Shape Blending Using the Star-Skeleton Representation

Michal Shapira Ari Rappoport

Institute of Computer Science, The Hebrew University of Jerusalem
Jerusalem 91904, Israel. arir@cs.huji.ac.il

Abstract: We present a novel approach to the problem of generating a polygon sequence that blends two simple polygons given a correspondence between their boundaries. Previous approaches to this problem, including direct vertex interpolation and an interpolation based on edge lengths and angles between edges, tend to produce self intersections and shape distortions. Our approach introduces the *star-skeleton* representation, a structure comprised of equivalent decompositions of the two polygons into star-shaped pieces and a skeleton connecting the star pieces. We utilized the concept of a star polygon since two star polygons can be fairly blended without any self intersections.

We present algorithms for creating *equivalent* star-skeletons for the two polygons and for blending two star-skeletons by interpolating their skeletons and *unfolding* intermediate polygons from the skeletons. We also demonstrate the usage of the star-skeleton for blending multiple polygons and images. We show examples from a working system that demonstrate the improved blend sequences generated by the method.

The intrinsic reasons for the good results obtained are the fact that the interiors of the polygons are considered, not only the boundaries, and that the star-skeleton explicitly models an interdependence between all the vertices of the polygons.

Keywords: Shape blending, shape interpolation, morph, star polygon, skeleton, star-skeleton, key-frame animation, equivalent decompositions.

Publication: To be published in *IEEE Computer Graphics and Applications*, March 1995.

1 Introduction

Shape blending is a central problem in 2-D computer animation, involving a metamorphosis of one shape into another. In spite of impressive productions which utilize morphing, the problem is far from being solved. Efforts are needed for greater automation and reduction of manual work.

1.1 Background

In most animation systems 2-D shapes are represented as polygons. Polygon blending is usually considered a two-part process: generation of vertex *correspondence* and vertex *interpolation* to create intermediate polygons.

The choice of a suitable correspondence is best made by the animator, since it has a strong emphasis on the resulting animation and should depend on the animation script. There have been attempts for automatic determination of correspondence, in 2-D [1] and even in 3-D [2] [3]. Whether correspondence is given by the user or computed automatically, it still remains to specify an interpolation, a task which is more suitable for automation.

As mentioned above, the most common type of interpolation is vertex interpolation. Vertex interpolation can produce highly unpleasant results, even for simple shapes and simple transformations. Polygon edges easily intersect each other, causing a loss of the sense of blending between *shapes*. Simple geometric properties, such as lengths, angles, and areas, do not change in a consistent manner. An example is shown in the lower part of Figure 2. The principal reason for the bad results given by vertex interpolation is that the path of each vertex is determined independently of the path of any other vertex, and depends only on its interpolated positions.

Figure 1: Above: compatible star decompositions, each star piece shown in a different color. Below: the same sequence, showing the skeletons.

Sederberg et al [4] presented an improved interpolation method, in which interpolated entities are edge lengths and angles between edges rather than vertex locations. An optimization algorithm is performed, solely for the

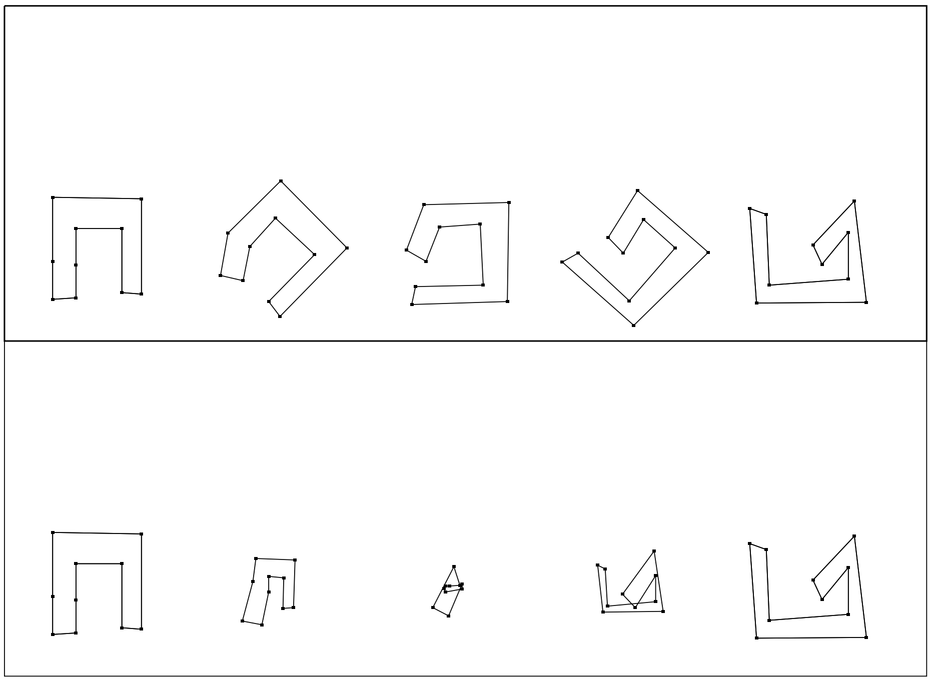

Figure 2: Above: star-skeleton. Below: vertex interpolation.

purpose of ensuring that the intermediate polygons define closed shapes. We will refer to this method as *edge-angle interpolation.* The method deals successfully with many situations, among which are cases in which the shapes are affine transformations of each other or in which parts of the shapes are transformed affinely. The upper part of Figure 2 shows the result obtained from the approach presented in the present paper. Edge-angle interpolation produces similar results.

However, there are many cases in which the method produces self-intersections of the boundary (Figures 5 and 6.) In addition, the method tends to distort the polygon area in intermediate shapes (Figures 3-6.) The reason is that different parts of the boundary are unaware of each other and that there is no explicit consideration of the interior area of the shape.

Similar types of problems also arise in image morphing, a computer animation technique in which intermediate information is required for each pixel, not only for a shape's outline. Beier and Neely [5] present an image morphing approach in which a set of corresponding line segments is drawn on the images. Every pixel is given local coordinates with respect to this set, an interpolation is performed on the line segments, and pixel values are determined according to their local coordinates and colors. Animators using the technique have demonstrated impressive results, but at a large manual and computational costs. The path of each line segment is independent of the paths of any other segment. As a result, one can encounter the same disturbing effects as with vertex interpolation. Fixing this problem necessitates manual intervention.

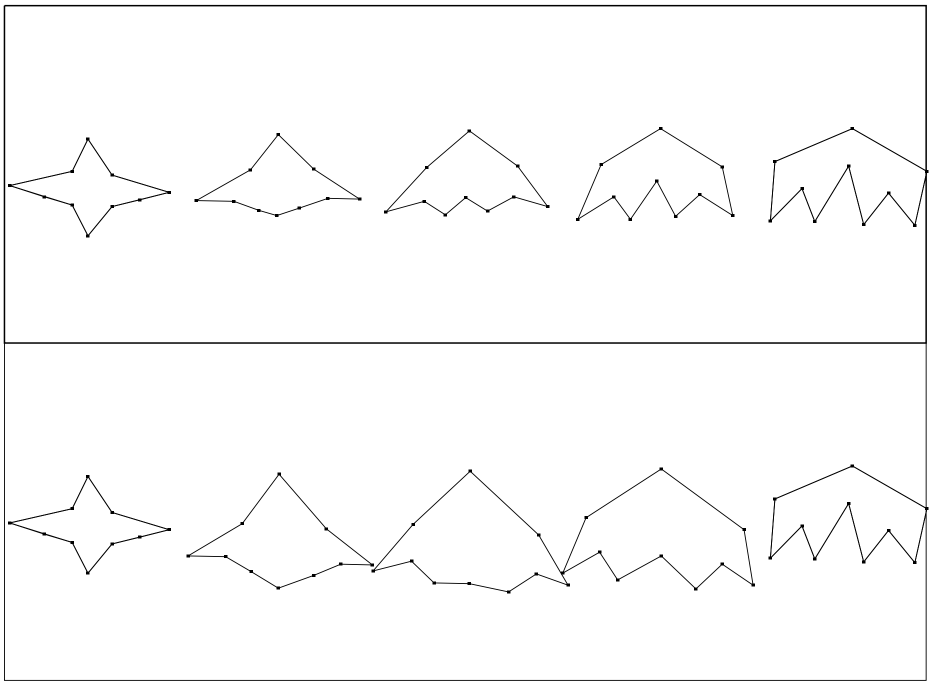

Figure 3: Above: star-skeleton. Below: edge-angle interpolation.

1.2 The Star-Skeleton Approach

We present a novel approach to automatic polygon blending, which takes into account the interior of the polygons as well as its boundary. The approach is based on a new polygon representation scheme which we call the *star-skeleton.* A *star polygon* is a polygon for which there exists a point (a *star point*) from which all other points are visible. The star-skeleton is comprised of two parts: a decomposition of the polygon into star-shaped pieces, each represented by its vertices and a special star point called the *star origin* (upper part of Figure 1); and a planar graph, the *skeleton*, that joins the star origins (lower part of Figure 1.) Shape interpolation is achieved by first interpolating between the skeletons and then *unfolding* the star pieces from the skeleton.

The star-skeleton is appropriate for the shape blending problem for two reasons. First, it represents all points in the interior of the shape as well as points on its boundary. Second, it defines an explicit dependence between interpolated polygon vertices relative to a common structure (the skeleton.) The motivation for choosing a star-based decomposition is that two star-shaped polygons can be blended such that their edges do not self intersect. The vertices of a star polygon are sorted by angle around any star point. Linear interpolation preserves the order of these angles. Thus the vertices of each intermediate polygon are also sorted by angle, which means that the polygon's edges do not self intersect. A star decomposition was preferred over, say, a convex decomposition since it has much fewer pieces. The star-skeleton produced better results than skeletons using a minimal convex decomposition or a triangulation.

In addition, we use the star-skeleton for multi-polygon shapes and image morphing. For the former, a meta-skeleton that joins the star-skeleton of each polygon is computed and interpolated. The latter is easy to achieve since the star-skeleton in effect defines local coordinates for each pixel inside every polygon.

The concept of a skeleton was used in animation already by Burtnyk and Wein in 1976 [6]. The user defined a very simple and restricted 'skeleton', and its interpolation on the key-frames by hand, and the system used it for correcting individual key-frames. A discrete medial-axis skeleton was used by Wolberg [7] for warping a single

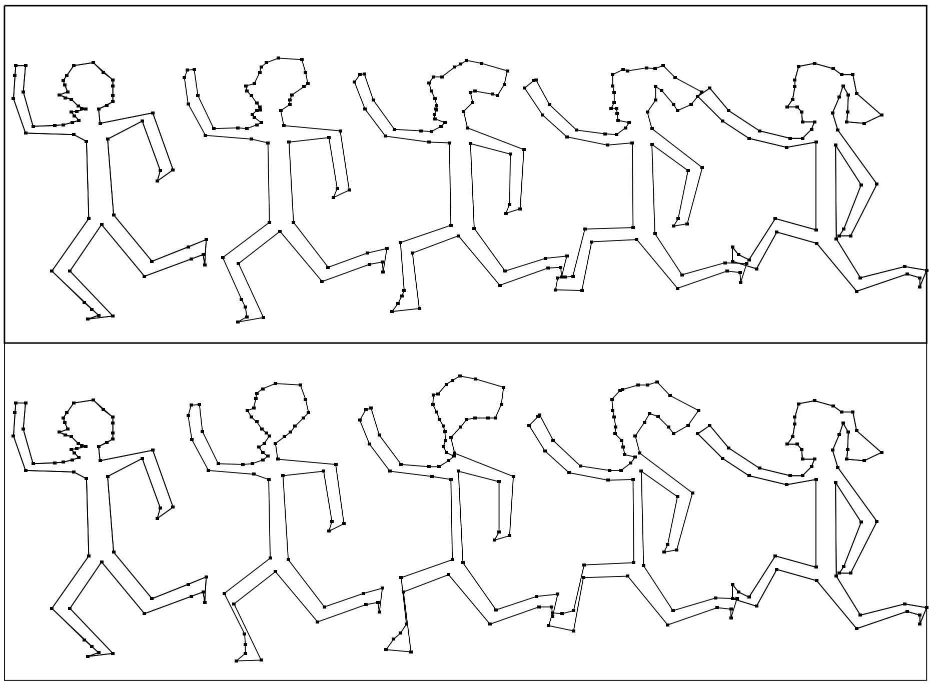

Figure 4: Above: star-skeleton. Below: edge-angle interpolation.

image from one polygon to another. This method does not generate intermediate images and works only in image-space. Lazarus et al [8] use a skeleton to define a deformation entity. The user draws a skeletal curve which defines local coordinates for each object point, computed by projecting the point on the nearest curve point. The user then modifies the curve, and the object is deformed using these local coordinates. The method is very different from ours, since we compute the skeleton automatically, our skeleton provides more flexibility and better control since it is a tree, not just a curve, and our computation of local skeletal coordinates is based on the shape itself, not on a simple projection on the skeleton.

From now on we assume that the problem is of blending between *two* polygons. Most presented concepts and algorithms are suitable for more than two polygons. The paper is organized as follows. Section 2 gives notations, an exact definition of the star-skeleton representation, and algorithms for computing compatible star-skeletons of two polygons from their vertex lists. Section 3 describes how the star-skeleton is used for polygon blending. Section 4 shows how to use the star-skeleton for blending multiple polygons and images. Results and examples are given in Section 5.

2 The Star-Skeleton Representation

In this section we describe our notations, define the star-skeleton representation, and present algorithms for computing compatible star-skeletons of two polygons from their vertex lists.

2.1 Notations and Basic Definitions

A *polygon* is defined by a sequence of points (*vertices*) such that the first vertex is identical to the last one. A polygon *edge* is the line segment connecting two consecutive vertices. A *simple polygon* is a polygon whose edges

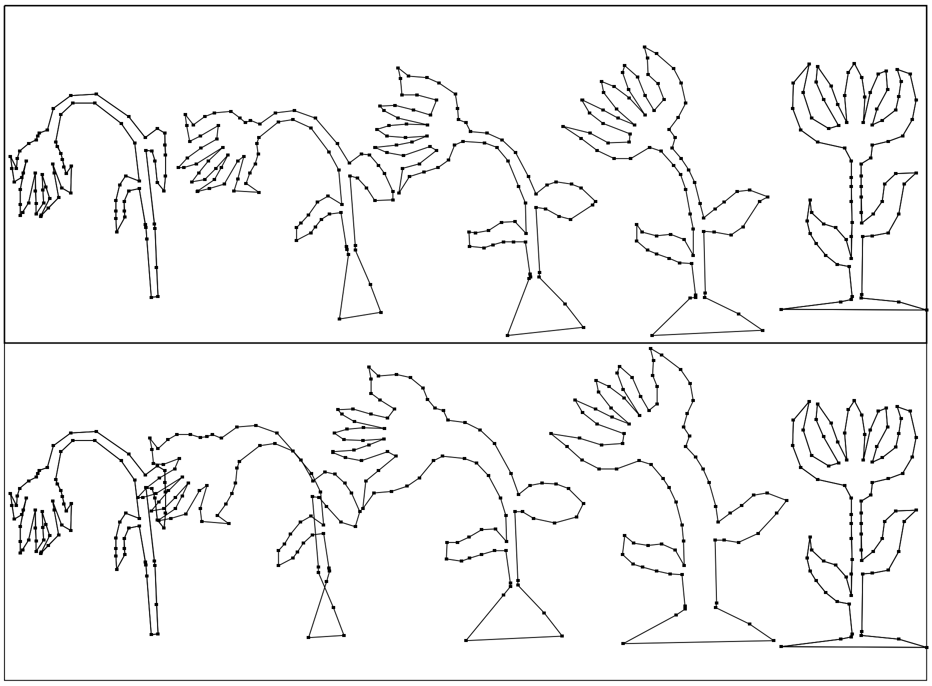

Figure 5: Above: star-skeleton. Below: edge-angle interpolation.

do not intersect each other except when they topologically share a vertex. Let P denote a polygon, including its interior and boundary. A point $v \in P$ is *visible* from a point $w \in P$ if the line segment $[v, w]$ is contained in P (including its boundary.) A simple polygon is *convex* if every pair of points in it see each other; it is *star-shaped* (or a *star polygon*) if there exists a point $v \in P$ that is visible from any other point in P. In such a case v is called a *star point.* If a vertex is a star point it is called a *star vertex.* A line segment $[u, w]$ is a *diagonal* of P if u, w are vertices of P and see each other. A point $u \in P$ is a *Steiner point* if it is not a vertex of P.

For every simple polygon we can define a graph called the *visibility graph.* Its nodes are the polygon vertices, and an arc exists between two nodes if and only if the corresponding vertices see each other. A *complete correspondence* between two polygons is a one-to-one mapping between their vertices that preserves vertex adjacencies and orientation.

2.2 Definition of the Star-Skeleton

The star-skeleton of a simple polygon P is comprised of two components:

- The *star set* – a set of star-shaped polygons called *star pieces.* Each star piece possesses a special star point called the *star origin,* and a direction called the *reference direction.*

  If there is more than one star piece in the star set, each star piece possesses at least one star piece neighbor, in the sense that they share an edge. Such an edge is called a *shared edge,* and its vertices are called *shared vertices.* A shared edge must be internal to the polygon P. Shared vertices do not have to be vertices of P, since vertices of star pieces are not necessarily vertices of P. It is not prohibited that two star pieces intersect each other.

  The vertices of each star piece are represented in polar coordinates with respect to the star origin of the piece and its reference direction. That is, each vertex is defined by the distance from the star origin and by the

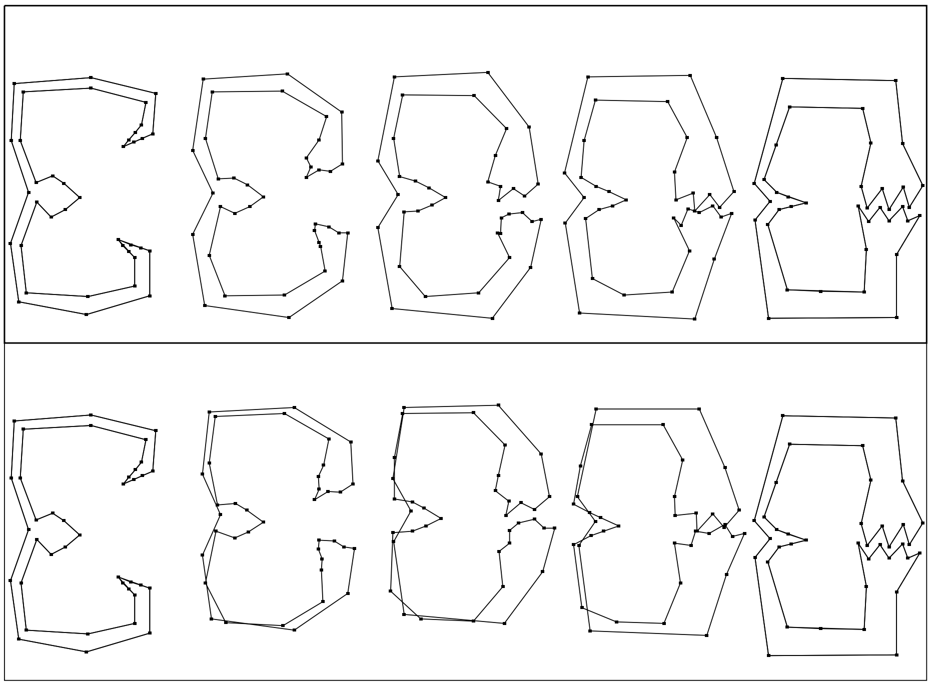
Figure 6: Above: star-skeleton. Below: edge-angle interpolation.

angle formed between the vector from the star origin in the reference direction and the vector from the star origin to the vertex.

- The *skeleton* – a planar tree alternately comprised of two kinds of points: the star origins and mid-points of shared edges, referred to as *mid-points*.

  Each skeleton vertex has an associated direction called the reference direction. The root's reference direction is the x axis. The reference direction of any other vertex is the vector from the vertex to its parent. The reference direction of a star piece is the reference direction of its star origin.

  The root is represented in Cartesian coordinates. Each other vertex is represented in polar coordinates with respect to the parent and its reference direction. That is, a vertex stores the distance to the parent and the angle between the reference directions of the parent and of itself.

Figure 7 shows two star-skeletons. For the polygon on the left, the star set consists of three star pieces. For example, the bottom one is defined by vertices 2-7. The three star origins are shown as black squares. The star origin of the middle star piece coincides with vertex 8, which is also the root of the skeleton. There are two shared edges, $[0, 11]$ and $[2, 7]$, and two mid-points, drawn as white squares.

Angles of star piece vertices are always measured from the star piece reference direction. However, there is still an ambiguity regarding the actual number that represents the angle, since 2π can be added or subtracted and the angle remains the same. The choice of number is important, since two such numbers are interpolated to form the angle in an intermediate polygon. It order to avoid self intersection of the star piece in an intermediate polygon, the angles of the star vertices should be equivalently sorted in both input polygons, and their range should not exceed 2π. This is achieved by denoting the angle of a vertex by the sum of the angle of the vertex from the the first vertex of the star, and the angle of the first vertex from the reference direction (α). Thus the angles of the vertices are equivalently sorted in both polygons, and ranged from α to $\alpha + 2\pi$.

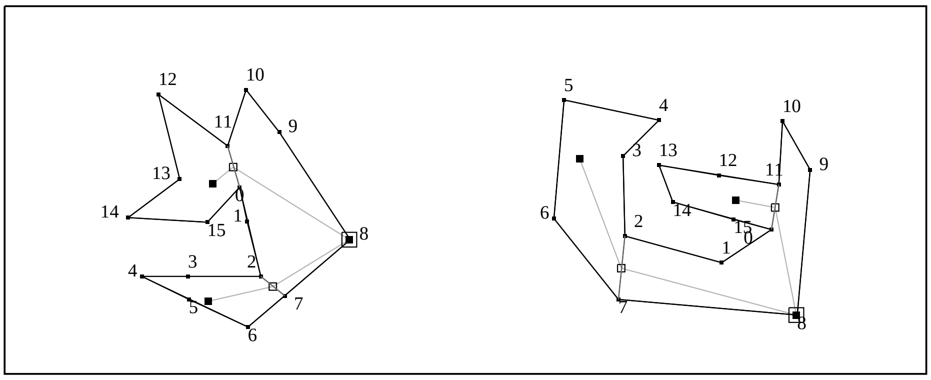

Figure 7: Compatible star-skeletons.

2.3 Definition of Compatible Star-Skeletons

Our task is to find a nice blend between *two* polygons P and Q. Compatible star-skeletons are needed in order to be able to interpolate the skeletons. A prerequisite for defining compatible star-skeletons is a complete correspondence between P and Q. If only a partial correspondence is given, we first expand it into a complete correspondence (Section 5.)

Let A and B be two star-skeletons of P and Q, respectively. Denote the i-th star piece in A by A_i and the j-th vertex of A_i by A_{ij}, and similarly for B. Assume that there exists a one-to-one mapping between the set V_A of vertices of all star pieces in A and the set V_B of vertices of all star pieces in B. Star piece vertices that are also vertices of P or Q possess such a mapping due to the complete correspondence requirement. The mapping assumption means that vertices which are not vertices of P or Q (Steiner vertices) must also possess such a mapping. In both cases we use the phrase 'a vertex in V_A corresponds to a vertex in V_B'.

We say that A and B are *compatible* if:

- The star sets of A and B are comprised of compatible star pieces. Two star pieces A_i, B_i are compatible if
  - For each j, A_{ij} corresponds to B_{ij}. That is, A_i and B_i are defined by corresponding vertices of V_A and V_B.
  - Their star origins are convex combinations involving corresponding vertices weighted by the same weights. In this case we say that we have corresponding star origins.

  Note that if two star pieces are compatible and they have shared edges then they have corresponding shared vertices and corresponding mid-points.
- The two skeletons are isomorphic when identifying corresponding star origins and corresponding mid-points. That is, the skeleton trees have exactly the same structure.

Figure 7 shows two compatible star-skeletons. Note that the numbers, adjacencies and character of star pieces, polygon vertices and skeleton vertices are identical.

The motivation for the convex combination requirement is that a correspondence between the origin points and the boundaries induces a correspondence between *all* internal points of the polygons. Hence the choice of star origins has to be consistent with the existing correspondence between the boundaries.

2.4 Computation of Compatible Star-Skeletons

Construction of compatible star-skeletons for two polygons is done in four stages: construction of compatible star decompositions; determination of a star origin in each star piece; construction of the skeleton itself; and representation of the vertices of each star piece in polar coordinates with respect to the skeleton.

2.4.1 Construction of Compatible Star Decompositions

Star decompositions of two polygons are compatible iff (1) every star piece in one decomposition has a mate in the other decomposition which is defined by corresponding vertices; (2) in each pair of corresponding star pieces there is at least one pair of corresponding star vertices. The reason for the second requirement is to ensure that corresponding star origins can be found for each pair of star pieces.

Note that a pair of polygons might not possess compatible star decompositions without adding Steiner vertices. This case is illustrated in Figure 8. The two polygons cannot be compatibly star decomposed without adding Steiner points. The polygon on the right is not a star, and any diagonal of this polygon is not a valid diagonal in the left polygon. In order to decompose the two polygons compatibly, Steiner points must be added, as shown in Figure 8. In this case, two Steiner points were added, yielding a star decomposition of four star pieces: $\{0, 1, a, 7\}, \{a, 1, 2\}, \{2, 3, 4, b, a\}, \{a, b, 4, 5, 6, 7\}$.

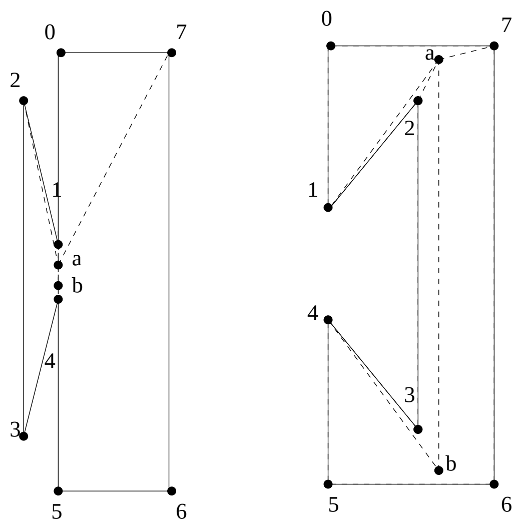

Figure 8: Compatible star decompositions with Steiner points.

Algorithms for constructing compatible star decompositions are given in [10]. These algorithms are not described here since they solve an independent problem, and can be regarded as a black box for the algorithms given here. Any algorithm which constructs compatible star decompositions of two polygons can be used to build the compatible star skeletons. Two algorithms are described in [10]. The first one constructs compatible star decompositions with minimal number of star pieces without adding Steiner points. This algorithm is a dynamic programming algorithm based on an algorithm by Mark Keil [11] [12] for computing a minimal star decomposition of a single simple polygon. It runs in $O(n^4)$ time, where n is the number of vertices of each polygon. The second algorithm constructs compatible star decompositions with Steiner points in time $O(C \times S)$, where C is the size of the minimal convex decomposition of one of the polygons, and S is the time needed for decomposing a single polygon into star pieces. The decompositions created by the second algorithm are not minimal, but possess no more than $O(n^2)$ Steiner points (which also means they possess no more than $O(n^2)$ star pieces.) The output of these algorithms consists of two structures. The first structure is two compatible star decompositions of the two polygons. The second is the visibility graphs of the star pieces in the two polygons which are needed for the determination of the star origins (see below.)

Decomposing a polygon into star pieces requires computation of the polygon's visibility graph. Hence, the star-skeleton blending algorithm's robustness is related to the robustness of the computational primitive "do two points see each other".

2.4.2 Determination of Star Origins

Each pair of corresponding star pieces is handled independently of any other pair. For each star piece pair we determine the set of common star vertices using the visibility graphs of the star pieces, which were created in the previous stage. Recall that each pair of corresponding star pieces includes at least one pair of corresponding star vertices. This ensures that the set of common star vertices is not empty. The star origins are taken to be convex combinations of these star vertices, which guarantees that the chosen star origin is a star point (since the set of all star points is convex.) In our implementation, the vertex weights in the convex combination are identical, so the star origin is simply the center of mass of the common star vertices. We experimented with other alternatives, but none was superior to the center of mass.

2.4.3 Skeleton Construction

The skeleton consists of two types of vertices, star origins and mid-points. The role of the skeleton is to join the star origins so that the relative motion of star pieces during the blending process can be determined. Mid-points were added to ensure that the skeleton lies completely inside the polygon, otherwise its motion does not reflect the motion of the shape as a whole.

To construct the skeleton, view the star pieces as forming the nodes of a planar graph, with an arc between two nodes whose corresponding star pieces have a shared edge. A tree must be extracted from this planar graph. The main issue here is the choice of a root, since after that any graph traversal (e.g. depth-first) can be used to extract a tree.

The choice of a root has an influence on the resulting blend. The root is the only skeleton vertex represented in Cartesian coordinates; during the blend, its location is simply an interpolation of its positions in the two blended polygons. As a result, a sense of being the shape's 'center' is created. One of the star origins or mid-points can be selected as a root by the user, or the system can find a star piece with a maximal number of neighbors and define the root to be its star origin.

2.4.4 Polar Vertex Coordinates

Once we have the skeleton, each star origin possesses an associated reference direction, and the vertices of each star piece can be represented by polar coordinates relative to the star origin and its reference direction.

3 Polygon Blending

In this section we describe the usage of the star-skeleton for polygon blending. Generation of an intermediate polygon at a given time instance t is done in two steps: first, generation of an intermediate star-skeleton, and second, unfolding the polygon from its star-skeleton. To get a feeling for the process, see Figure 1.

3.1 Star-Skeleton Interpolation

The star-skeleton is defined by a collection of points of different types: polygon vertices, star origins, and mid-points. Generation of an intermediate star-skeleton is performed by interpolating the coordinates of these points at the desired time instance. Each of the above points, except the skeleton root, is represented in polar coordinates. Polar coordinates interpolation involves interpolation of distances and angles. In our implementation we perform a simple linear interpolation. Interpolation of the root vertex is done in Cartesian coordinates.

3.2 Unfolding the Polygon

By 'unfolding' we mean the computation of the Cartesian coordinates of each vertex on the boundary of the polygon so that it can be directly displayed. Unfolding is comprised of two stages: computation of the Cartesian coordinates of each skeleton vertex, and computation of the Cartesian coordinates of each vertex of the resulting polygon. The skeleton vertices adjacent to the root are first computed, then their neighbors in a recursive manner. Once all star origins on the skeleton have been computed, the vertices of each star piece can be computed.

It is now clear why skeleton vertices and polygon vertices are represented using a reference direction. The reference direction of a skeleton vertex is the skeleton edge connecting it to its parent in the tree. The coordinates of its children are interpolated relative to this direction to achieve coordinated motion between the different star pieces composing the skeleton. The reference direction of a polygon vertex is the reference direction of the star origin of the star piece to which it belongs, itself changing during the interpolation since it is a skeleton edge. The fact that the vertex coordinates are interpolated relative to this direction ensures that the star piece will move in accordance with the skeleton motion.

It is certainly possible (in fact, it is the common occurrence) that a shared vertex will be given different Cartesian coordinates by the two star pieces to which it belongs. In this case we take the mid-point of the segment connecting the two positions.

4 Multiple Polygons and Image Morphing

It is easy to extend the star-skeleton approach to provide blending for shapes defined by multiple simple polygons and to morphing of images contained within simple polygons. For multiple polygons, we first compute compatible star-skeletons for each pair of corresponding polygons. We then compute a *meta-skeleton*, which is a planar tree connecting the roots of the computed star-skeletons. Meta-skeleton representation is similar to the representation of the skeleton part of the star-skeleton. During the interpolation process, the star-skeletons will move according to the motion of the meta-skeleton. This creates an automatic interdependence between the motions of the different polygons. A simple example for multiple polygon blending is shown in Figure 9.

When images are contained within one or both of the input polygons, the star-skeleton can be used to compute an intermediate image as well as an intermediate outline. The star-skeleton of a polygon provides local coordinates for each point inside the polygon, not only for points on its boundary, by representing them in polar coordinates relative to the star origin and its reference direction. To create an intermediate image, for each of its pixels we find their corresponding pixels in the initial and final images and use any appropriate filtering procedure to generate the pixel color [9]. An example is shown in Figure 10.

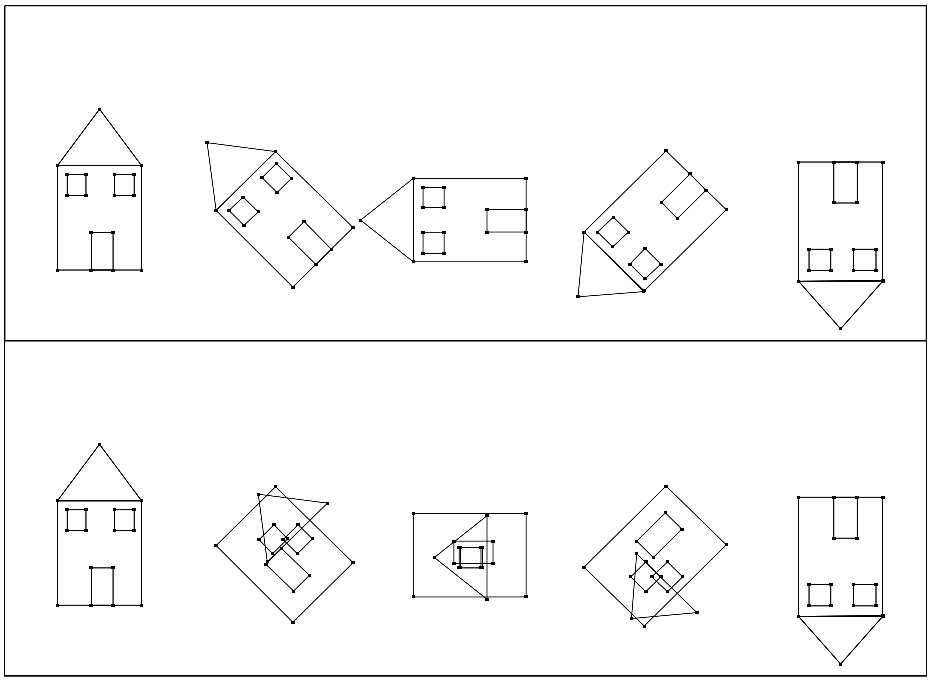

Figure 9: Above: shape blending using the meta-skeleton. Below: separate interpolation of each polygon.

5 Results and Discussion

To experiment with the ideas presented in the paper, we implemented a 2-D shape blending system. The system is written in C and runs under SGI Unix workstations using the Motif toolkit for the user interface and SGI/GL for graphics. It lets the user draw polygons, specify partial correspondence between their boundaries, and choose between three types of blends: vertex interpolation, edge-angle interpolation [4], and star-skeleton interpolation. A partial correspondence is automatically expanded into a complete correspondence, if necessary adding new vertices according to edge lengths. In addition, an image can be positioned inside a polygon, in which case the system computes an image morph sequence. The system is capable of generating postscript files containing several keyframes of an animation sequence. All figures in this paper, except Figure 8, were created using this software.

We compared our results to those obtained by the other two interpolation methods. Vertex interpolation produces unpleasant results in most cases (e.g. Figure 2) hence Figures 3-6 show only our results (in the upper part) vs. edge-angle interpolation (in the lower part.) Note that edge-angle interpolation readily produces self intersections and shape and area distortions, effects automatically solved by the star-skeleton method. The star-skeleton method might also create self intersections. However, these sporadic self intersections are either ‘natural’ or local. We deem a self intersection to be ‘natural’ if the polygon would be valid in a 2.5D world (Figure 11.) Self intersections like those shown in Figures 2, 5 and 6 where the whole polygon is distorted by one edge penetrating another, causing the polygon’s interior to disappear, are avoided.

We feel that the reasons for the good results produced by the method are the fact that the polygon interiors are explicitly considered and that the star-skeleton models an interdependence between the vertices of the polygons. The fact that the method is so naturally generalized to deal with blending of multiple polygons (Figure 9) and

Figure 10: An image morph sequence.

image morphing (Figure 10) provides another indication to its advantages.

To produce the simplest star-skeletons possible, we preferred a *minimal* star decomposition. The decomposition algorithms described in [10] are computationally expensive, taking a few seconds for the examples in Figures 4 and 5. It is important to emphasize that the blend itself is performed in real-time once the two compatible star-skeletons are computed. To improve the usability of the method, faster algorithms for compatible star decompositions should be explored. These should not necessarily compute an optimal decomposition like ours, but one that yields skeletons which are simple enough.

References

[1] Sederberg, T.W., Greenwood, E., A physically based approach to 2D shape blending, *Computer Graphics*, 26:25-34, SIGGRAPH'92.

[2] Bethel, E.W., Uselton, S.P., Shape distortion in computer-assisted keyframe animation, *Computer Animation '89*, Magnenat-Thalmann and Thalmann, (Eds), pp. 215-224, Springer, Tokyo, 1989.

[3] Kent, J.R., Carlson, W.E., Parent, R.E., Shape transformation for polyhedral objects, *Computer Graphics*, 26:47-54, SIGGRAPH'92.

[4] Sederberg, T.W., Gao, P., Wang, G., Mu, H., 2D shape blending: an intrinsic solution to the vertex path problem, *Computer Graphics*, 27:15-18, SIGGRAPH'93.

[5] Beier, T., Neely, S., Feature-based image metamorphosis, *Computer Graphics*, 26:35-42, SIGGRAPH'92.

[6] Burtnyk, N., Wein, M., Interactive skeleton techniques for enhancing motion dynamics in key-frame animation, *Communications of the ACM*, 19(10):564-569, Oct. 1976.

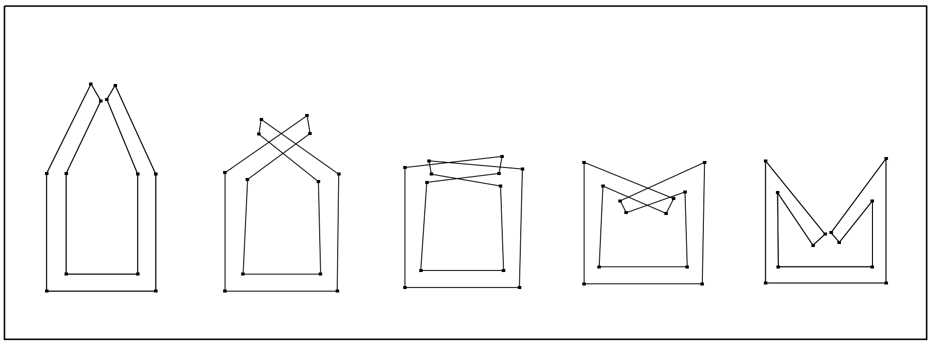

Figure 11: A ‘natural’ self intersection of a polygon.

[7] Wolberg, G., Image warping among arbitrary planar shapes, *New Trends in Computer Graphics*, Magnenat-Thalmann and Thalmann, (Eds), pp. 209-218, Springer, 1988 (proceedings *Computer Graphics International ’88.*)

[8] Lazarus, F., Coquillart, S., Jancène, P., Interactive axial deformations, *IFIP Conference on Geometric Modeling in Computer Graphics,* Genova, Italy, June 1993. Published in: Falcidieno, B., Kunii T.L. (Eds), Geometric Modeling in Computer Graphics, pp. 241-254, Springer, 1993.

[9] Wolberg, G., Digital Image Warping, IEEE Computer Society Press, Los Amiltos, California, 1990.

[10] Shapira, M., Rappoport, A., On compatible star decompositions, Technical Report, Institute of Computer Science, The Hebrew University of Jerusalem, 1994.

[11] O’Rourke, J., Art Galleries Theorems and Algorithms, ch. 8-9, Oxford University Press, NY, 1987.

[12] Keil, M., Decomposing a polygon into simple components, *SIAM J. Comput.*, 14:799-817, Nov. 1985.