A Fuzzy Approach to Digital Image Warping

Yuefeng Zhang
Honeywell Hi-Spec Solutions

A new algorithm uses fuzzy techniques to warp polygons that have different locations, orientations, sizes, and numbers of vertices. The algorithm is robust and extensible to curved shapes.

Digital image warping addresses the problem of how to smoothly transform one digital image into another. The warping process has wide applications in computer animation and can be divided into two classes depending on the type of images being transformed. *Gray image warping* considers the transformation of one grayscale image into another—a process also known as image metamorphosis.[1] *Binary image warping*, on the other hand, addresses the transformation of binary images such as polygonal shapes.[2,3] My focus here is on warping polygons.

We can approach the warping of polygonal shapes in two steps.[2] The first establishes a correspondence between the vertices of two given polygons. The second step interpolates the corresponding vertices to generate vertices of an intermediate polygon. This article presents new approaches to both steps.

Related work in vertex correspondence

The vertex correspondence problem also arises in pattern recognition and surface reconstruction. In pattern recognition, partially obscured and overlapping polygonal objects can be recognized by establishing the vertex correspondence of the polygons.[4,5] Kalvin et al.[4] proposed a method of finding the corresponding vertices between an observed object and a reference object by minimizing the distance between the two sequences of the objects' vertices. One disadvantage of this method is that it does not work for objects of different size (that is, scale). Tsang et al.[5] avoided this difficulty by using polygon edge ratios and interior angles to establish the vertex correspondence of two polygonal objects. The algorithm I present here uses this idea in conjunction with fuzzy techniques[6] to establish the vertex correspondence of two polygons.

Given a set of planar contours representing cross sections of a polyhedral object, reconstructing the object's surface requires the vertex correspondence between two adjacent cross sections. Researchers have used graph theory successfully to find such a correspondence. Based on graph theory, Fuchs et al.[7] proposed an approach that can find an optimal correspondence. To handle polygons of different size and location, Christiansen and Sederberg[8] presented a technique that establishes a vertex correspondence of two polygons by mapping them onto the same unit square.

In the recent physically based blending approach, Sederberg et al.[2] introduced a cost function to associate weights with the graph nodes. The vertex correspondence of two polygons is established by finding a path with the minimum cost in the graph. The new vertex correspondence method presented here is similar to this method in that a correspondence is established by finding a path in a graph. The main difference is that the new method uses a totally different function to assign weights to the graph nodes. In particular, a new fuzzy polygon similarity function associates a weight with each graph node. The weight represents the similarity degree of the triangles formed by the two polygon angles associated with this node and their adjacent edges.

Related work in vertex interpolation

After a correspondence between the vertices of two polygons has been established, the next problem in transforming one polygon into the other is how to interpolate their corresponding vertices. Such an interpolation should produce a smooth transformation at any intermediate polygon.

The most widely used interpolation technique is linear interpolation.[2] The disadvantage of applying this technique directly to the corresponding vertices of polygons under warping is that the resulting in-betweening is sometimes significantly distorted compared with the original polygons. The physically based blending approach[2] alleviates this difficulty by imposing constraints on the angles of the polygons. This approach is further improved in the intrinsic shape-blending method[3] by interpolating the lengths and orientations of the edges rather than the vertices of the polygons.

0272-17-16/96/$5.00 © 1996 IEEE

The new interpolation method presented in this article interpolates both the vertices and orientations of the polygons. It is based on Shoemake and Duff's method,[9] which produces an in-betweening by interpolating transformation matrices. The transformation matrices are decomposed into polar form[9] before interpolation begins. Generally speaking, it is difficult to use the transformation matrix approach to warp nontrivial polygons, since a transformation matrix common to all the vertex correspondence between the two polygons may not exist. The method presented here avoids this difficulty by using fuzzy affine transformation.

Fuzzy vertex correspondence

Vertex correspondence plays an important role in warping polygons. A reasonable correspondence can guide the interpolation process of producing smooth in-betweening. An unreasonable correspondence, however, misleads the process and thus prohibits the generation of smoothly changing intermediate polygons. In general, finding a good correspondence without human intervention is difficult if the source and target polygons have different locations, orientations, sizes, or shapes. It is even more complicated when the two polygons have different numbers of vertices.

The fuzzy technique described in this section can establish the vertex correspondence of two polygons with different locations, orientations, sizes, and numbers of vertices. The technique uses the target polygon to construct a fuzzy set of polygons that are similar to a given source polygon. The vertex correspondence is established by finding a polygon in the fuzzy set that maximizes the membership function of the set. In the following, this membership function is called the *polygon similarity function*.

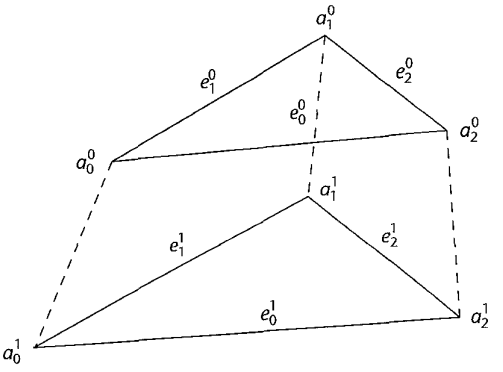
Fuzzy polygon similarity

We know that two polygons $P^0 = \{a_0^0, a_1^0, \ldots, a_n^0\}$ and $P^1 = \{a_0^1, a_1^1, \ldots, a_n^1\}$ are similar if their corresponding angles are equal and corresponding edges are proportional. As an example, Figure 1 shows two similar triangles $a_0^0a_1^0a_2^0$ and $a_0^1a_1^1a_2^1$. In a fuzzy set of polygons,[6] one polygon can be similar to the other in some degree, which is determined by a polygon similarity function. Two polygons are considered quite similar if the degree of similarity determined by the similarity function is high. Likewise, we say that two polygons are quite different if the degree of similarity between them is very low.

We now define a new similarity function of two polygons. The new function represents the degree to which the corresponding angles and edges of the polygons are equal and proportional, respectively.

Observe that two polygons are similar if the pairs of corresponding triangles formed by the corresponding angles of the polygons and the adjacent edges of these angles are similar. Thus we can define the similarity function of two polygons by using the similarity functions of these triangles.

Consider the triangles $a_0^0a_1^0a_2^0$ and $a_0^1a_1^1a_2^1$ in Figure 1. Assume that edges e_1^0, e_2^0 and angle a_1^0 correspond to edges e_1^1, e_2^1 and angle a_1^1, respectively. Then these two triangles are similar if $e_1^0 \times e_2^1 = e_1^1 \times e_2^0$ and $a_1^0 = a_1^1$.



1 Two similar triangles.

If these triangles are not completely similar, then $e_1^0 \times e_2^1 - e_1^1 \times e_2^0 \neq 0$ and/or $a_1^0 - a_1^1 \neq 0$. The similarity degree depends on these differences. Based on these differences, we can define the triangle similarity function as follows:

$$\mathrm{sim}_t = w_1 \times \left(1 - \frac{|e_1^0 \times e_2^1 - e_1^1 \times e_2^0|}{e_1^0 \times e_2^1 + e_1^1 \times e_2^0}\right) + w_2 \times \left(1 - \frac{|a_1^0 - a_1^1|}{360 \text{ deg.}}\right)$$

where $0 < w_1, w_2 < 1$, and $w_1 + w_2 = 1$. The experiments presented later in "Experimental results" set both w_1 and w_2 to 0.5.

The above triangle similarity function has the following properties:

- $0 \leq \mathrm{sim}_t \leq 1$,
- $\mathrm{sim}_t = 1$ if the two triangles are similar,
- sim_t is large if the two triangles are quite similar, and
- sim_t is small if the two triangles are quite different.

Using this triangle similarity function, given a vertex correspondence

$$\mathrm{map} : P^0 \rightarrow P^1$$

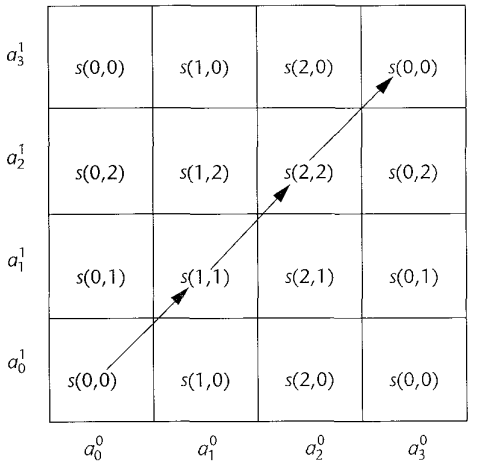
we can define the similarity function of the polygons P^0 and P^1 as follows:

$$\mathrm{sim}_p(P^0, P^1) = \frac{1}{n+1} \times \sum_{i=0}^{n} \mathrm{sim}_t(T_i^0, T_j^1)$$

where $j = \mathrm{map}(i)$, T_i^0 and T_j^1 are corresponding triangles formed by the corresponding angles a_i^0 and a_j^1 and their adjacent edges e_i^0, e_{i+1}^0 and e_j^1, e_{j+1}^1 of the polygons, and $\mathrm{sim}_t(T_i^0, T_j^1)$ is the similarity function of these two corresponding triangles. For convenience, in the following, the phrase *polygon angle* (or angle) refers to the triangle formed by this angle and its adjacent edges.

Function sim_p has the following properties:

- $0 \leq \mathrm{sim}_p(P^0, P^1) \leq 1$,
- $\mathrm{sim}_p(P^0, P^1) = 1$ if the polygons P^0 and P^1 are similar,
- $\mathrm{sim}_p(P^0, P^1)$ is large if P^0 and P^1 are quite similar, and
- $\mathrm{sim}_p(P^0, P^1)$ is small if P^0 and P^1 are quite different.



2 The similarity graph of the polygons in Figure 1, $s(i, j) = \text{sim}_t(a_i^0, a_j^1)$.

So far, it is assumed that polygons P^0 and P^1 have the same number of vertices. Polygons with different numbers of vertices can be handled in the same way.

In this article, a "polygon" means a sequence of ordered vertices of a polygon. Thus, we can obtain multiple polygons from the same polygon by considering different sequences of its vertices. For convenience, the following examples assume that the vertices of a polygon are given in clockwise direction. In this case, n polygons (sequences of ordered vertices) can be generated from a polygon with n vertices. For example, using the target polygon P^1, we can generate $n + 1$ polygons $P^{1,i}$, $i = 0, 1, \ldots, n$, where $P^{1,i} = \{a_i^1, a_{(i+1) \bmod(n+1)}^1, \cdots, a_{(i+n) \bmod(n+1)}^1\}$. Given the source polygon P^0, the set of these polygons and the polygon similarity function sim_p form a fuzzy set $P = <x, \text{sim}_p>$. A polygon X belongs to this set with the membership degree of $\text{sim}_p(P^0, X)$. In other words, X is similar to P^0 with the similarity degree of $\text{sim}_p(P^0, X)$.

Using fuzzy techniques, the issue of establishing the vertex correspondence between the vertices of P^0 and P^1 becomes an issue of finding a polygon in the fuzzy set P that maximizes its membership function sim_p. The next two subsections present a graph approach for obtaining such a solution.

Fuzzy similarity graph

In a manner similar to the physically based blending method of Sederberg et al.,[2] given the polygons $P^0 = \{a_0^0, a_1^0, \ldots, a_m^0\}$ and $P^1 = \{a_0^1, a_1^1, \ldots, a_n^1\}$, $m \geq n$, the fuzzy similarity degrees of the pairs of possible corresponding angles are represented in a graph $G[m + 2, n + 2]$ of $(m + 2) \times (n + 2)$ nodes, where $G[i, j]$, $0 \leq i \leq m + 1$, $0 \leq j \leq n + 1$, represents the fuzzy similarity degree of the angles a_i^0 and a_j^1. Here $G[i, n + 1] = G[i, 0]$ and $G[m + 1, j] = G[0, j]$. As an example, Figure 2 shows the fuzzy similarity graph of the polygons $a_0^0 a_1^0 a_2^0$ and $a_0^1 a_1^1 a_2^1$ in Figure 1. In this article, such a graph is called the *fuzzy similarity graph* (or similarity graph) of the polygons. The cost of a path in the similarity graph is the summation of the similarity degrees associated with the nodes on this path.

Sederberg[2] noted that, in a complete shape transformation, every vertex in P^0 should correspond to at least one vertex in P^1 and vice versa. Observe that the warping method presented here produces better results if every vertex in P^0 corresponds to one and only one vertex in P^1, where it is assumed that P^0 has more vertices than P^1. This condition means that no "north" step is allowed in a path in the graph. As with the physically based blending method, angle a_i^0 can correspond to angle a_j^1 if their adjacent angles correspond, that is, a_{i-1}^0 corresponds to a_{j-1}^1 or a_j^1. This means that a path in the similarity graph should contain only northeast or east steps. Such a path is called a *legal path* in this article. In addition, a path is called a *complete path* if the nodes on the path represent a vertex correspondence of the polygons. For example, the path in Figure 2 is both legal and complete.

We can determine the vertex correspondence that maximizes the similarity function of two polygons by searching for the longest legal complete path in the fuzzy similarity graph of these polygons.

To use a standard graph algorithm given a similarity graph G, consider a graph G^{-1}, where $G^{-1}[i, j] = 1 - G[i, j]$. In this article, G^{-1} is called the complement of G. It is obvious that finding the longest legal complete path in G becomes an issue of finding the shortest legal complete path in G^{-1}. Given a starting node $[i_1, j_1]$ and an ending node $[i_2, j_2]$ in G^{-1}, we can use a standard graph algorithm to determine the shortest legal path that combines these two nodes. The starting and ending nodes are determined by one pair of corresponding angles. For example, if we know a_0^0 corresponds to a_0^1 for polygons P^0 and P^1, then the starting and ending nodes are $[0, 0]$ and $[m + 1, n + 1]$, where $G^{-1}[m + 1, n + 1] = G^{-1}[0, 0]$.

If no pair of corresponding angles is known, then we can let the angle a_0^0 of the polygon P^0 correspond to every angle a_i^1, $i = 0, 1, \ldots, n$, of the polygon P^1. To use the standard shortest path algorithm in this case, we extend $(m + 2) \times (n + 2)$ graph G^{-1} to $(m + 2) \times (2 \times (n + 2) - 1)$ graph EG^{-1}, where $EG^{-1}[i, j] = G^{-1}[i, j \bmod(n + 1)]$, $0 \leq i \leq m + 1$, $0 \leq j \leq 2 \times (n + 1) - 1$. For each of the nodes $[0, j]$, $j = 0, 1, \ldots, n$, the shortest legal complete path combining the nodes $[0, j]$ and $[m + 1, n + 1 + j]$ in EG^{-1} is determined. In this manner, $n + 1$ paths will be determined. In this article, the shortest path among them is used to establish the correspondence between the vertices of the polygons P^0 and P^1.

Fuzzy interpolation

After establishing the correspondence between the vertices of two polygons, the next step in warping one polygon into the other is to interpolate the corresponding vertices[2] and/or edges[3] for generating the inbetweening of these polygons. Sederberg et al.[3] noted that interpolating the lengths and orientations of the polygon edges can avoid shrinkage that normally occurs when only the polygon vertices are linearly interpolated.[2] Their method has two main difficulties. One is that it cannot handle curved shapes, and the other is that it can seriously distort the intermediate polygons for polygons with short edges because the orientation of a short edge cannot be determined reliably. The technique described here avoids these difficulties by using a fuzzy affine transformation technique in conjunction with

normalized local coordinates. This fuzzy interpolation method interpolates both the vertices and orientations of the polygons.

Choosing affine transformation

Recall that three pairs of vertices determine a unique affine transformation. Given two polygons under warping, consider the set of possible affine transformations determined by the possible corresponding triples of the polygon vertices. To choose a good affine transformation, consider a fuzzy set $T = <y, \text{smooth}_t>$ formed by the above set of affine transformations and the function smooth_t of the affine transformations below. For an affine transformation y in this set, $\text{smooth}_t(y)$ represents the "goodness" of this transformation in terms of warping a source polygon into a target polygon smoothly.

Our strategy of determining smooth_t is based on the goodness function smooth_a of a pair of corresponding polygon angles. Given a pair of corresponding angles a_i^0 and a_j^1 of the polygons P^0 and P^1, the following factors of the corresponding pair of triangles (see Figure 3) are considered in defining smooth_a:

- the similarity degree S of the triangles $T_i^0 = a_{i-1}^0 a_i^0 a_{i+1}^0$ and $T_j^1 = a_{j-1}^1 a_j^1 a_{j+1}^1$ in both shape and size,
- the minimum rotation angle R required in transforming T_i^0 into T_j^1, and
- the sum A of the areas of these triangles,

where

$$S = w_1 \times \left(1 - \frac{\sum_{i=0}^{2} |e_i^0 - e_i^1|}{\sum_{i=0}^{2} (e_i^0 + e_i^1)}\right) + w_2 \times \left(1 - \frac{\sum_{i=0}^{2} |a_i^0 - a_i^1|}{180 \text{ deg.}}\right)$$

$0 < w_1, w_2 < 1$, and $w_1 + w_2 = 1$. In the "Experimental results" presented below, both w_1 and w_2 are set to 0.5. R and A can be determined from the vertices of the triangles a_{i-1}^0, a_i^0, a_{i+1}^0 and a_{j-1}^1, a_j^1, a_{j+1}^1. In this article, R is determined by using polar decomposition techniques.[9] In addition, both a_i^0 and a_i^1 must fall inside the range of (0 degrees, 180 degrees) since the vertices a_{i-1}^0, a_i^0, a_{i+1}^0 and a_{j-1}^1, a_j^1, a_{j+1}^1 may be used to form local coordinate systems in the process of interpolation (for details, see "Determining in-betweening of polygons"). Taking this into account, we set smooth_a to zero if the above condition is not satisfied. Otherwise, we use S, R, and A to determine smooth_a as follows:

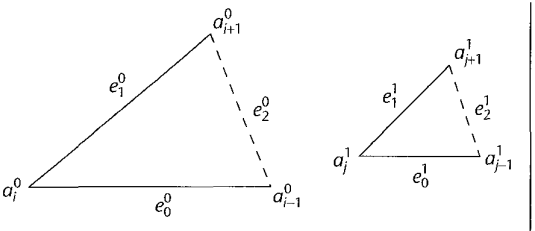
$$\text{smooth}_a(a_i^0, a_i^1) = a \times S + b \times \left(1 - \frac{R}{180}\right) + c \times \frac{A}{A + d}$$

where $a, b, c, d \geq 0$, $a + b + c = 1$. In the experiments presented in this article, the weights a, b, c, and d are chosen as follows: $a = b = 0.4$, $c = 0.2$, and $d = 2$.

Using the goodness function smooth_a, given three pairs of corresponding angles (a_i^0, a_i^1), (a_j^0, a_j^1), and (a_k^0, a_k^1), we define the goodness function smooth_t of the corresponding affine transformation as follows:

$$\begin{aligned}\text{smooth}_t(T(i,j,k)) = {} & \text{smooth}_a(a_i^0, a_i^1) \\ & \times \text{smooth}_a(a_j^0, a_j^1) \\ & \times \text{smooth}_a(a_k^0, a_k^1)\end{aligned}$$

In this article, a good affine transformation (that is, three pairs of corresponding vertices) is chosen from the fuzzy set of good affine transformations by maximizing the affine transformation goodness function smooth_t. Note that the three pairs of corresponding vertices determined in this way are not required to be adjacent.



3 A pair of corresponding polygon angles.

Interpolating affine transformation

We can now consider a method of interpolating the affine transformation that transforms one polygon into another according to the fuzzy technique presented above. We can use the resulting intermediate affine transformations to obtain the in-betweening of the source and target polygons.

Using homogenous matrix representation, an affine transformation can be formulated as follows:

$$(u', v') = (u, v)\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} + (a_{31}, a_{32})$$

In the terminology of geometric transformations, (a_{31}, a_{32}) is a translation vector and

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

is a composite transformation matrix representing rotation, scale, and shear transformations. Here $a_{i,j}$ can be determined from three pairs of polygon vertices.

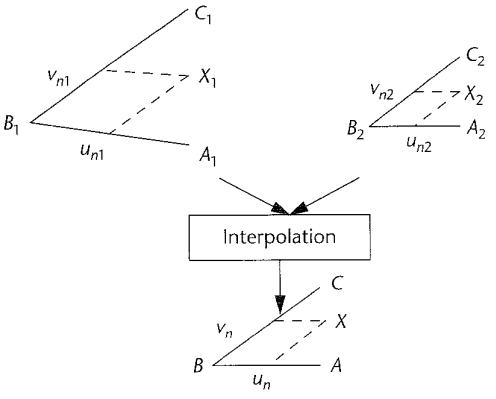
One possible way of interpolating an affine transformation is to interpolate its composite transformation matrix and translation vector linearly as follows:

$$(1-t)\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + t\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

and $t(a_{31}, a_{32})$, where $t \in (0, 1)$.

Shoemake and Duff[9] noticed that this direct matrix interpolation fails to achieve smooth transformation from time to time. Accordingly, they proposed an approach that avoids this difficulty by decomposing the transformation matrix

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$



4 Normalized local coordinates.

into a polar form

$$\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

where

$$\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

is a pure rotation matrix.

Let A, B, and C represent these three matrices. Then given matrix A, matrices B and C can be determined as follows:

$$B = A + \text{sign}(\det(A)) \begin{pmatrix} a_{22} & -a_{21} \\ -a_{12} & a_{11} \end{pmatrix}$$

multiplied by a factor that makes the columns unit vectors, and

$$C = B^{-1}A$$

where $\det(A)$ is the determinant of matrix A and B^{-1} is the inverse of matrix B.

The details of decomposing a matrix into polar form can be found in Shoemake and Duff.[9]

Matrix B can be reformulated as

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

since it is a pure rotation matrix. Combining this representation into the polar form, the composite transformation matrix is interpolated as follows:

$$(1-t)\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} \cos(t\times\theta) & -\sin(t\times\theta) \\ \sin(t\times\theta) & \cos(t\times\theta) \end{pmatrix} \begin{pmatrix} t\times c_{11} & t\times c_{12} \\ t\times c_{21} & t\times c_{22} \end{pmatrix}$$

Interpolation of the translation vector is the same as before. Decomposing the composite transformation matrix into polar form makes it possible to interpolate the orientations of the polygons explicitly.

Determining in-betweening of polygons

The technique presented in this section applies the intermediate affine transformations to a source polygon P^0 to generate in-betweening. This technique was motivated by the work of Beier and Neely,[1] who used vector features to warp gray images. In particular, they used a single vector from each of two images to determine a correspondence between the pixels of these images. For each of these two vectors, they created a perpendicular unit vector. These two orthogonal vectors form a local coordinate system. In this manner each of the two given images is associated with a local coordinate system. In the local coordinate system associated with an image, every pixel of the image has unique local coordinates. Based on these local coordinates, a pixel correspondence can be established by mapping every pixel of one image to the pixel of the other image with the same local coordinates.

Generalizing this idea to the case presented here, we can use the three pairs of corresponding vertices of polygons P^0 and P^1 (see above, "Choosing affine transformation") to form local coordinate systems for these two polygons. Figure 4 shows three pairs of the corresponding vertices (A_1, A_2), (B_1, B_2), (C_1, C_2) and another pair of corresponding vertices (X_1, X_2). For convenience, the following discussion uses uppercase letters to represent pixel coordinates and pairs of pixel coordinates to represent line segments.

Using this representation, we have the following relationship between the local and global coordinates of the vertices X_1 and X_2:

$$X_1 = B_1 + u_1 \times \frac{(A_1 - B_1)}{\|A_1 - B_1\|} + v_1 \times \frac{(C_1 - B_1)}{\|C_1 - B_1\|}$$

$$X_2 = B_2 + u_2 \times \frac{(A_2 - B_2)}{\|A_2 - B_2\|} + v_2 \times \frac{(C_2 - B_2)}{\|C_2 - B_2\|}$$

where (u_1, v_1) and (u_2, v_2) are parameters representing local coordinates. These two equations can be rewritten into two sets of equations. Then the local coordinates (u_1, v_1) and (u_2, v_2) of the vertices X_1 and X_2 can be determined by solving these two sets of equations, respectively.

Observe that the local coordinates change with the size of the edges formed by the vertices. To make the local coordinates independent from the size of the edges, we can normalize the local coordinates as follows:

$$u_{n1} = \frac{u_1}{\|A_1 - B_1\|}, \; v_{n1} = \frac{v_1}{\|C_1 - B_1\|},$$

$$u_{n2} = \frac{u_2}{\|A_2 - B_2\|}, \; v_{n2} = \frac{v_2}{\|C_2 - B_2\|}$$

where $u_{n1}, v_{n1}, u_{n2}, v_{n2}$ are normalized local coordinates.

To interpolate the vertices X_1 and X_2, we can obtain an intermediate affine transformation (see above, "Interpolating affine transformation") and apply it to

the vertices A_1, B_1, and C_1 to generate three intermediate vertices A, B, and C. Let X be the interpolation of X_1 and X_2. We can obtain the normalized local coordinates (u_n, v_n) of X by linearly interpolating the normalized local coordinates of X_1 and X_2, that is, $u_n = (1-t)\times u_{n1} + t\times u_{n2}$ and $v_n = (1-t)\times v_{n1} + t\times v_{n2}$. The location of X can then be determined as follows:

$$u = u_n \times \|A-B\|$$
$$v = v_n \times \|C-B\|$$
$$X = B + u\times\frac{(A-B)}{\|A-B\|} + v\times\frac{(C-B)}{\|C-B\|}$$

Notice that the vertices X_1 and X_2 are an arbitrary pair of corresponding vertices of the given polygons. Thus, applying this method to every pair of the corresponding vertices generates the in-betweening for the given polygons.

This interpolation method can handle all affine transformations: translation, rotation, scale, and shear. In addition, it can be used to warp curved shapes.

In-betweening of curved shapes

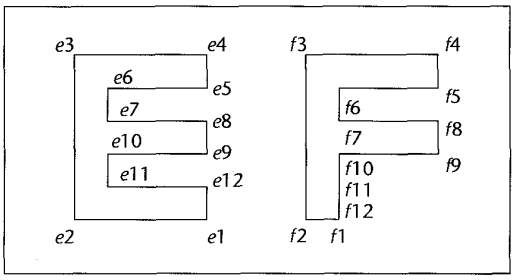
To use this fuzzy method for warping one curved shape into the other, we must specify a pair of corresponding sequences of ordered key points on the boundaries of the given shapes. These two sequences of key points form two polygons on the two given shapes. Thus, we can apply the techniques described in "Fuzzy vertex correspondence" and for "Choosing affine transformation" to determine an appropriate affine transformation for these polygons. Recall that this affine transformation corresponds to a pair of triples of key points on the given curved shapes. The triple of key points on each curved shape forms a local coordinate system for that shape (see "Determining in-betweening of polygons"). In this local coordinate system, each point on the boundary of the shape is associated with local coordinates.

To interpolate a source curved shape and a target curved shape, we must establish a correspondence between the points on the boundaries of these two shapes. Here, we define for each point on the source curved shape a corresponding point on the target curved shape such that the target point's local coordinates are nearest to the local coordinates of the point on the source curved shape.

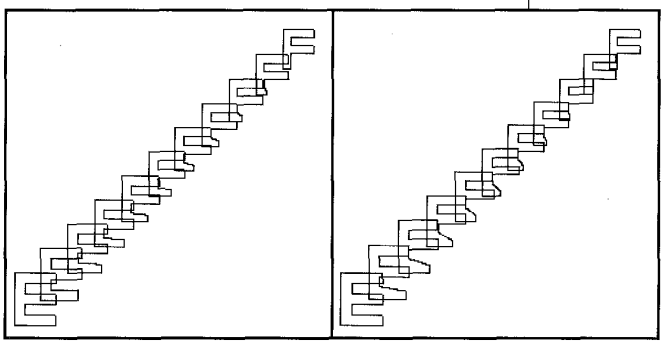
After we determine the affine transformation and point correspondence, we can use the techniques described for "Interpolating affine transformation" to generate the in-betweening of the two given curved shapes.

The essence of this fuzzy warping method is similar to that of free-form deformation techniques.[10] In free-form deformation, control points are used to assign local coordinates to the points on the objects under deformation. To deform an object involves three steps. First, control points are selected and local coordinates are assigned. Then the control points are transformed into a set of new points. Finally, based on these new control points, the local coordinates are used to determine a new object. In free-form deformation, the control points usually do not belong to the objects being deformed. Because of this, the deformation process is hard to control. The approach presented in this article, however, takes control points from the objects being warped. The process is therefore more controllable than normal free-form deformation techniques.

The smooth transformation of control points is an important issue in free-form deformation. The approach presented here offers a good candidate for solving this problem.



5 Two polygons representing E and F.

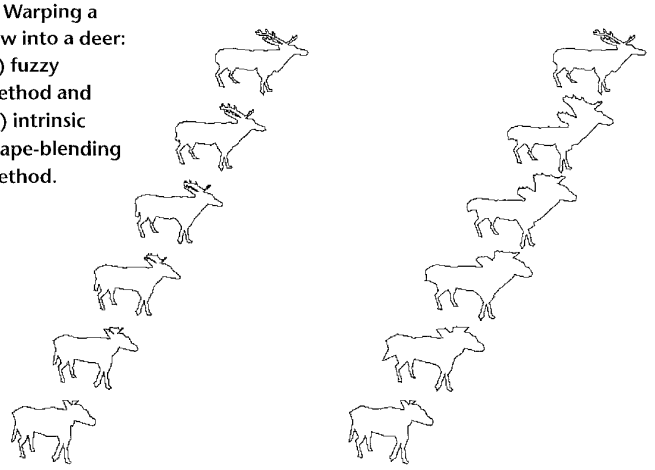


6 The in-betweening between E and F generated by (a) the fuzzy method and (b) the intrinsic shape-blending method.

Experimental results

We can now review some complicated polygons used to evaluate the performance of the fuzzy warping approach presented in this article. These polygons are hand-drawn reproductions of the "difficult" polygonal shapes used by Sederberg and Greenwood to demonstrate their physically based blending method of warping polygons.[2] The emphasis here, however, is on comparing the new method to the more accurate intrinsic shape-blending approach.[3]

Figure 5 shows two polygons representing the letters E and F. The in-betweening produced by using the fuzzy approach is shown in Figure 6a. The fuzzy correspondence technique described earlier is used to establish a vertex correspondence without human intervention. Then the fuzzy transformation technique is used to generate the in-betweening of the polygons. In particular, the method presented in "Choosing afffine transformation" is used to determine an affine transformation that transforms polygon E into F. This affine transformation is interpolated as described in "Interpolating affine transformation." Based on the resulting intermediate affine transformations, the normalized local coordinate technique described in "Determining in-betweening of polygons" is used to produce the inter-



7 Warping a cow into a deer: (a) fuzzy method and (b) intrinsic shape-blending method.

(a) (b)

mediate polygons of E and F.

Figure 6b shows the in-betweening of the letters E and F produced by the intrinsic shape blending method.[3] To make a fair comparison in this and the following experiments, the same correspondences of polygon vertices determined by the fuzzy method are used to generate the in-betweenings of polygons in both the fuzzy warping method and the intrinsic blending method. Comparing Figure 6 shows the fuzzy warping method to be comparable with the intrinsic blending method in warping characters.

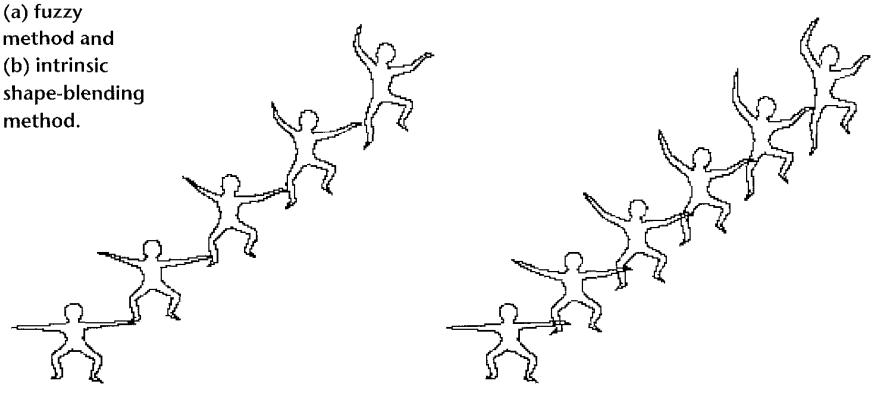
Sederberg and Greenwood[2] noticed that several commercial shape-blending software packages generate highly distorted intermediate shapes in warping E to F. We can observe from Figure 6 that both the fuzzy and the intrinsic blending methods avoid this difficulty.

Generally speaking, two polygons with very different shapes require human intervention to establish a reasonable vertex correspondence. The fuzzy approach requires specification of only one pair of corresponding vertices to establish the vertex correspondence, so Figures 7 through 9 were generated in this way. Figures 7a and 7b show the warping of a cow into a deer by the fuzzy method and the intrinsic blending method, respectively. Figures 8a and 8b show the results of warping between two dance positions by these methods. Figures 9a and 9b show the in-betweenings of two chickens.

The polygons representing humans consist of long edges and the polygons representing cows and chickens have short edges. We can see from Figure8 that the fuzzy and intrinsic blending methods produce comparable results in the case of polygons with long edges. For polygons with short edges, however, the in-betweening produced by the intrinsic blending method can be highly distorted, as we can see in Figures 7b and 9b. As noted earlier, the main reason for this distortion is that the orientation information for short edges cannot be determined reliably, which leads in turn to wrong polygon angle information. The fuzzy approach avoids this difficulty, as the results in Figures 7a and 9a show.

The other disadvantage of the intrinsic blending technique is that it cannot handle curved shapes—a disadvantage shared by physically based shape-blending methods. In contrast, the fuzzy method presented in this article can be extended to curved shapes if two corresponding sequences of ordered key points on the shape boundaries can be specified. For example, Figure 10 shows the in-betweening of a circle and a hexagon produced using the techniques described earlier (see "In-betweening of curved shapes"). Three pairs of corresponding points are manually specified on the circle and the hexagon. Other than that, the process of warping proceeds automatically.

The fuzzy vertex correspondence technique is similar to the physically based blending method[2] in that the vertex correspondence is established by finding a minimum-cost path in a graph. This cost is determined by a function, and the main difference between these two approaches is in the cost function. The physically based blending method assumes that the corresponding vertices of the two polygons under warping are linearly interpolated. Thus, it may not work well for two polygons with different orientations. The fuzzy membership function removes the assumption of linear interpolation by considering the geometric similarity of the polygons. Thus, it avoids the difficulty with different orientations.



8 Warping one dancing movement into another: (a) fuzzy method and (b) intrinsic shape-blending method.

(a) (b)

Recall that interpolating only the vertices of the polygons can cause shrinkage.[3] The intrinsic blending method[3] avoids this difficulty by interpolating the lengths and orientations of the polygon edges. The fuzzy interpolation technique, however, avoids this difficulty in a different way. Rather than interpolating polygon edges, it interpolates both the vertices and orientations of the polygons. Thus, it avoids the distortions of intermediate polygons that can occur with the intrinsic blending method.

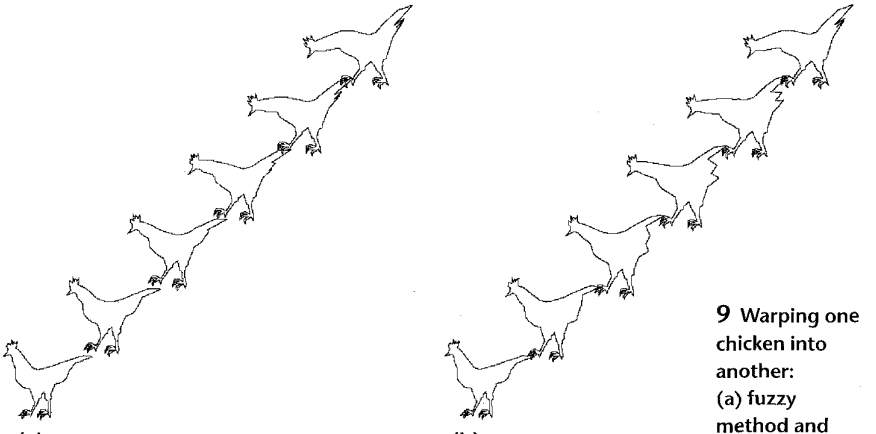
In summary, experimental results show that the fuzzy warping tech-

nique is more robust and generic than the intrinsic shape-blending method in that it avoids distortions and can warp curved shapes.

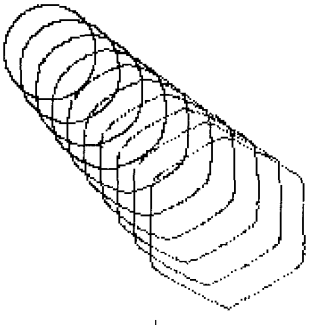
Conclusion

The fuzzy warping method presented here smoothly transforms a source polygon into a target polygon. The method uses a fuzzy vertex correspondence technique to establish the correspondence between the vertices of the source and target polygons and a fuzzy transformation technique to interpolate both the vertices and orientations of the polygons to produce the in-betweening of them. The fuzzy vertex correspondence method establishes the correspondence of the source and target polygons by maximizing the membership function (a polygon similarity function) of a fuzzy set of polygons that are similar to the source polygon. Similarly, the fuzzy transformation method determines an affine transformation by maximizing the membership function of a fuzzy set of good affine transformations. Normalized local coordinates are used to interpolate the corresponding vertices.

This fuzzy method is similar to the physically based blending method of Sederberg et al.[2] The fuzzy transformation technique can be viewed as a natural extension to Beier and Neely's metamorphosis technique.[1] It is also similar to free-form deformation techniques[10] in that in-betweening is produced by interpolating control points. The main difference between the fuzzy approach and normal free-form deformation techniques is that, in the fuzzy approach, the control points belong to the objects to be warped. This has an advantage of making the warping process more controllable. It also makes the warping approach presented here a good candidate for ensuring that the control points used in deforming free-form objects are interpolated appropriately. ■



9 Warping one chicken into another: (a) fuzzy method and (b) intrinsic shape-blending method.



10 Warping a circle into a hexagon by the fuzzy method.

Acknowledgments

I would like to thank Robert E. Webber and Sheng Yu for their proofreading and discussions about fuzzy techniques. I also thank the referees for their instructive comments.

References

1. T. Beier, "Feature-Based Image Metamorphosis," *Computer Graphics* (Proc. Siggraph), Vol. 26, No. 2, 1992, pp. 35-42.
2. T.W. Sederberg, "A Physically Based Approach to 2D Shape Blending," *Computer Graphics (Proc. Siggraph)*, Vol. 26, No. 2, 1992, pp. 25-34.
3. T.W. Sederberg et al., "2D Shape Blending: An Intrinsic Solution to the Vertex Path Problem," *Computer Graphics (Proc. Siggraph)*, 1993, ACM, New York, 1993, pp. 15-18.
4. A. Kalvin et al., "Two-Dimensional Model-Based Boundary Matching Using Footprints," *Int'l J. Robotics Research*, Vol. 5, No. 4, 1986, pp. 38-55.
5. P.W.M. Tsang, P.C. Yuen, and F.K. Lam, "Classification of Partially Occluded Objects Using 3-Point Matching and Distance Transformation," *Pattern Recognition*, Vol. 27, No. 1, 1994, pp. 27-40.
6. H. Bremermann, "Pattern Recognition by Deformable Prototypes," in *Structural Stability, the Theory of Catastrophes and Applications in the Sciences*, Springer Notes in Mathematics, Springer-Verlag, Berlin and New York, Vol. 25, 1976, pp. 15-57.
7. H. Fuchs, Z.M. Kedem, and S.P. Uselton, "Optimal Surface Reconstruction from Planner Contours," *Comm. ACM*, Vol. 20, No. 10, 1977, pp. 693-702.
8. H.N. Christiansen and T.W. Sederberg, "Conversion of Complex Contour Line Definitions," *Computer Graphics* (Proc. Siggraph), Vol. 12, No. 3, 1978, ACM Press, New York, pp. 187-192.
9. K. Shoemake and T. Duff, "Matrix Animation and Polar Decomposition," *Graphics Interface 92*, Canadian Information Processing Soc., Toronto, Ontario, Canada, 1992, pp. 258-264.
10. T.W. Sederberg and S.R. Parry, "Free-Form Deformation of Solid Geometric Models," *Computer Graphics* (Proc. Siggraph), Vol. 20, No. 4, 1986, pp. 151-160.

Yuefeng Zhang *has worked for Honeywell Hi-Spec Solutions since 1993. His main research interests are in computer graphics and image processing. He received a PhD in computer science at the University of Western Ontario in 1993.*

Readers may contact Zhang at Software Technology Group, Honeywell Hi-Spec Solutions, 343 Dundas St., Suite 100, London, Ontario, Canada N6B 1V5, e-mail yuefeng.zhang@ontario.honeywell.com.