-mail: [vitus|barequet]@cs.technion.ac.il
‡Faculty of Electrical Engineering, The Technion—IIT, Haifa 32000, Israel.
E-mail: ayellet@ee.technion.ac.il

with the geometry of the adjacent slices. Moreover, the surface generated reveals the underlying topological structure of the solid object whose boundary is reconstructed.

The result is a three-dimensional triangulated surface whose vertices are those of the input slice contours. This surface blends slices with possibly different topologies. In this case, some of the triangles comprising the surface might be $xy$-parallel (lying on the plane that contains one of the input slices), or some edges which do not belong to the input contours might be added on the above plane.

We say that an edge is *bridging* if it lies on a slice plane but does not belong to any of the original input contours of this slice. A triangle is said to be *bridging* if one of its edges is bridging. Note that there are always bridging triangles in a triangulation which blends two slices with different topologies, when all the triangulation vertices must lie on the contours of the input slices. However, the presence of bridging triangles can be exploited to analyze the difference between slice topologies.

The drawback of a reconstructed surface which contains bridging triangles is the following. Consider the given contours of a single slice along with the edges of all the bridging triangles lying in the plane of this slice. The resulting topology is different from the initial topology of the slice. Thus, the blending between two slices defined by the reconstructed surface has an abrupt change of topology that occurs at that slice. This sudden change is very visible (and disturbing) during the morphing.

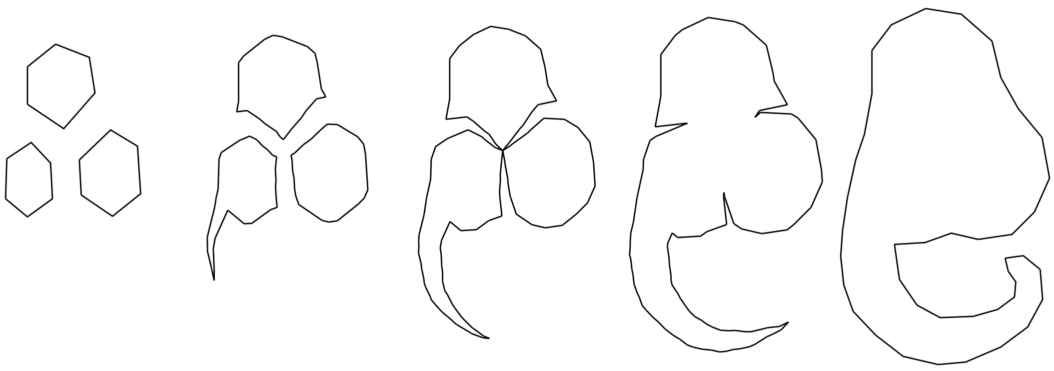
The aim of our work is thus to achieve a smooth change between the topologies of the slices without changing the initial topology of any of the input slices. This should be done while preserving the good properties of the original reconstruction algorithm. Clearly, this can be done only by introducing additional vertices.

We propose a simple and easy to implement algorithm to accomplish this. The input to our algorithm is the output of the algorithm of [2]. Each bridging triangle of the input surface is subdivided into a few triangles; some of its edges are subdivided in this process. A new vertex added in a subdivision of a bridging edge is placed between the slices planes.

In order to compute the locations, namely, heights, of the newly-added vertices we build a graph that contains the bridging triangles. Next, we construct the dual graph of each of the connected component of this graph. By analyzing the dual graphs we locate the new vertices satisfying various requirements for global consistency.

Figures 1, 2 and 3 show an example of surface reconstruction and morphing between two slices. The first (upper) slice contains three polygons; see Figure 1(a). The second (lower) slice contains one polygon (with a long "tail"); see Figure 1(e). The reconstruction algorithm of [2] produces the result shown in Figure 3(a). Our method improves this surface as shown in Figure 3(b). Note that there is no resemblance between this tail part of the second slice to any portion of the contours of the first slice. Thus, the tail was triangulated mostly with bridging triangles. The same holds for the area between the three contours of the first slice. Our algorithm resolves this situation by adding new vertices at intermediate heights between the slices (see Figure 3(b)). Figures 1(a–e) show a few snapshots of the morphing sequence (cross-sections of the improved interpolated surface) between the two slices

Let us refer to the course of our algorithm in detail. Figure 2(a) shows an $xy$-projection of all the triangles of the reconstructed surface. Some of them are bridging. Figure 2(b) shows the dual graph of the bridging triangles. The algorithm now adds vertices on the edges of the bridging triangles in the following manner: Every bridging triangle that corresponds to a leaf vertex of the dual graph is subdivided into two triangles by adding a new vertex on its single bridging edge, and by connecting it to the opposite edge by a new edge. The common vertex of the two new triangles is called a *leaf vertex* of the triangulation. Every triangle of the graph with two bridging edges is subdivided into four triangles by introducing three new vertices, one



(a) Upper (b) (c) (d) (e) Lower

Figure 1: The morphing sequence: The input (the left-most and the right-most) slices and the intermediate slices.

on each edge of the triangle, and connecting them with new edges. For every triangle of the graph with three bridging edges, we add four new vertices: one on every edge of the triangle, and one on its barycenter. These new vertices are connected with new edges so as to create six new triangles. The vertex added at the barycenter is called a *junction vertex* of the triangulation. The result of this refinement is shown in Figure 2(c).

The main question is where to place the newly-added vertices. If a vertex is on one of the slice contours the answer is simple: it is added at the midpoint of the edge.

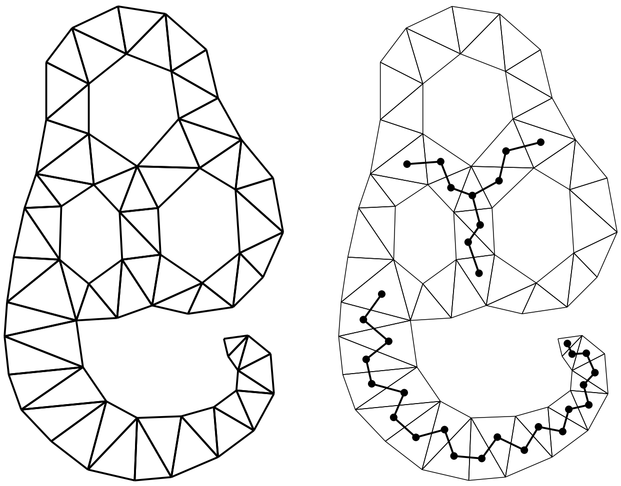
Let us consider the vertices that were added at the barycenters of the bridging triangles. This triangle corresponds to a junction vertex (of degree 3) of the dual graph. This is where contours of the two slices "meet" and cause a change of topology. See, for example, the dual-graph vertex between the three polygons of the first slice (Figure 2(b)). We assume that this event should occur at the midway (height-wise) between the two slices (see Figure 3(b)).

The new vertices of the latter type are introduced by the subdivision of bridging edges. Consider the vertex of the dual graph, that corresponds to the triangle containing a new vertex of this type. Each such vertex is located on a path between junctions and/or leaves of the dual graph. The coordinates of all the corresponding junction and leaf vertices of the triangulation are already known. Therefore we can compute the coordinate of the new vertex by interpolation between the coordinates of the vertices corresponding to the endpoints of the path. The example in Figure 2 exhibits four such paths. One path is in the tail-shaped part, gradually bringing the end of tail (in the lower slice) to the level of the three contours (in the upper slice). Three paths begin at the junction of the graph, found among the three contours of the upper slice, and end at the periphery of the graph (see Figure 2(b)). We used linear interpolation for ease of implementation. However, any smooth interpolation function will do, e.g., B-spline or Bézier functions of any order.

## References

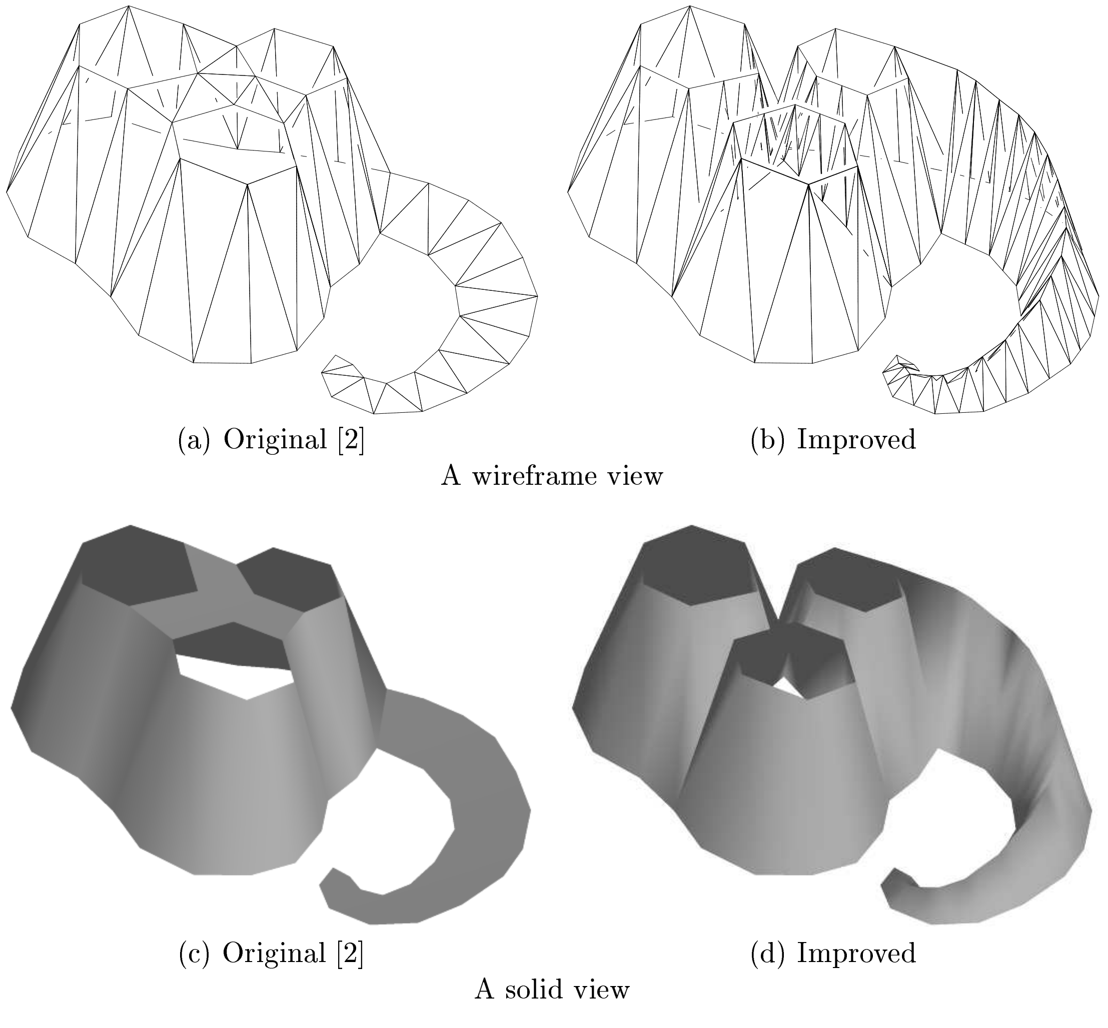

[1] G. Barequet, D. Shapiro, and A. Tal, *Multi-level sensitive reconstruction of polyhedral surfaces from parallel slices*, The Visual Computer, to appear.

[2] G. Barequet and M. Sharir, *Piecewise-linear interpolation between polygonal slices*, Computer Vision and Image Understanding, 63 (1996), pp. 251–272.



(a) Original [2] (b) Dual graph (c) Refined triangulation

Figure 2: The $xy$-projections demonstrating the course of the algorithm.



(a) Original [2] (b) Improved

A wireframe view

(c) Original [2] (d) Improved

A solid view

Figure 3: A 3-dimensional view of the reconstruction.

[3] J. Boissonnat, *Shape reconstruction from planar cross sections*, Computer Vision, Graphics and Image Processing, 44 (1988), pp. 1–29.

[4] E. Carmel and D. Cohen-Or, *Warp-guided object-space morphing*, The Visual Computer, 13 (1997), pp. 465–478.

[5] H. Fuchs, Z. Kedem, and S. Uselton, *Optimal surface reconstruction from planar contours*, Comm. of the ACM, 20 (1977), pp. 693–702.

[6] E. Goldstein and C. Gotsman, *Polygon morphing using a multiresolution representation*, Proceeding of Graphics Interface, (1995).

[7] T. Samoilov and G. Elber, *Self-intersection elimination in metamorphosis of two-dimensional curves*, The Visual Computer, 14 (1998), pp. 415–428.

[8] T. W. Sederberg, P. Gao, G. Wang, and H. Mu, *2D shape blending: an intrinsic solution to the vertex path problem*, Computer Graphics (SIGGRAPH '93), 27 (1993), pp. 15–18.

[9] T. W. Sederberg and E. Greenwood, *A physically based approach to 2D shape blending*, Computer Graphics (SIGGRAPH '92), 26 (1992), pp. 25–34.

[10] M. Shapira and A. Rappoport, *Shape blending using the star-skeleton representation*, IEEE Trans. on Computer Graphics and Application, 15 (1995), pp. 44–51.

[11] A. Tal and G. Elber, *Image morphing with feature preserving texture*, Graphics Forum, 18 (1999).