EUROGRAPHICS 2002 / I. Navazo Alvaro and Ph. Slusallek
(Guest Editors)

Interpolating 2D Shape Hierarchically

Henry Johan and Tomoyuki Nishita

Department of Computer Science, The University of Tokyo, Japan

Abstract

Shape interpolation has been widely used in the field of computer graphics for modeling and for creating visual effects. This paper presents a novel hierarchical method to interpolate between two 2D shapes. A hierarchical representation, which is a hierarchy of triangles, is proposed to represent the interior and the details of each shape. By constructing the compatible hierarchical representations of the two shapes, the intermediate shapes are computed by interpolating the corresponding triangles at the lowest level to the highest level of the representations, From experimental results, the proposed method produces smooth interpolation sequences.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling - Hierarchy and geometric transformations

1. Introduction

Shape interpolation is a technique used to generate a sequence of intermediate shapes that transform a source shape into a target shape. Combining the shape interpolation technique with a morphing technique provides a powerful tool for creating visual effects. Nowadays, these techniques have been used widely in motion-pictures, television, and other commercial images. There are two main problems in shape interpolation, the vertex correspondence and the vertex path problems. Vertex correspondence is the problem of establishing the correspondences between the vertices of the two shapes, while vertex path is the problem of determining the positions of the vertices during the interpolation.

This paper presents a novel solution to the 2D vertex path problem. A smooth interpolation between two shapes can be realized by first interpolating between their interiors and then gradually adding the details. For this purpose, we introduce a new *"hierarchical representation"* which is used to represent the interior and the details of a shape. Hierarchical representation of a shape is defined as a hierarchy of triangles used to describe the shape. This hierarchy of triangles can be represented as a tree, and we call it *"hierarchical tree"*. The basic idea of this representation is that, we start from a single triangle, then by adding triangles recursively, we reconstruct the original shape.

If two hierarchical representations have the same hierarchical tree, we say that the two hierarchical representations are compatible, and we call them *"compatible hierarchical representations (CHR)"*. Note that in their compatible representations, each triangle in the source representation has exactly one corresponding triangle in the target representation. Intermediate shapes are computed by interpolating the corresponding triangles at the lowest level to the highest level of the compatible hierarchical representations, that is, adding the details (vertices) gradually.

The advantages of using hierarchical approach are that the interpolation starts from the interior, then gradually adds the details, resulting in smooth interpolation sequence, also the features of the source and target shapes are guaranteed to be preserved during the interpolation. For example, if the source and target shapes have bumpy boundaries, then the generated intermediate shapes will also have bumpy boundaries. The main contributions of this paper are: (1) a novel hierarchical representation of a shape and a method to construct the compatible hierarchical representations of two shapes, (2) an algorithm to interpolate the compatible hierarchical representations of two shapes.

This paper is organized as follows. Section 2 describes previous work on the vertex path problem. Section 3 explains the definition of compatible hierarchical representations. Section 4 describes the algorithm to construct the compatible hierarchical representations of two shapes. Section 5 discusses our method to interpolate these compatible hierarchical representations, and Section 6 shows the results of our algorithm. The final section concludes this paper.

2. Previous Work

The simplest solution to the vertex path problem is achieved by linearly interpolationg the coordinates of the two corresponding vertices. However, this approach suffers from the problem of shrinkage. This problem occurs frequently especially when the shapes undergo rigid body motion.

Sederberg et al.[9] proposed an algorithm that interpolates the edge lengths and the vertex angles of the two polygons. However, this method tends to distort the area of the intermediate polygons. Methods using multi-resolution representations were proposed by Goldstein and Gotsman[4], Zhang and Huang[15]. Goldstein and Gotsman use the polygon evolution scheme, while Zhang and Huang use a wavelet approach to construct the multi-resolution representations.

Carmel and Cohen-Or[2] and Zhang[14] compute the vertex path with a transformation that consists of a rigid part and an elastic part. In the rigid part of the transformation, rotation and translation are performed, while in the elastic part, the corresponding vertices are linearly interpolated. The shortcoming of their approaches is that a single rotation and translation are usually not enough to generate a smooth interpolation. Samoilov and Elber[8] proposed methods for eliminating self-intersection when interpolating between freeform curves. However, according to Surazhsky and Gotsman[11], their methods cannot guarantee that the resulting interpolation will be self-intersection free.

All of the methods described above take into account only the boundaries of the polygons. Shapira and Rappoport[10] noticed that the interiors of the polygons play an important role in the interpolation process. They express the interiors of the polygons by using the *"star-skeletons"* representation. If the two polygons differ significantly, however, it is quite ambiguous to construct the corresponding skeletons.

Recent research publications such as Tal and Elber[13], Floater and Gotsman[3], Alexa et al.[1], Gotsman and Surazhsky[5], and Surazhsky and Gotsman[11, 12] have dealt with the problem of interpolating two compatible triangulations. Alexa et al. proposed an excellent method to interpolate between two triangulations. For each pair of corresponding triangles, the ideal affine transformation is defined. The global transformation is performed such that its minimizes the local deformation. In the methods of Surazhsky and Gotsman, the original polygons are first enclosed by two convex polygons. Then the new polygons are compatible triangulated, after which the triangulations are morphed using the method proposed by Floater and Gotsman[3].

The methods proposed by Surazhsky and Gotsman guarantee that the intermediate shapes will be free from self-intersection. However, since their methods use polygons to enclose the original shapes and perform the interpolation between these enclosed polygons, the computed intermediate shapes can exhibit area distortion if the source and target shapes differ significantly.

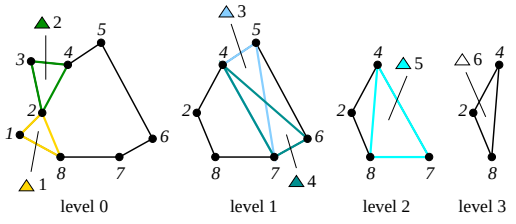


Figure 1: *Hierarchical representation of a shape.*

3. Compatible Hierarchical Representations

A hierarchical representation of a shape is defined as a set of triangles used to describe the shape with hierarchical relationships defined among the triangles. The basic idea of the hierarchical representation is similar to that of building-bricks, starting from a single triangle, then by adding triangles recursively, the original shape is reconstructed. Figure 1 shows an example of a hierarchical representation. Each level contains triangles which represent some of the details of the shape at that particular level. In the example shown in Figure 1, triangle 248 is the triangle at the coarsest level of the representation. The shape at the next higher level (level 2) is created by adding triangle 478 to triangle 248. Triangle 478 represents vertex 7 which does not exist in the shape at the next lower level (level 3). By adding triangles recursively, the original shape is reconstructed. Note that this representation is not a triangulation. For instance, triangle 457 and triangle 467 overlap each other in the representation.

We use the following terms to refer to the vertices of a triangle: *"center vertex"* and *"base vertices"*. Center vertex is the vertex which is represented by the triangle and base vertices are the rest vertices of the triangle. Edge which connects the two base vertices is called *"base edge"*. For instance, triangle 478 represents vertex 7, and thus vertex 7 is the center vertex, vertices 4, 8 are the base vertices, and edge 48 is the base edge. All triangles at the same level are assigned the same level number. The triangles at the highest level of representation have level number 0. Triangles in the next lower level have level number 1 and so on. The lowest level of the representation consist of a single triangle.

Except the triangle at the lowest level of the representation, every triangle has one *"parent triangle"*. The parent triangle is defined as follows. Let v_i, v_j, v_k be the vertices of triangle T. Assume that v_i and v_k are the base vertices. There are two triangles, a triangle with v_i as its center vertex and a triangle with v_k as its center vertex, which are suitable candidates for the parent triangle of T. For instance, triangles 5 and 6 in Figure 1 are the candidates for the parent of triangle 3. From these two triangles, the triangle which has its level number closest to the level number of T is chosen as the parent triangle. In the case of triangle 3, triangle 5 becomes its parent triangle.

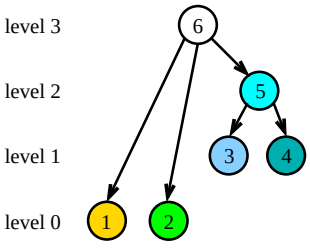


Figure 2: *The hierarchical tree for the hierachical representation shown in Figure 1.*

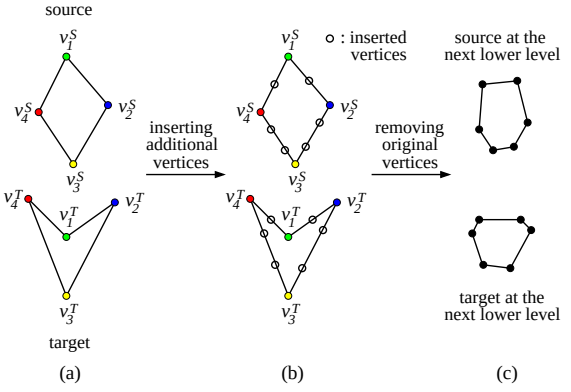


Figure 3: *Constructing the compatible hierarchical representations of (a) the source and target shapes by (b) inserting additional vertices, (c) removing some of the original vertices and creating the source and target shapes at the next lower hierarchical level.*

Hierarchical representation can be expressed by using a tree. The nodes of the tree represent the triangles in the representation. In particular, the root of the tree represents the triangle at the coarsest level of the representation. The edges of the tree represent the parent-child relationships (hierarchy) between the triangles in the representation. We call this tree a *"hierarchical tree"*. Figure 2 shows the hierarchical tree of the hierarchical representation in Figure 1. Given two shapes A and B. Assume that H_A and H_B are the hierarchical representations of shapes A and B, respectively. If the hierarchical trees of these two representations are the same, then we say that H_A and H_B are compatible hierarchical representations. Note that if H_A and H_B are compatible, then there exists a one-to-one correspondence between the triangles in H_A and H_B.

4. Construction of Compatible Hierarchical Representations

We can build many types of compatible hierarchical representations of two shapes. For the purpose of shape interpolation, the goal is to build representations which express the corresponding details and interiors of two shapes at a higher level and a lower level of the representations, respectively. The interiors of the given shapes can be obtained by recursively removing the details.

We assume that the source and target shapes have the same number of vertices and there exist one-to-one correspondence relationships between the vertices of the two shapes. The outline of the compatible hierarchical representations construction is as follows (see Figure 3). First, we insert additional vertices into the source and target shapes (Section 4.1), creating new source and target shapes. Then we search for vertices to be removed. After that we perform the actual vertex removal operation (Section 4.2), create triangles to represent the removed vertices, and finally create the source and target shapes at the next hierarchical level (Section 4.3). We repeat these operations until the shapes at the next hierarchical level are either triangles or lines. We treat lines as degenerate triangles.

4.1. Inserting additional vertices

If the original shapes are represented by small numbers of triangles (the size of triangles are relatively large compared to the size of the original shapes), and the source and target shapes differ significantly, natural interpolation cannot be achieved since the trajectories of the vertices during the interpolation are restricted. Thus, it is important that the number of triangles are sufficient to express the differences between the source and target shapes. By inserting new vertices in the edges of the source and target shapes, smaller triangles can be generated. As a result, the number of triangles in the representations are increased.

Before we insert any vertices, the values l_{src_max} and l_{tgt_max} are determined to restrict the edge lengths of the triangles in the source and target representations, respectively. Let l_{src} be the average edge length of the source shape. We define $l_{src\_current} = \gamma \cdot l_{src}$. The value γ is chosen depending on the degree of similarity between the original source and target shapes. If the two shapes are similar, we set γ to be a large value, otherwise we set it to be a small value. In our experiment, γ is set between 0.5 to 2.0. Let l_{src_upper} be the value of l_{src_max} at the previous upper level. l_{src_max} is computed using the following equation,

$$l_{src\_max} = \max(l_{src\_current}, l_{src\_upper}). \qquad (1)$$

By considering the value of l_{src_upper}, we avoid inserting new vertices over and over again and thus guarantee that the algorithm will terminate. The above computation is also performed on the target shape, yielding l_{tgt_max}.

We define v_i and v_{i+1} as the end points of the i-th edge. We also assume that θ_i and θ_{i+1} are the internal angles at v_i and v_{i+1}, respectively. We do not insert a vertex into the i-th edges if either one of the following conditions is satisfied.

1. The lengths of the i-th edges in the source and target shapes are smaller than l_{src_max} and l_{tgt_max}, respectively.
2. $|\pi - \theta_i| < \varepsilon_\theta$ and $|\pi - \theta_{i+1}| < \varepsilon_\theta$ are satisfied in both the source and target shapes.

The second condition is used to avoid inserting vertices into a region which is nearly a straight line. From experimental results, setting ε_θ to 15° is sufficient to deciding whether an edge lies on a flat region or not. If the above conditions are not satisfied, we insert vertices in the i-th edges. Let l_i^{src} and l_i^{tgt} be the edge lengths of the i-th edges in the source and target shapes, respectively. We define $\eta = \max(floor(l_i^{src}/l_{src\_max}), floor(l_i^{tgt}/l_{tgt\_max}))$. If the value of η is one, we insert a new vertex at the middle of the i-th edges (parameter $u = 0.5$) . Otherwise, we insert two vertices, one at $u_1 = 1.0/(\eta+1)$ and another one at $u_2 = 1.0 - u_1$, of the i-th edges. In the following subsections, we will use the term *"original vertices"* to refer to the vertices that belong to the original shapes and the term *"inserted vertices"* to refer to the vertices inserted at this stage.

4.2. Removing some of the original vertices

The next step is to remove some of the original vertices which represent the details in the source and target shapes. For each original vertex v, we assign a *"detail value"* D_v which is a pair of two values (α, β). These values are determined as follows. Assume that θ_S and θ_T are the internal angles at v in the source and target shapes, respectively. Let $d_{max} = 2\pi - \min(\theta_S, \theta_T)$, $d_{min} = 2\pi - \max(\theta_S, \theta_T)$.

1. If either θ_S or θ_T is smaller than π, then $D_v = (d_{max}, d_{min})$.
2. If θ_S and θ_T are larger than or equal to π, then $D_v = (d_{min}, d_{max})$.

Let $D_{v1} = (\alpha_1, \beta_1)$, $D_{v2} = (\alpha_2, \beta_2)$. D_{v1} is smaller than D_{v2} if and only if either $\alpha_1 < \alpha_2$, or $\alpha_1 = \alpha_2$ and $\beta_1 < \beta_2$. That is, the lexicographic order is used to compare between two detail values. Note that vertices whose detail values are relatively large compared to other vertices tend to represent fine details at the source and target shapes.

Vertices to be removed are determined by searching for the vertices with the largest detail values. These vertices are put together in one list called CANDIDATE. Then, we try to remove the vertices in this list (Section 4.2.1). The removed vertices are represented by using triangles (Section 4.2.2). These triangles are the triangles of the hierarchical representations. If we cannot remove any of the vertices in list CANDIDATE, then we search for vertices with the next largest detail values. We repeat the search until we find vertices that can be removed.

In the case when none of the original vertices can be removed, the value of l_{src_max} and l_{tgt_max} is halved and the vertex insertion step (Section 4.1) is repeated. By inserting new vertices near the original vertices, it is guaranteed that we can always find vertices that can be removed since the possibility of causing self-intersections and folding problems (i.e., the ordering of the vertices are reversed) is decreased when we remove those original vertices (refer to Section 4.2.1).

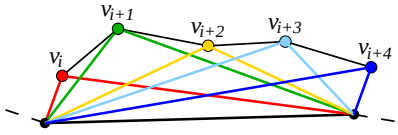


Figure 4: *Removing many vertices simultaneously.*

4.2.1. Removing the vertices in list CANDIDATE

We group the vertices in the list into groups which consist of successive vertices. For example, if the shape has nine vertices $(v_1, ..., v_9)$, and list CANDIDATE contains $\{v_2, v_3, v_4, v_7\}$, then we can group the vertices in the list into two groups, which are group of $\{v_2, v_3, v_4\}$ and group of $\{v_7\}$. Then we process these groups independently. If a group consists of one vertex, we check if the vertex can be removed. A vertex can be removed if its removal does not cause the source and target shapes to exhibit self-intersection and does not cause the folding problem. If a group has two vertices, we check if we can remove these two vertices simultaneously without causing the self-intersection and the folding problems. By removing the two vertices simultaneously, we guarantee that if the source and target shapes are symmetric, then the shapes at all hierarchical levels are also symmetric.

Next, we consider the case of a group G which consists of more than two vertices. We assume that $v_i, v_{i+1}, ..., v_{i+k}$ are those vertices. If all these vertices are removed simultaneously, relatively large triangles most likely will be needed to represent these vertices (see Figure 4). However, as we mentioned previously in Section 4.1, large triangles are not desireable when computing the intermediate shapes. Therefore, when G consists of more than two vertices, we check its vertices as follows. Depending on the number of vertices in the group (equal to $k+1$), we define list G_1 as follows.

1. If $(k+1)$ is an odd number, we define $G_1 = \{v_i, v_{i+2}, v_{i+4}, ..., v_{i+k}\}$.
2. If $(k+1)$ is an even number and $(k+1)/2$ is an odd number, we define $G_1 = \{v_i, v_{i+2}, ..., v_{i+(k-1)/2}, v_{i+(k+1)/2}, ..., v_{i+k-2}, v_{i+k}\}$.
3. If $(k+1)$ is an even number and $(k+1)/2$ is an even number we define $G_1 = \{v_i, v_{i+2}, ..., v_{i+(k-3)/2}, v_{i+(k+3)/2}, ..., v_{i+k-2}, v_{i+k}\}$.

Next, we define list $G_2 = G - G_1$. Note that the vertices in G_1 and G_2 can be grouped into smaller groups which consist of one or two vertices. We first check if we can remove the vertices in G_1. If we cannot find at least one vertex that can

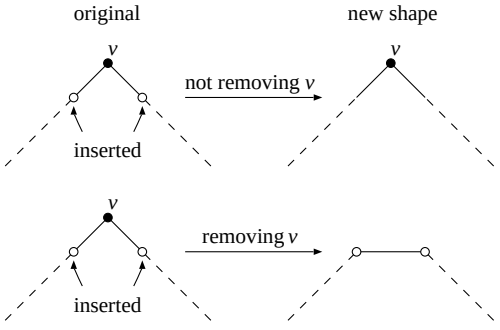


Figure 5: *Determining vertices of the new shape at the next hierarchical level.*

be removed, then we check the vertices in G_2.By using this approach, we avoid removing more than two vertices simultaneously and thus avoid generating relatively large triangles in the representations.

4.2.2. Representing the removed vertices

To represent the removed vertices, we use triangles. The removed vertex is the center vertex of the triangle and its two neighbor vertices are the base vertices of the triangle. In the case when we remove two consecutive vertices, their two neighbor vertices are used as the base vertices. For instance, in Figure 1, vertices 4 and 7 are the base vertices when we remove vertices 5 and 6.

4.3. Lower level shape

After removing some of the original vertices, the next step is to create the source and target at the next lower level. The shape at the next lower level is composed of original vertices which are not removed and the inserted vertices which are the direct neighbors of the removed vertices (see Figure 5). Figures 13 (a) and (c) show the source and target shapes at some levels of the compatible hierarchical representations, respectively. As shown in this figure, the details of the shapes are removed gradually as the algorithm proceeds.

5. Interpolation between Two Compatible Hierarchical Representations

The intermediate shapes are computed hierarchically, that is, interpolating the triangles at the coarsest level, then gradually adding the details at each level of the representations.

The outline of the interpolation method is as follows (see Figure 6). First, we interpolate the triangles (Section 5.1) at the lowest level of the hierarchical representations. Then, we move to the next higher level, interpolate all the triangles at this level, and finally we compute the shape at this level (Section 5.2). We perform these operations until we have finished processing the triangles at the highest level of the representations.

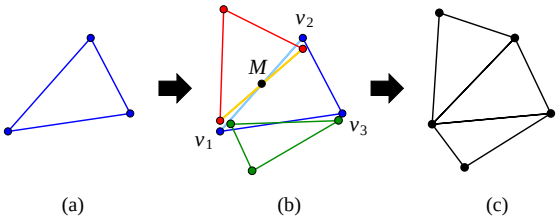


Figure 6: *Interpolating the compatible hierarchical representations. (a) Interpolate the lowest level triangles, (b) interpolate the triangles at the next higher level, and (c) compute the final shape at this level.*

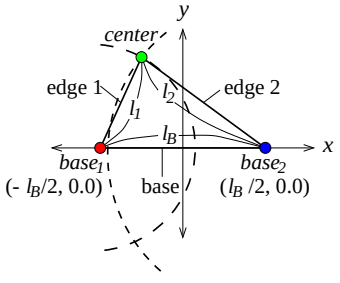


Figure 7: *Computation of ideal intermediate triangle.*

5.1. Intermediate triangle

To compute an intermediate triangle, we first determine its ideal shape (Section 5.1.1), then compute the orientation of its base edge (Section 5.1.2), and finally compute its location (Section 5.1.3).

5.1.1. Ideal intermediate triangle

We define an ideal transformation between two triangles as a transformation that linearly changes the lengths of the edges. By using this approach, the intermediate shapes can be generated without causing shrinkage. To compute the ideal intermediate triangle at a time t, we define a local coordinate system with the base edge of the triangle as the x-axis (see Figure 7). Since the lengths of the edges change linearly, we first compute the length l_B of the base edge, and then the coordinates of the base vertices are given by $(+/-l_B/2, 0.0)$. The coordinates of the center vertex can be computed as an intersection between two circles centered at $base_1$ and $base_2$, having radii of l_1 and l_2, respectively (see Figure 7). l_1 and l_2 are the lengths of edge 1 and edge 2 at time t. If the y coordinates of the source and target triangle are equal to or larger than zero, then the y coordinate of the center vertex is equal to or larger than zero. Otherwise, the y coordinate is smaller than zero.

There is a special case if the source and target triangles have different orientations (see Figure 8). First, we check

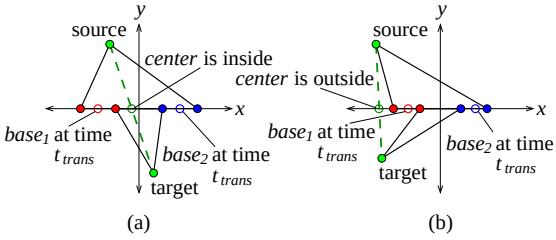


Figure 8: *Interpolation between triangles with different orientations. (a) The case when linearly interpolating the center vertex is adequate and (b) the case when we cannot simply linearly interpolate the center vertex.*

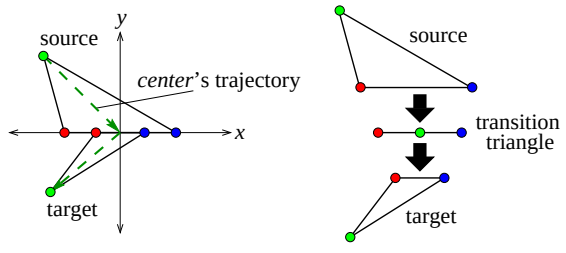


Figure 9: *Interpolation using transition triangle.*

whether it is sufficient to linearly interpolate the center vertex. We compute the time t_{trans} when the center reaches the x-axis. Then we compute the coordinates of the base vertices at time t_{trans}. If the center vertex is located inside the base edge of the triangle, then we use linear interpolation to interpolate the center vertices (Figure 8(a)). If the center vertex is located outside the base edge (Figure 8(b)), we cannot simply linearly interpolate the center vertices since the resulting intermediate shape will exhibit self-intersection. In such a case, we first transform the center of the source triangle to the origin of the local coordinate system, and then we transform from the origin to the center of the target triangle using linear interpolation. This operation can be viewed as transforming the source triangle into a transition triangle which is a line, and then transforming the transition triangle into the target triangle (see Figure 9).

5.1.2. Orientation of the base

If the interpolation involves rotations, then the rotations can be well expressed by rotating the intermediate triangles. The angle of rotation can be determined by computing the orientation of a triangle relative to the orientation of a *"reference triangle"*. Since the intermediate shapes are built gradually, the triangles at a particular level can be used as the reference triangles for the triangles in the higher level. As mentioned in Section 3, except the triangle at the lowest level of the representation, all triangles have one parent triangle. In fact, the parent triangle is the best choice for the reference triangle.

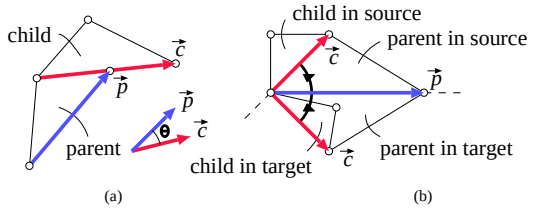


Figure 10: *(a) Angle computation between the base edge of a triangle and its reference axis and (b) rotate a triangle such that the total rotation angle is minimum.*

The details of the algorithm are as follows. For each triangle, the base edge of its parent triangle is used as a reference axis. For the triangle at the lowest level, we use the x-axis of the global coordinate system (the coordinate system where the original source and target are defined) as the reference axis. We then compute the angle between the base edge of the triangle and its reference axis (see Figure 10). In Figure 10(b), we have chosen the smallest angle possible, since we want to avoid local self-intersection. By linearly interpolating this angle, we determine the orientation of the base edge with respect to the reference axis. We add the orientation of the reference axis to this value in order to compute the correct orientation of the base in the global coordinate system. Note that we have already processed the parent triangle and therefore we know the orientation of its base edge in the global coordinates system.

5.1.3. Location of the triangle

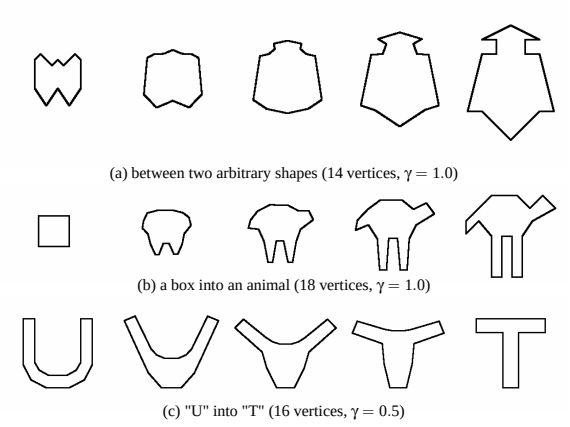
There exists one edge, that belongs to the shape at the previous level, which corresponds to the base edge of the triangle. The computed ideal intermediate triangle is placed such that the middle point of its base edge (point M in Figure 6) coincides with the middle point of such edge.

5.2. Intermediate lower level shapes

In the hierarchical representation, a vertex v can be the vertex of more than one triangle. The vertices v_1, v_2, and v_3 in Figure 6 are the examples of these. Assume that vertex v is the vertex of m triangles. We compute the coordinates of v at a time $t, (0.0 \leq t \leq 1.0)$ by using the following equation.

$$v(t) = \frac{\sum_{i=1}^{m} w_i \cdot v^i(t)}{\sum_{i=1}^{m} w_i}. \qquad (2)$$

$v^i(t)$ $(i = 1, ..., m)$ are the coordinates of v in the i-th triangle at time t. w_i is the weight of i-th triangle. In order to produce a smooth interpolation, if the corresponding triangles in the source and target representations are similar, then their intermediate triangles must not suffer great distortions. Therefore, the weight of such a triangle should be large. To realize this, weight w_i is computed as follows. A *"triangle*



(a) between two arbitrary shapes (14 vertices, $\gamma = 1.0$)

(b) a box into an animal (18 vertices, $\gamma = 1.0$)

(c) "U" into "T" (16 vertices, $\gamma = 0.5$)

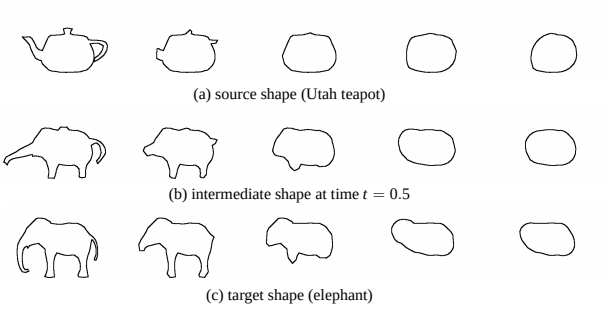
Figure 11: *Examples of interpolation.*

difference value" of the *i*-th triangle is defined as the maximum value of differences between corresponding angles of the *i*-th triangles in the source and target representations. If the two triangles have different orientations, the difference value is the sum of the difference value between the source and the transition triangles and the difference value between the transition and the target triangles. We use the reciprocal of this value as the weight w_i. Note that when the triangles in the source and target representations are similar, then their triangle difference value is small, and thus the weight becomes large. In the implementation, a small value is added to the triangle difference value before we compute the reciprocal in order to avoid division by zero.

The coordinates of the vertices can be computed efficiently by processing each triangle at this level instead of processing each vertex independently. This is done by using a *"coordinates accumulation buffer"* and a *"weight buffer"*. The coordinates accumulation buffer and the weight buffer have entries for all the vertices. For each triangle at this level, we compute the coordinates of its vertices at a time t, add their weighted coordinates to the accumulation buffer, and add their weights (the weight of the triangle) into the weight buffer. After we have finished processing all the triangles at this particular level, we compute the coordinates of the vertices of the shape at this level by dividing the values in the accumulation buffer by the values in the weight buffer.

6. Examples

Before we perform the interpolation process, the user first establish the correspondences between the vertices of the two given shapes. We let the user to specify the correspondences in order to make the interpolation sequence looks natural and aesthetic. For example, for the interpolation shown in Figure 12(b), we want to make sure that the spout of the teapot goes to the trunk of the elephant, the handle of the teapot goes to the tail of the elephant.



(a) source shape (Utah teapot)

(b) intermediate shape at time $t = 0.5$

(c) target shape (elephant)

Figure 13: *Intermediate shapes at different levels of the hierarchical representations.*

Figure 14: *Interpolation from letter "U" to letter "S".*

Figures 11 and 12 show examples of interpolation sequences. In some of the examples, some parts of the shapes undergo rotations. This means that linear interpolation cannot produce smooth interpolation sequences for these examples. In contrast, our method has produced smooth interpolation sequences for all these examples. Figure 13 shows the intermediate shapes at different levels of the hierarchical representation for the interpolation shown in Figure 12(b).

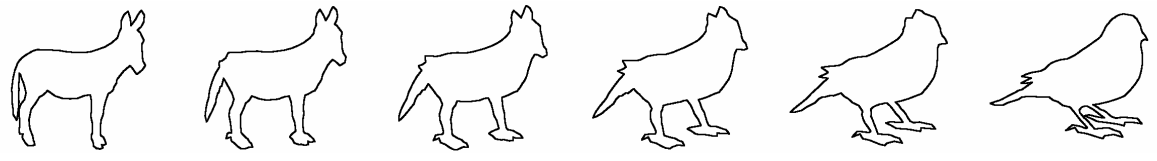
From our experimental results, the proposed algorithm is fast. The compatible hierarchical representations can be computed in less than 0.4 second in all the examples. The intermediate shapes can be generated fast. We can generate more than 60 intermediate shapes per second for all the examples shown in this paper. The computation is performed on machine with Pentium III 800Mhz running Linux.

6.1. Limitation of the proposed method

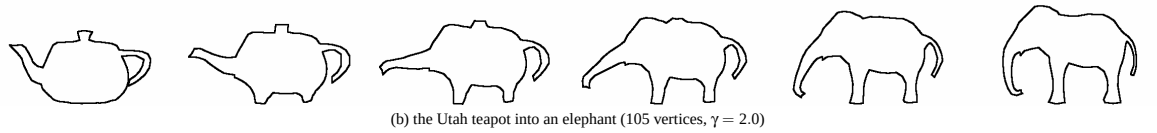
We have tested the proposed method using various types of inputs. From these experiments, we found out that the generated intermediate shapes can exhibit some area distortion in the case when the source and target shapes are skinny-long-stick-like shapes (Figure 14). This is caused by the fact that the intermediate triangle is generated by linearly interpolating the edge lengths, which expands the intermediate triangle too fast in the case of interpolating between a triangle whose center vertex is near the base edge and a triangle whose center vertex is far from the base edge.

7. Conclusions and Future Work

We have presented a new hierarchical shape interpolation method that can be used to smoothly interpolate between



(a) a donkey into a bird (151 vertices, $\gamma = 2.0$)



(b) the Utah teapot into an elephant (105 vertices, $\gamma = 2.0$)

Figure 12: *Interpolation between two polygons.*

two 2D shapes. We introduce a hierarchical representation of a shape, which is a set of triangles used to describe the shape. To interpolate between two shapes, the method first constructs the compatible hierarchical representations of the two shapes. Then, interpolating the triangles at the lowest level of the representations and adding vertices by interpolating the triangles at the next higher level successively generates the intermediate shapes. Image morphing can be performed by combining our method with the techniques proposed in Lee et al.[6], Ruprecht and Muller[7]. To summarize, the advantages of the proposed hierarchical method are: (1) interpolation starts from the coarse version of the shapes, then gradually adds the fine details, resulting in smooth interpolation sequence, (2) the features of the source and target shapes can be preserved during the interpolation.

For future research, we would like to develop a better method for computing the intermediate triangle in order to resolve the area distortion problem. By computing the intermediate shapes hierarchically, we can easily control the appearance of the generated shapes, and thus give us the ability to detecting and fixing self-intersections. Therefore, we are looking forward to extending the proposed method in order to realize self-intersection free interpolation.

Acknowledgment

This research is supported in part by Japan Society for the Promotion of Science Research Fellowship.

References

1. M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. *Proceedings of SIGGRAPH 2000*, 157-164, 2000.
2. E. Carmel and D. Cohen-Or. Warp-guided object-space morphing. *The Visual Computer*, 13(9/10):465-478, 1998.
3. M. S. Floater and C. Gotsman. How to morph tilings injectively. *Journal of Computational and Applied Mathematics*, 101:117-129, 1999.
4. E. Goldstein and C. Gotsman. Polygon morphing using a multiresolution representation. *Proceedings of Graphics Interface'95*, 247-254, 1995.
5. C. Gotsman and V. Surazhsky. Guaranteed intersection-free polygon morphing. *Computers and Graphics*, 25(1):67-75, 2001.
6. S. Lee, K.Y. Chwa, S.Y. Shin, and G. Wolberg. Image metamorphosis using snakes and free-form deformations. *Proceedings of SIGGRAPH 95*, 439-448, 1995.
7. D. Ruprecht and H. Muller. Image warping with scattered data interpolation. *IEEE Computer Graphics and Application*, 15:37-43, 1995.
8. T. Samoilov and G. Elber. Self-intersection elimination in metamorphosis of two-dimensional curves. *The Visual Computer*, 14(8/9):415-428, 1998.
9. T. Sederberg, P. Gao, G. Wang, and H. Mu. 2D shape blending: An intrinsic solution to the vertex path problem. *Proceedings of SIGGRAPH 93*, 15-18, 1993.
10. M. Shapira and A. Rappoport. Shape blending using the star-skeleton representation. *IEEE Computer Graphics and Application*, 15(2):44-51, 1995.
11. V. Surazhsky and C. Gotsman. Controllable morphing of compatible planar triangulations. *ACM Transactions on Graphics*, 20(4):203-231, 2001.
12. V. Surazhsky and C. Gotsman. Morphing stick figures using optimized compatible triangulations. *Proceedings of Pacific Graphics 2001*, 40-49, 2001.
13. A. Tal and G. Elber. Image morphing with feature preserving texture. *Computer Graphics Forum (Eurographics'99)*, 18(3):339-348, 1999.
14. Y. Zhang. A fuzzy approach to digital image warping. *IEEE Computer Graphics and Applications*, 33-41, 1996.
15. Y. Zhang and Y. Huang. Wavelet shape blending. *The Visual Computer*, 16(2):106-115, 2000.