Blending Polygonal Shapes with Different Topologies [1]

Tatiana Surazhsky

Department of Applied Mathematics, The Technion—IIT, Haifa 32000, Israel.

Vitaly Surazhsky Gill Barequet

Department of Computer Science, The Technion—IIT, Haifa 32000, Israel.

Ayellet Tal

Department of Electrical Engineering, The Technion—IIT, Haifa 32000, Israel.

Abstract

In this paper we propose a new method for morphing between two polygonal, possibly non-simply-connected, shapes in the plane. The method is based on reconstructing an xy-monotone surface whose extreme cross-sections coincide with the given shapes. The surface generated by our algorithm does not contain any self-intersections, does not change the topologies of the input slices, does not contain any horizontal triangles, and guarantees that all the topology changes occur at a mid-height which is a degenerate form of both input topologies. All these properties are highly desirable for blending shapes of different topologies.

Key words: Metamorphosis, morphing, interpolation, blending, reconstruction.

Email addresses: tess@cs.technion.ac.il (Tatiana Surazhsky), vitus@cs.technion.ac.il (Vitaly Surazhsky), barequet@cs.technion.ac.il (Gill Barequet), ayellet@ee.technion.ac.il (Ayellet Tal).

URLs: http://www.math.technion.ac.il/~tess (Tatiana Surazhsky), http://www.cs.technion.ac.il/~vitus (Vitaly Surazhsky), http://www.cs.technion.ac.il/~barequet (Gill Barequet), http://www.ee.technion.ac.il/~ayellet (Ayellet Tal).

[1] This research was supported by the fund for the promotion of research at the Technion.

1 Introduction

Metamorphosis is the gradual transformation of one shape into another. It has numerous practical applications in various areas, such as computer animation, scientific visualization, and solid modeling. Usually, the morphing process requires manual assistance in order to achieve intuitive and accurate results. A major research challenge is thus to develop techniques that minimize the user assistance when it is not necessary. Morphing has been investigated for various shapes in two dimensions, including polygons, polylines [5,7,10–13], and freeform curves [9]. It has also been investigated for images (see [14] for a survey) and in three dimensions (see [8] for a survey).

Morphing requires the solutions to two subproblems. The first problem is to find a correspondence (matching) between features of the two shapes. The second problem is to find trajectories that the corresponding features traverse during the morphing process. These trajectories transform the initial shape into the final shape. 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. We are not aware of any currently-available method that guarantees that the morph is self-intersection free when the given polygons have genus greater that zero. In this paper we present such an algorithm.

A related intriguing problem that has been studied in the literature is the reconstruction of a solid object from a series of parallel planar cross-sections [1–4,6]. It has important applications in 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 is defined as follows:

> Given a pair of parallel planar slices, each consisting of a set of mutually nonintersecting, 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 does not have self-intersections, as expected from real-life objects.

A reconstructed surface can be viewed as a morphing sequence between two slices (by varying the height continuously). When viewed this way, the reconstructed surface can be used as the path of interpolation between the polygons in the two slices. Reconstructed surfaces can be guaranteed not to intersect themselves, and thus the corresponding morph is also free of self-intersections. Some surface-reconstruction algorithms (e.g., these of [1,3]) handle nonzero-genus slices with different topologies. These are invaluable properties for morphing. One could thus assume that reconstruction algorithms can be used for solving the metamorphosis problem. In this paper we show the problems that arise when using a reconstruction algorithm for morphing in a straightforward manner. We present a novel algorithm which overcomes the shortcomes by improving a reconstruction algorithm.

Our algorithm has several desirable properties. First, the topologies of the input slices are not altered. Second, the surface we build is $xy-$ monotone. Third, the surface, and thus the morphing sequence, is free of self–intersections for any topology and genus of the input slices. Fourth, the reconstructed surface does not contain any horizontal triangles. Finally, all the topology changes occur at the mid-height between the two slices and the latter slice has a degenerate form of both input topologies.

The rest of the paper is organized as follows. In Section 2 we discuss the background for this work. In Section 3 we define the problem discussed in this paper. In Section 4 we present our algorithm. In Section 5 we prove some properties of our algorithm and of the surface it generates. In Section 6 we provide results of our experimentation with the algorithm. We conclude in Section 7.

2 Background

In this section we review the reconstruction problem and the algorithm upon which we base our solution.

2.1 The reconstruction problem

The input to the reconstruction problem consists of two "slices," where each slice is a planar shape composed of several nonintersecting, but possibly nested, simple polygons. The two slices are assumed to lie within parallel planes in a 3-dimensional space. The goal is to find the boundary of a polyhedral solid object, whose cross-sections coincide with the input slices.

The input to the morphing problem is identical, with the minor difference that the two polygonal shapes are assumed to be contained in the same plane. A straightforward solution to the morphing problem by reconstruction can be done by positioning the two shapes in separate planes, solving the reconstruction problem, obtaining the solution to the morphing problem by "slicing" the reconstructed surface (that is, by intersecting it with xy-parallel planes), and projecting the slices onto the xy-plane.

Definition 1 xy-monotonicity: *A surface is xy-*monotone *if no vertical line intersects it in more than one point.*

We consider a solution to the reconstruction problem which generates an xy-monotone piecewise-linear surface consisting of triangles only. The two horizontal cross-sections of the surface at the heights of the input slices coincide with the two slices.

Definition 2 Bridging edge: *An edge of the interpolated surface is* bridging *if it lies within an input slice plane but it does not belong to any of the input polygons of* *the slice.*

Definition 3 Bridging triangle: *A triangle of the interpolated surface is* bridging *if at least one of its edges is bridging.*

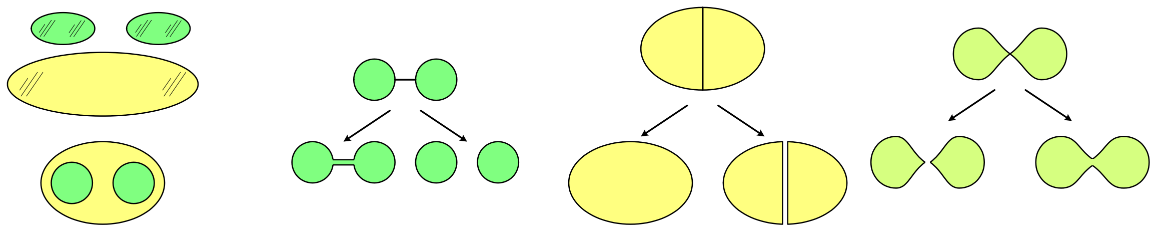
2.2 A Brief Overview of the Reconstruction Algorithm

Our algorithm is based on the reconstruction algorithm of Barequet and Sharir [3]. This algorithm handles slices with multiple nested polygons and does not rely on any resemblance between the input slices. In addition, it does not introduce any intermediate slices or artificial bridges that may conflict with the geometry of the adjacent slices. Moreover, the generated surface reveals the underlying topological structure of the solid object whose boundary is reconstructed.

The algorithm first matches similar portions of polygons of the two slices. Then it tiles each pair of matched portions by a sequence of adjacent triangles. The remaining (unmatched) portions of slice polygons form a collection of closed spatial polygons (holes), each of which may be composed of pieces of polygons of both slices and of some edges of the connecting triangles. The matching procedure guarantees that the xy-projections of these holes are pairwise disjoint, although they may be nested within each other. If no nesting occurs, the algorithm simply triangulates each of these spatial polygons, using a dynamic-programming procedure which minimizes the total area of the triangulation. If nesting occurs, the algorithm takes one polygon P with all polygons $P_1, \ldots, P_k$ whose xy-projections are directly nested within that of P, and applies a minimum spanning-tree procedure that introduces k edges connecting these polygons and yielding an xy-projection which is simply connected. The algorithm then triangulates the simplified polygon as above.

The result is a three-dimensional xy-monotone triangulated surface whose vertices are those of the input slice contours. This surface blends slices with possibly different topologies. Some of the triangles comprising the surface may be xy-parallel (lying within a plane that contains one of the input slices). Some triangulation edges which do not belong to the input contours may also be contained in one of the input planes.

The latter edges, although seem to be the correct choice for the reconstruction, cause abrupt changes in the topology when the surface is used for morphing. This is so because these edges imply that the topology changes at the beginning or at the end of the morph sequence, rather than evolving gradually. This sudden change is very visible and disturbing. In this paper we propose a solution for this problem.



(a) Input (b) In 2 contours (c) In 1 contour (d) Intermediate

Fig. 1. Degeneracies in blending slices with different topologies

3 Statement of the Problem

Consider the interpolation between two slices with different topologies. Since a topology is a "discrete" notion and cannot change gradually and continuously, there is always at least one intermediate level where the topology changes abruptly. The primary problem in constructing a blending sequence between slices with different topologies is to choose the intermediate location where this happens. Consider, for example, the example shown in Figure 1(a), in which the upper slice contains two contours (shown in green), whereas the lower slice contains one contour (shown in yellow). The change of topology can occur at the level of the upper slice Figure 1(b)), the lower slice Figure 1(c)), or in between (Figure 1(d)).

The interpolation algorithm of [3] has the property that it does not add vertices in intermediate heights. Therefore changes of topology in the reconstructed surface occur only at the levels of the input slices. As a result, this algorithm (or any other algorithm with this property) always produces bridging triangles in the interpolation between two slices with different topologies. The drawback of a surface reconstructed with bridging triangles is that the topology of its xy cross-section at the height of a slice is different from that of the original slice. Thus, an abrupt change of topology occurs at the very beginning or at the very end of the morphing sequence.

The goal of this work is to achieve a smooth transformation between the topologies of the slices without changing the topologies of the input slices. The topology should change in the midst of the intermediate (interpolated) sequence of slices. Consequently, the resulting morph sequence will appear much smoother. Obviously, if the topology changes in intermediate levels, additional vertices must be introduced at these levels.

4 The Algorithm

Given two polygonal slices, we first run the reconstruction algorithm described above. (Or, for this purpose, any interpolation algorithm whose outcome is an xy-monotone surface.) The input to our algorithm consists then of the two original slices and of the reconstructed surface that interpolates between them. We would like to emphasize that xy-monotonicity is a property of the reconstruction algorithm we use and not of the input slices whose xy-projections are general and can intersect. The only vertices of this surface are the original vertices of the input slices. As mentioned above, when the topologies of the two slices are different, the surface contains bridging triangles. Our algorithm modifies the surface so as to eliminate all the bridging triangles, thus achieving a smooth morph between the input slices.

We describe our algorithm by illustrating its behavior on the example shown in Figures 2–4. The first (upper) slice contains three polygons; see Figure 2(a). The second (lower) slice contains one polygon with a long "tail"; see Figure 2(e). Figure 3(a) shows an xy-projection of the triangles of the reconstructed surface. Some of them are bridging. Our algorithm proceeds in four steps:

(1) Find the connected regions of the given triangulated surface graph G, consisting solely of bridging triangles.
(2) Compute the dual graph G^* of the union of the above regions (Figure 3(b)).
(3) For each region, subdivide the triangles so as to eliminate the bridging edges (Figure 3(c)). This is done by adding new vertices, with the aid of the dual graphs, as explained below.
(4) Assign new z coordinates to the newly-added vertices, as described below.

Steps 1 and 2 are straightforward and are performed using well-known techniques. We now elaborate on Steps 3 and 4.

4.1 Triangle Subdivision

In Step 3 we subdivide the bridging triangles. First, the bridging triangles are classified according to the number of their bridging edges. Since every bridging edge (in the reconstructed surface) corresponds to an edge in G^*, this number is actually the number of neighboring triangles in the connected regions.

- **One bridging edge:** A bridging triangle that contains one bridging edge corresponds to a leaf vertex in G^*. The vertex opposite to the respective bridging edge in G is called a *leaf vertex* of the triangulation. The triangle is subdivided into two triangles by splitting its bridging edge and by connecting the newly-added vertex to the leaf vertex. See Figure 5(a).
- **Two bridging edges:** A bridging triangle with two bridging edges is subdivided into four triangles by splitting its three edges. This step adds three vertices and three edges. See Figure 5(b).
- **Three bridging edges:** A bridging triangle with three bridging edges is subdivided by adding four new vertices. Three of the new vertices are located on the triangle edges, and the additional vertex (called a *junction* vertex) is located inside the

triangle. The latter vertex is connected with new edges to the six vertices that lie on the original triangle. See Figure 5(c). The x and y coordinates of the junction vertex are set to the barycentre of the xy-projection of the triangle.

4.2 Assigning z-coordinates

Finally, in Step 4, we assign heights (z coordinates) to the new vertices created in Step 3. We do that by analyzing the graph G^*, as described below. This step allows us to accomplish smoothness and global consistency.

The graph G^* contains vertices whose maximum degree is three. Consider first the degree-3 vertices: the junction vertices. A vertex of this type corresponds to a triangle in G with three bridging edges. Such triangles realize the "pants effect," where different contours of one slice are merged into a single contour in the other slice, thereby causing a change of topology. The location of this change controls the behavior of the entire interpolation. Since the exact height of this effect is unknown, we assume that it occurs at the midway (height-wise) between the two slices. The algorithm allows the user to set other heights for junction vertices, possibly according to the local neighborhood. For example, the larger a bridging triangle with three bridging edges is, the further (z-wise) from it the corresponding junction vertex should be located.

By removing the junction vertices from G^*, the graph is split into path subgraphs with only degree-2 and degree-1 vertices. These vertices correspond to bridging edges in G. The endpoints of the path subgraphs are either junction vertices or leaf vertices. The heights of the junction vertices have already been set as was explained above. The height of each leaf vertex is set to the height of the slice that contains the respective bridging edge.

It remains to compute the heights of the internal (degree-2) vertices of each path by interpolating the heights of the path endpoints. We experimented with B-spline and Bèzier interpolation functions, but found that linear interpolation is satisfactory. (Again, the choice of the interpolation function is left to the user or the application.) We observed, however, one case that needed a special treatment, namely, when the heights of the two endpoint vertices of a path were set to the same slice height (that is, they were leaf vertices at the same slice). Setting the height of the entire path to the height of the slice does not produce a plausible result since, again, it yields a topology change at that level. Instead, we split such a path into two subpaths, set the height of the splitting vertex to the average height of the two slices, and interpolate the heights of the internal vertices of the sub-paths separately.

The example shown in Figure 3 contains four paths. One path forms the tail-shaped part, gradually blending the end of the tail (in the lower slice) with the three contours (in the upper slice). Three paths emanate from a junction vertex in the graph, found among the three contours of the upper slice, and terminate at the periphery of the graph (see Figure 3(b)).

Figure 4 illustrates the goodness of our algorithm on the example of Figure 3. Figures 4(a–b) show the surface constructed by the algorithm of [3]. Figures 4(c–d) show the improved surface generated by our algorithm. Note that there is no resemblance between the tail part of the second slice to any portion of the contours of the first slice. Thus, in [3] the tail was triangulated mostly with bridging triangles. The same holds for the area between the three contours of the first slice.

4.3 Complexity Analysis

We analyze the complexity of the algorithm as a function of n, where n is the number of triangles in G, the surface generated by the preprocessing reconstruction algorithm. Obviously, since G is planar and by Euler's formula, the number of vertices and edges in G are $O(n)$. Step 1 of the algorithm can be carried out by a simple DFS search in $O(n)$ time. Step 2 also requires only $O(n)$ time. Each of the triangle subdivision cases involves constant work, for a total of $O(n)$ time required for the entire step. By the end of this step the complexity of G is still $O(n)$. Therefore Step 4 also requires $O(n)$ time for interpolating the final heights of the new vertices.

5 Properties of the Algorithm

In this section we show several properties of our algorithm.

(1) The algorithm does not leave any bridging triangles in the reconstructed surface, therefore, the topologies of the input slices are not altered. This property is obvious from the definition of the algorithm.
(2) The algorithm preserves the xy-monotonicity of the surface. Thus, the new surface created by our algorithm is self-intersection free. This is so since the algorithm changes only the z coordinates of selected points on the surface and interpolates between them linearly (using a triangular mesh).
(3) The algorithm eliminates all horizontal triangles in the surface. We will prove this property in Theorem 2 below.
(4) The algorithm ensures that all topology changes (of horizontal cross-sections of the surface) occur only at the mid-height between the two interpolated slices. Moreover, this intermediate slice either has the same topology of the two slices, or has a degenerate form of one or both of them. We will prove this property in Theorem 1 and Corollary 3 below.

For simplicity of exposition we refer to the lower and upper slices as if their vertical heights are 0 and 1, respectively. We denote by $h(\cdot)$ the height of a geometric entity.

Theorem 1 *Every cross-section of the reconstructed surface at* $0 < h < \frac{1}{2}$ *(resp.,* $\frac{1}{2} < h < 1$*) has the same topology as the lower (resp., upper) slice.*

Proof: Since the reconstructed surface (together with the bounding slices) is a 2-manifold without self-intersections, the topology of its horizontal cross-sections changes only in local (vertical) extrema of the surface. Due to the definition of the algorithm, such extrema may occur only in new vertices added by the algorithm. Such a vertex either lies on a bridging edge that belongs to two adjacent bridging triangles, or it is the junction vertex of a triangle with three bridging edges. In the latter case, the vertex is at height of 1/2 by the definition of the algorithm.

Let us then consider the different cases of two adjacent bridging triangles denoted as $i - j$ (for $1 \leq i, j \leq 3$), where i and j are the numbers of bridging edges in the triangles. Assume without loss of generality that the bridging edge shared by the two bridging triangles and containing the newly-added vertex v is at height $h = 0$ (that is, in the lower slice). Note that horizontal cross-sections of the surface include collections of closed contours. The topology can change only in cross-sections where contours touch, or where a contour shrinks into a point or into a polygonal chain (possibly one line segment). Our goal is to prove that this happens only at height $h = 1/2$. We restrict our attention to cross-sections that pass through vertices added by our algorithm, since a change of topology cannot happen in other cross-sections because the original surface is a 2-manifold.

- 1-1 (two leaf triangles): The algorithm always locates this vertex at height 1/2.
- 1-2: Refer to Figure 6. Here the two bridging triangles are $\triangle_{abc}$ (with the bridging edge $\overline{bc}$) and $\triangle_{cbg}$ (with the bridging edges $\overline{cg}$ and $\overline{cb}$). In order to show that the cross-section does not contain a degenerate topology, we need to show that the horizontal plane $\mathcal{P}$ that passes through the vertex v intersects portions of only one of the original bridging triangles (the result of subdividing the triangles). By our assumption $h(\overline{cb}) = 0$. The second bridging edge $\overline{cg}$ is at the same height, since all the bridging edges are parallel to the planes of the slices, and the edges $\overline{cb}$ and $\overline{cg}$ have in common the vertex c. Thus, $h(b) = h(g) = 0$, and the non-bridging edge $\overline{bg}$ is a contour edge. Therefore $h(f) = 0$ as well, since this vertex lies on a slice contour.

  The vertex a can be either of height 0 or of height 1:

  (1) If $h(a) = 0$, then $0 < h(v) < h(e)$ by the definition of the algorithm; see Section 4.2.

  Consider the horizontal plane $\mathcal{P}$ that passes through the vertex v at $h(v)$. The interiors of all the triangles obtained by subdividing the bridging triangle $\triangle_{abc}$ are below $\mathcal{P}$: $0 = h(a) < h(v)$, $0 = h(b) < h(v)$, and $0 = h(c) < h(v)$. Therefore $\mathcal{P} \cap \{\triangle_{acv} \cup \triangle_{abv}\} = v$, where we use the fact that a function defined on a triangle is linear and hence monotone in any direction. A similar reasoning shows that $\mathcal{P} \cap \triangle_{bfv} = v$.

The only triangles intersected by $\mathcal{P}$ are $\triangle_{cev}$, $\triangle_{efv}$, and $\triangle_{gef}$. We are not interested in $\triangle_{gef}$ since it is not adjacent to v. For the other triangles we have

$$0 = h(c) < h(v) < h(e) \Longrightarrow \mathcal{P} \cap \mathrm{Int}(\triangle_{cev}) \neq \emptyset;$$
$$0 = h(f) < h(v) < h(e) \Longrightarrow \mathcal{P} \cap \mathrm{Int}(\triangle_{efv}) \neq \emptyset,$$

where $\mathrm{Int}(\triangle)$ denotes the interior of the triangle $\triangle$. The triangles $\triangle_{cev}$ and $\triangle_{efv}$ were obtained by subdividing the bridging triangle $\triangle_{bcg}$. Thus $\mathcal{P}$ intersects the two triangles; therefore none of degenerate situations occurs in this case.

(2) If $h(a) = 1$ then $1 > h(v) > h(e)$. By the same method as above we argue that $\mathcal{P} \cap \mathrm{Int}(\triangle_{abv}) \neq \emptyset$ and $\mathcal{P} \cap \mathrm{Int}(\triangle_{acv}) \neq \emptyset$. These two triangles were obtained by subdividing the bridging triangle $\triangle_{abc}$.

- 2-2, 2-3, and 1-3: These cases are treated similarly to the case 1-2.
- 3-3: By construction (see Section 4.2) $h(v) = \frac{1}{2}$.

□

Theorem 2 *The reconstructed surface does not contain any horizontal triangle.*

Proof: For any height other than 1/2 the claim follows from the definition of the algorithm. We will show the property for $h = 1/2$ by showing that there is no triangle in the reconstructed surface with all three vertices at this height. There are two types of triangles in the reconstructed surface: either original triangles of the input surface, or triangles created by our algorithm in the subdivision process. Triangles of the first type obviously never lie in the plane $z = \frac{1}{2}$. Let us then assume that the triangle is of the second type. Consider the three cases show in Figure 5:

- A triangle with a single bridging edge (Figure 5(a)): There is at least one vertex v lying on the original contour, thus $h(v) \neq \frac{1}{2}$.
- A triangle with two bridging edges (Figure 5(b)): In such a triangle both bridging edges lie at the same height, since all such edges are parallel to the planes of the slices, and the two bridging edges have a common vertex. Thus, the third non-bridging edge is a contour edge and the height of all the vertices of the triangle is either 0 or 1. The newly-added vertex v on the contour edge naturally lies at the same height. Therefore, every triangle created in the subdivision process in this case has at least one vertex that does not lie at height $h = 1/2$.
- A triangle with three bridging edges (Figure 5(c)): In this case all the vertices lie on one of the slices, i.e., at height $h = 0$ or $h = 1$. Therefore every triangle (after the subdivision step) has at least one vertex which is an original vertex of the input surface and hence does not lie at height $h = 1/2$.

□

Corollary 3 *The topology of the cross-section of the reconstructed surface at height $h = 1/2$ is a degenerate case of the topologies of both slices at heights $h = 0$ and $h = 1$.*

Proof: This claim is a result of Theorems 1 and 2, and the fact that the reconstructed surface is C^0-continuous. □

6 Experimental Results

In this section we show a few examples that illustrate the behavior of our algorithm on typical topologies.

Figure 7(a) shows a morphing example with a "pants" effect, where two separate contours become connected at their top. The intermediate slices demonstrate how smooth the morphing between the two shapes is. The course of the algorithm in this example is shown in Figures 7(b,c,d).

Figure 8 shows a one-to-many (contours) morphing example. In this example the multiplicity of the contours (and the gaps between them) induces the multiplicity of paths handled by our algorithm.

Figure 9 shows a morphing example between two brain slices (soft tissue). In this example one of the shapes contains a hole contour nested within another contour, whereas the other shape does not contain a matching hole. The morphing sequence shows how the hole appears "out of the blue" in an intermediate level and develops smoothly. Figure 10 shows a three-dimensional visualization of the reconstructed brain layer between the two slices.

Figure 11 shows a morphing between a 'b d' shape and an 'H' shape. This figure demonstrates the difficulty of morphing between two synthetic shapes: we expect the intermediate frames to consist of "smooth" contours, although the morphing sequence that optimizes some measure (in this case, minimizing the area of the reconstructed surface) may yield unsatisfactory slices.

7 Conclusion

In this paper we have described an algorithm for morphing between two-dimensional polygonal shapes. Our algorithm is based on a reconstruction algorithm; it modifies the reconstructed surface in order to create a morph sequence that changes the topology in a pleasing and smooth manner.

Our algorithm can be applied to any reconstruction algorithm that maintains xy-monotonicity, and guarantees that the reconstructed surface is free of self-intersections. Given any such surface, our algorithm maintains the property of avoiding self-intersection. Moreover, the topologies of the input slices are not altered, horizontal triangles are prohibited, and all the topology changes occur at a mid-height which is a degenerate form of both input topologies. Finally, the running time of our algorithm is linear in the complexity of the surface.

A couple of extensions of the algorithm are possible. First, features of the input contours can be matched either manually or automatically in various ways. In the software which we implemented, the matching is done by the reconstruction algorithm based upon xy-proximity. Second, one can use curve-smoothing algorithms for avoiding non-smooth features in the intermediate frames of the morphing.

References

[1] C. Bajaj, E. Coyle, and K. Lin, *Arbitrary topology shape reconstruction from planar cross sections*, Graphical Models and Image Processing, 58 (1996), pp. 524–543.

[2] G. Barequet, D. Shapiro, and A. Tal, *Multi-level sensitive reconstruction of polyhedral surfaces from parallel slices*, The Visual Computer, 16 (2000), pp. 116–133.

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

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

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

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

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

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

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

[10] 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.

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

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

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

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

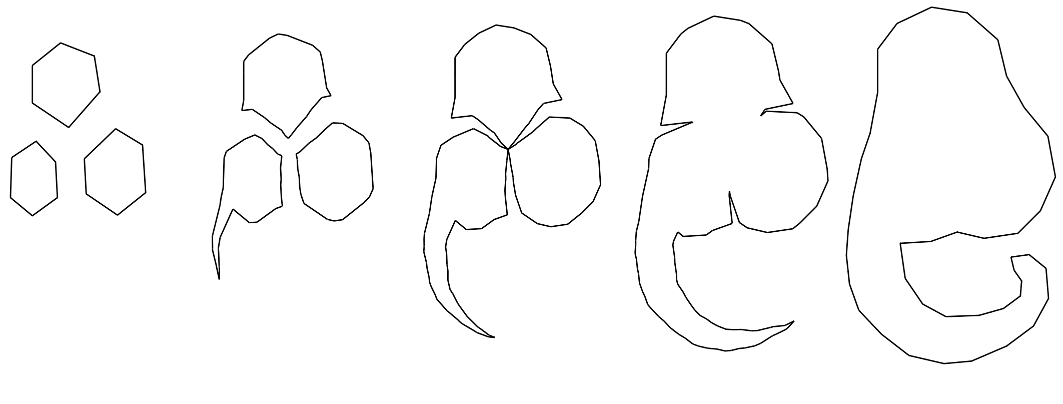

(a) Upper (b) (c) (d) (e) Lower

Fig. 2. The morphing sequence: The input slices ((a) and (e)) and the intermediate slices (b–d)

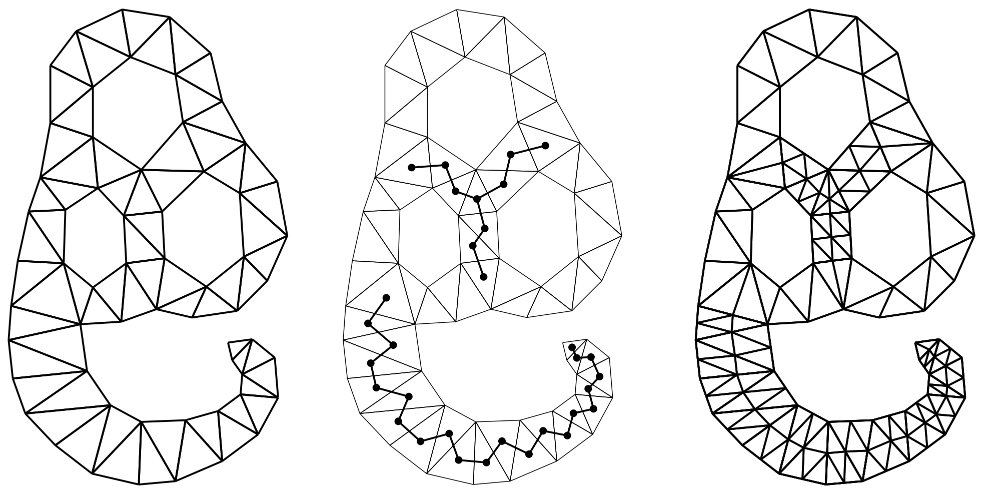
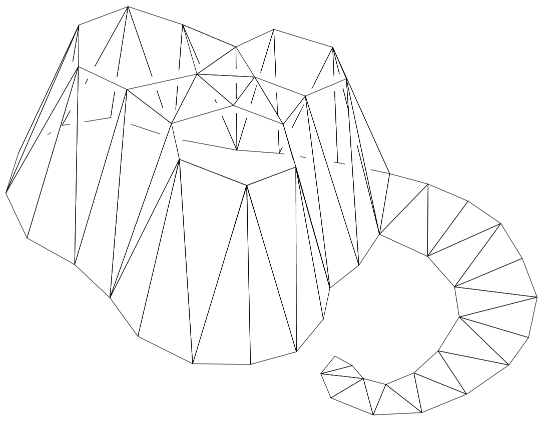
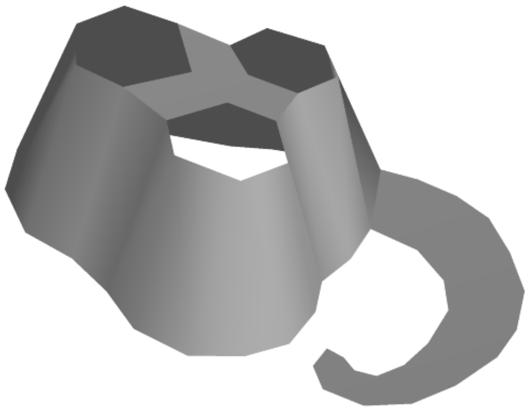

(a) Original [3] (b) Dual graph (c) Refined triangulation

Fig. 3. The xy-projections of the reconstructed surface

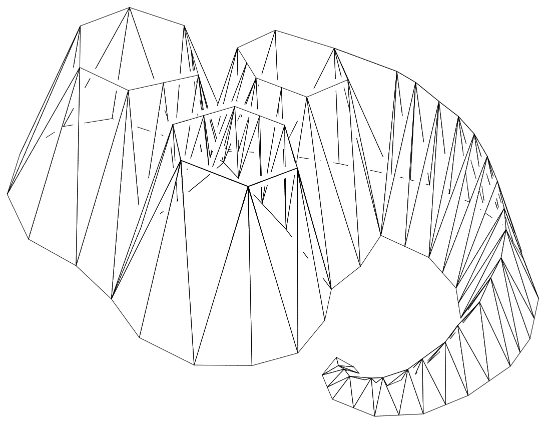


(a) A wireframe view

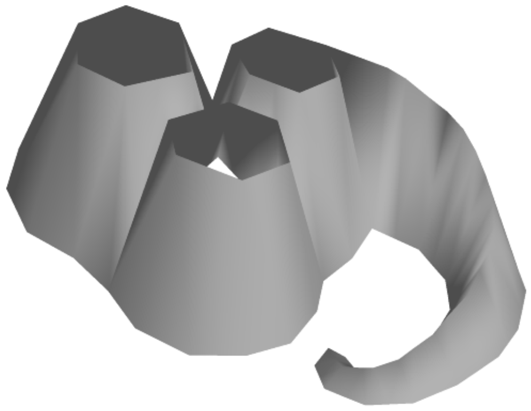


(b) A solid view

Original reconstruction [3]



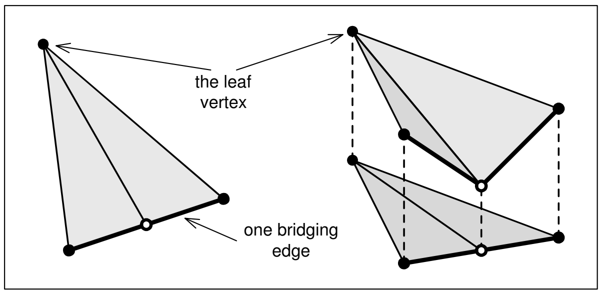
(c) A wireframe view



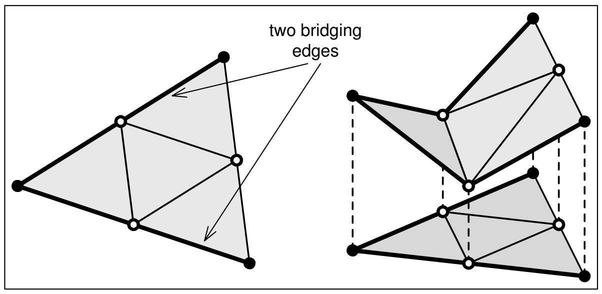
(d) A solid view

Improved reconstruction

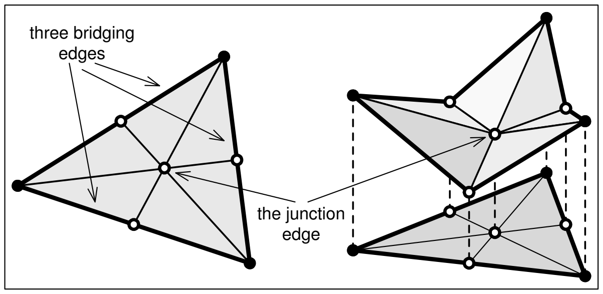
Fig. 4. A 3-dimensional view of the reconstructed surface



(a) One bridging edge



(b) Two bridging edges



(c) Three bridging edges

Fig. 5. Subdividing bridging triangles

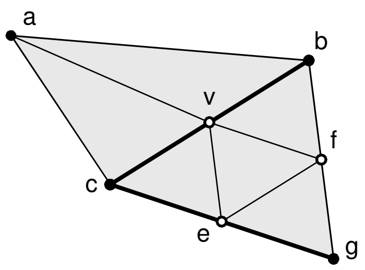
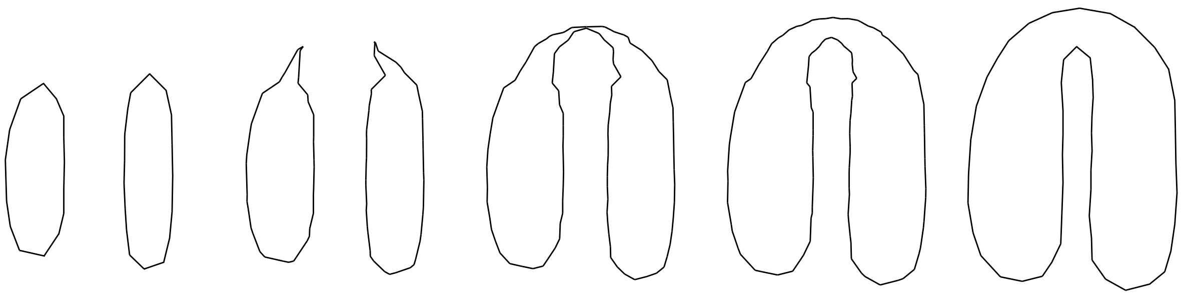
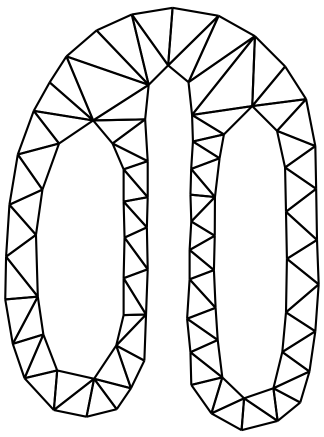


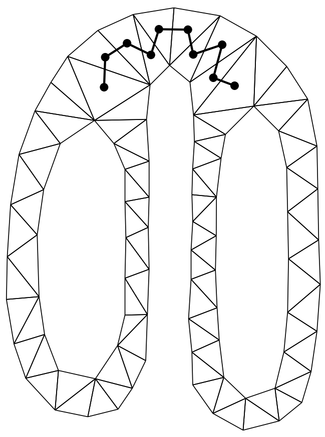
Fig. 6. Adjacent bridging triangles



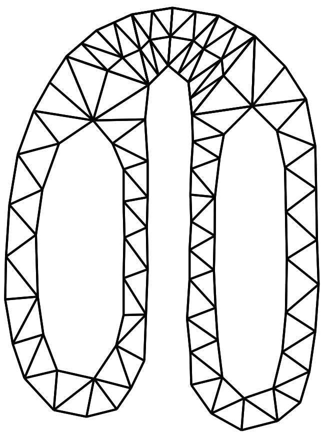
(a) Morphing sequence



(b) Original surface

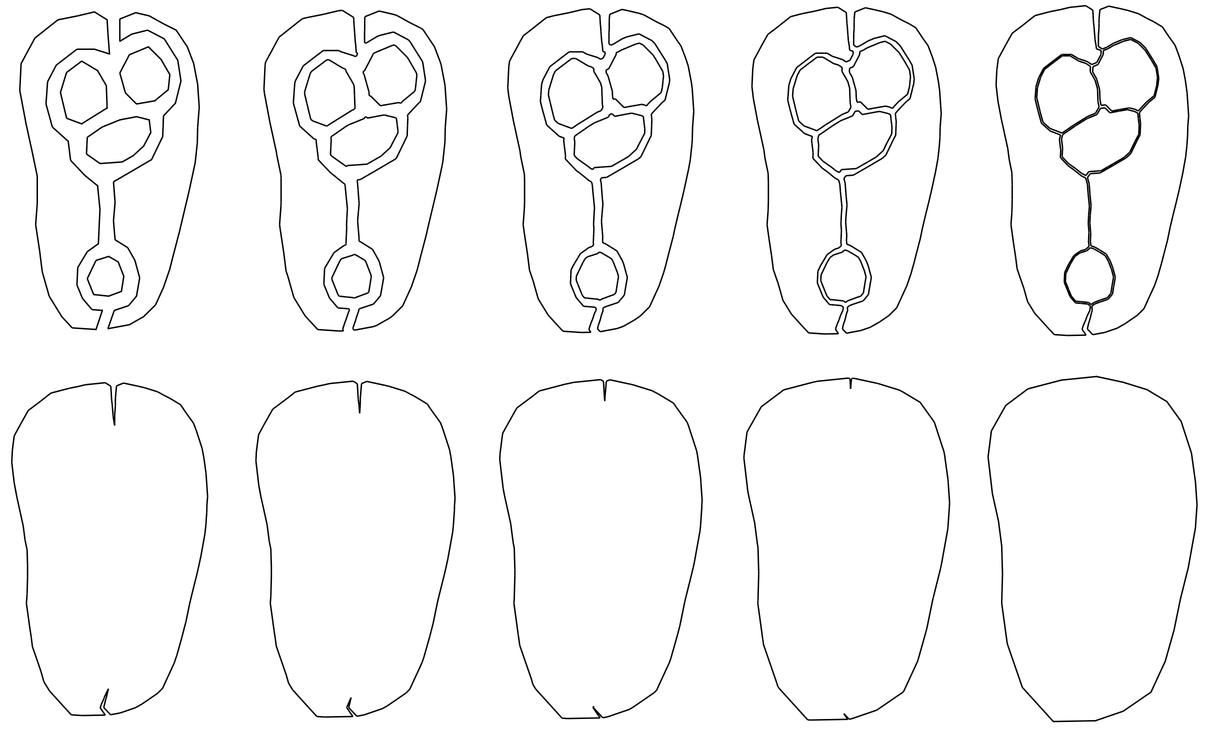


(c) Dual graph

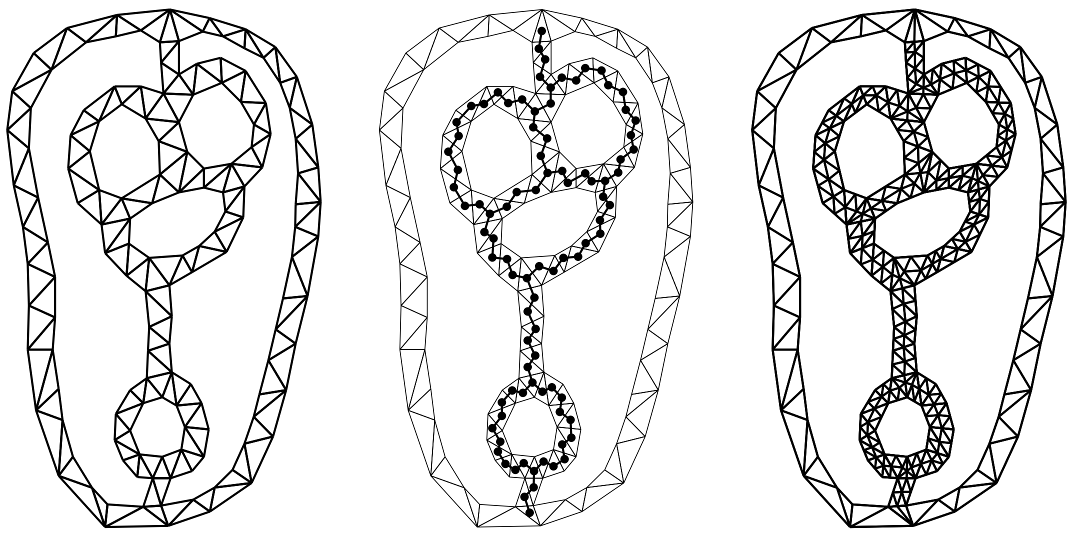


(d) Refined triangulation

Fig. 7. A "pants" effect

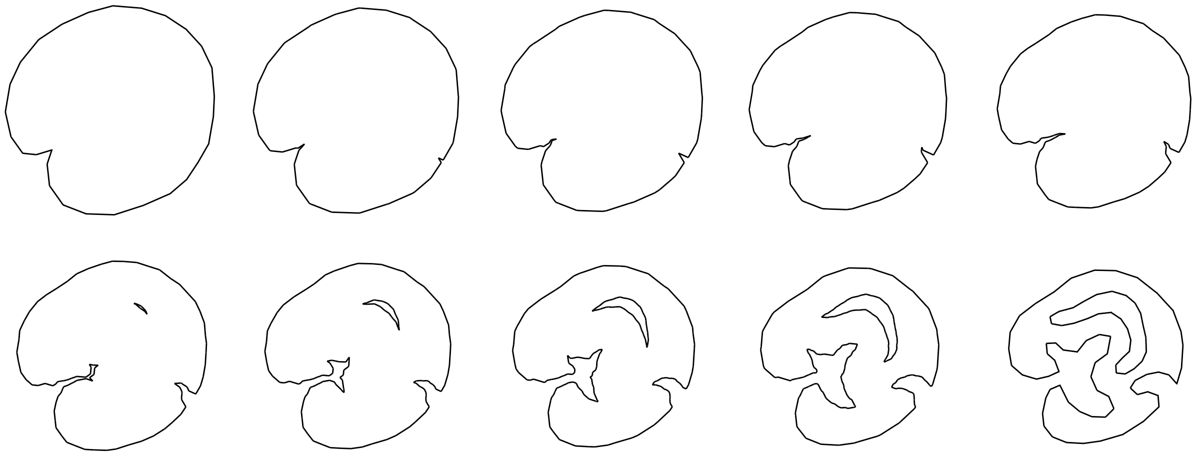


(a) Morphing sequence

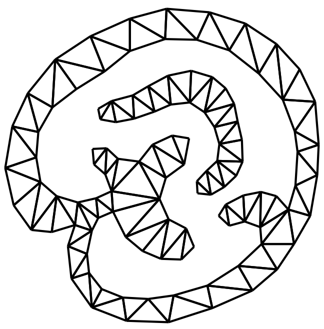


(b) Original surface (c) Dual graph (d) Refined triangulation

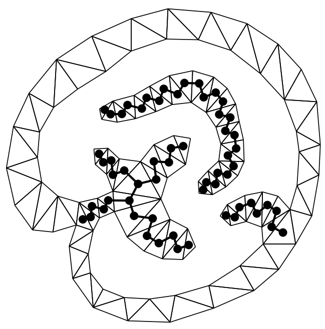
Fig. 8. Morphing between one to many contours



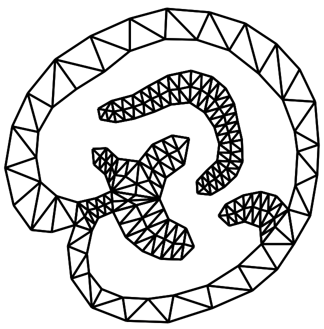
(a) Morphing sequence



(b) Original surface

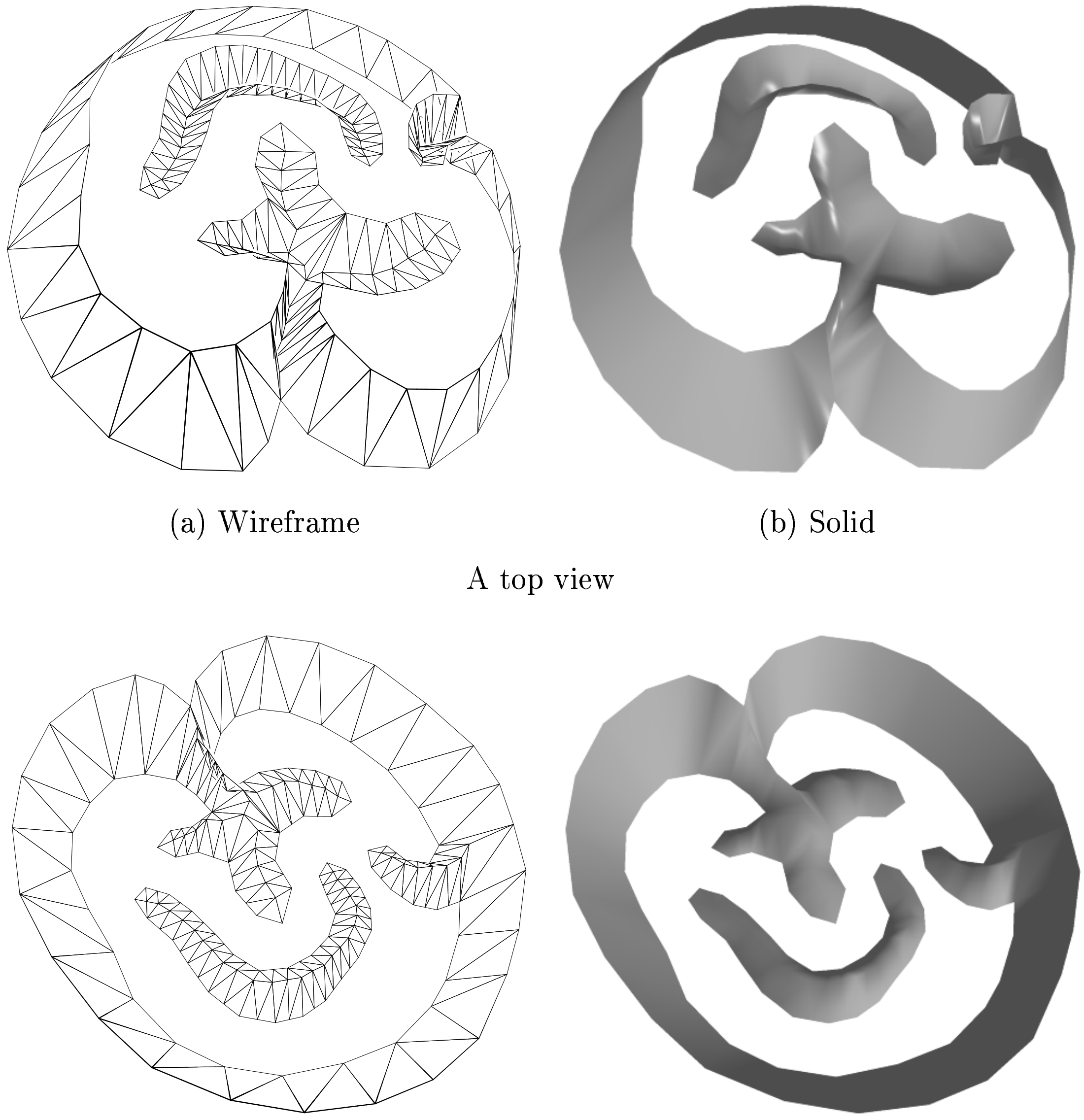


(c) Dual graph



(d) Refined triangulation

Fig. 9. Morphing between two brain slices



(a) Wireframe (b) Solid

A top view

(c) Wireframe (d) Solid

A bottom view

Fig. 10. A 3-dimensional view of the reconstructed brain tissue

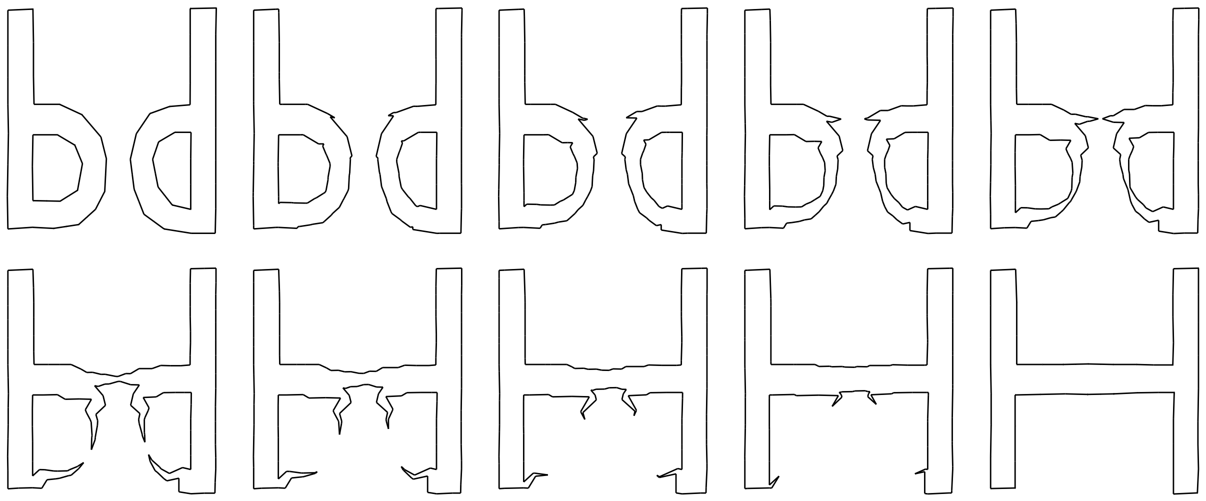


Fig. 11. Morphing between the letters ‘b d’ and the letter ‘H’