Image Morphing with Feature Preserving Texture

Ayellet Tal *
Department of Electrical Engineering
Technion, IIT
Haifa 32000, Israel

Gershon Elber
Department of Computer Science
Technion, IIT
Haifa 32000, Israel

Abstract

Image metamorphosis as an animation tool has been mostly employed in the context of the entire image. This work explores the use of isolated and focused image based metamorphosis between two-dimensional objects, while capturing the features, colors, and textures of the objects. This pinpointed approach allows one to independently overlay several such dynamic shapes, without any bleeding of one shape into another. Hence, shape blending and metamorphosis of two-dimensional objects can be exploited as animated sequences of clip arts.

1 Introduction

The continuous evolution from a source object into a target object is generally known as metamorphosis, or morphing. Metamorphosis can produce compelling transitions between objects, and thus have numerous applications in scientific visualization and in animations in the film and advertising industries. Many morphing algorithms have been proposed. Some are based on two dimensional polylines or piecewise linear curves [7, 11, 14, 26, 27], some are based on three dimensional polyhedra [16, 28] and others are based on images [2, 5, 19, 25, 29]) or even volumetric data sets [12, 21]. In this paper we focus on image morphing. We refer to a two dimensional object in the image as an *object* or a *shape*.

The goal of morphing and shape blending is to generate in–between geometry which smoothly transforms the source shape into the target shape. This general problem requires solutions to two subproblems: the correspondence problem that aims at establishing a match between points on the two shapes and the path problem which aims at defining the way the transaction moves through. The correspondence problem can be quite difficult, and it is often the case that solutions to this problem involve manual interventions.

Methods have been developed in image morphing to mostly govern the way the correspondence is defined and help users to manually prescribe this correspondence. Such methods include prescriptions of feature points on both the source and target images that are to be interpolated [2, 20, 25, 30], line segments and edges on both images [5], or even mesh–based techniques [23, 30] where nonuniform (Bézier) meshes are used to specify the corresponding geometric features. Finding the correspondence for every pixel in the image has experienced

*Ayellet Tal is the Samuel & Miriam Wein academic lecturer

a whole variety of methods such as the exploitation of the mesh–based techniques [23, 30], radial basis functions [2], and contouring [8].

Frequently, in particular in the context of image morphing, the path problem is solved using linear interpolation, though higher interpolation scheme can be utilized as well. The path problem for two-dimensional polygons is discussed in [26], where interpolation is performed on edge lengths and angles. In [27], the interior of the polygon as well as its boundary are taken into account. In [14], a different representation of polygons, with a multi-resolution character is proposed. Polygons are then morphed accounting first for global shape, and then for local deformations.

Focused on pictorial data, this work proposes a different approach to image morphing. Rather than interpolating the entire image, perform the shape blending on isolated objects in the scenes. For instance, given two pictures of two animals, isolate the animals and blend only them. Such a scheme can be used for composing animations of various objects in a similar way to clip arts in paint packages.

Further, we seek an approach that preserves basic features in the two shapes. In the ensuing discussion, we refer to a *feature* not in its geometric meaning as lines or points in the image but rather in its real-life meaning as parts of the objects such as eyes or legs. The preserved features might be enclosed within one another, and the presented algorithm handles this case. For instance, the pupils are enclosed within the eyes, which are, in turn, enclosed within the outer boundaries of the animals.

The proposed approach can be viewed as a hybrid between image morphing methods and polygonal morphing methods. It is related to polygonal morphing since geometrically, the nested features can be often described as a set of simple two-dimensional polygons nested within each other, which need to be morphed appropriately. However, it is related to image morphing since the colors and the textures of the objects need also be captured as they appear in the two images, and morphed accordingly.

The rest of this paper is organized as follows. Section 2 presents the different stages of our algorithm. In Section 3 we present a few examples. Finally, we conclude in Section 4.

2 Algorithm

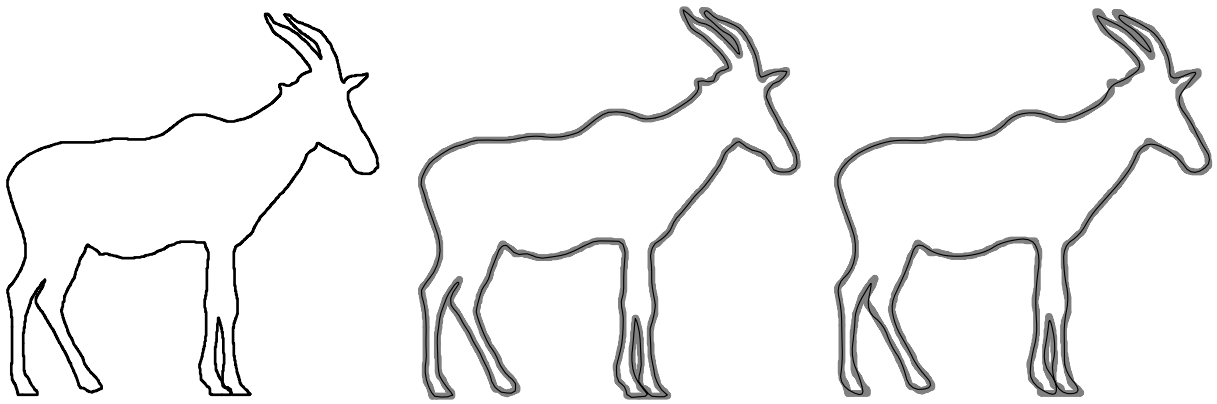
Given two images, $\mathcal{I}_i,\ i = 1, 2$, containing two objects, $\mathcal{O}_i \subset \mathcal{I}_i,\ i = 1, 2$, the proposed morphing approach consists of the following four stages:

1. *Outline extraction of* $\mathcal{O}_i,\ i = 1, 2$. The outline (or boundary) of an object i is defined as a set of non-intersecting simple closed curves $C_i(t),\ t \in [0, 1]$, which might be nested. This process is briefly presented in Section 2.1.

2. *Establishment of correspondence.* The correspondence is established between the two sets of curves $C_1(t)$ and $C_2(t),\ t \in [0, 1]$, a stage that is described in Section 2.2.

3. *Compatible triangulation.* The matched curves are sampled and replaced by piecewise linear curves, P_1 and P_2, and a compatible triangulation between P_1 and P_2 is computed, a stage discussed in Section 2.3.

4. *Texture mapping.* Finally, the end result is formed with the aid of the original colors and the original textures from $\mathcal{I}_i,\ i = 1, 2$, the outlines of P_i, the compatible triangulation and the established correspondence. This final stage is portrayed in Section 2.4.



(a) (b) (c)

Figure 1: (a) shows a piecewise linear outline of a topi as extracted via the outlining tools. It contains over 3500 points. In (b), a quadratic Bspline curve (in black) with 200 control points has been fitted to the shape of the piecewise linear curve (in gray) of (a). (c) shows a similar fit with 100 control points. The piecewise linear outline shown in (a) is also displayed in (b) and (c) in gray.

2.1 Outline Extraction

Given two images, $\mathcal{I}_i$, $i = 1, 2$, containing two objects, $\mathcal{O}_i$, $i = 1, 2$, we seek the outline boundaries of $\mathcal{O}_i$ in $\mathcal{I}_i$. While it seems plausible that edge detection tools (e.g, [6]) can be successfully employed in this context, we found that a semi-automatic procedure is much preferred. Following the approach suggested in [22], we employ an interactive tool that snaps to the area with the large gradient in the image. Further, we found that the user must be able to precisely control this extraction of the outlines of the shape as in many cases the gradients of the image do not lead to the exact curve that the user seeks. Clearly, other extraction methods of outlines of shapes may be employed.

Because this stage can generate a large amount of point data, this data is reduced with the aid of least squares fitting of Bspline curves. Figure 1 shows an example of a piecewise linear outline reduced via least squares.

2.2 Matching the Curves

Given two sets of n closed Bspline curves, $\mathcal{C}_1 = \{C_{11}(t), C_{12}(t), \ldots, C_{1n}(t)\}$ and $\mathcal{C}_2 = \{C_{21}(t), C_{22}(t), \ldots, C_{2n}(t)\}$, $t \in [0, 1]$, we would like to establish a correspondence between common features in each pair of curves $C_{ik}(t)$, $i = 1, 2$, for all $1 \leq k \leq n$. We assume that the two curves are alike. That is, both curves are simultaneously outlines of animals, faces, or cars. Otherwise, the more different the two shapes are, the lesser is the need to preserve the different features of the shapes.

Features such as the legs or the mouth can be distinguished via their patterns in the normal or Gaussian map of the shape of the curve. Such features generate a distinct turning signature in the curve that yields a large variation in the Gaussian image.

Let $T_{ik}(t)$ be the unit tangent field of $C_{ik}(t)$, $t \in [0, 1]$, $i = 1, 2$, $1 \leq k \leq n$. For planar curves, the normal vector field of $C_{ik}(t)$ is always perpendicular to $T_{ik}(t)$. Hence,

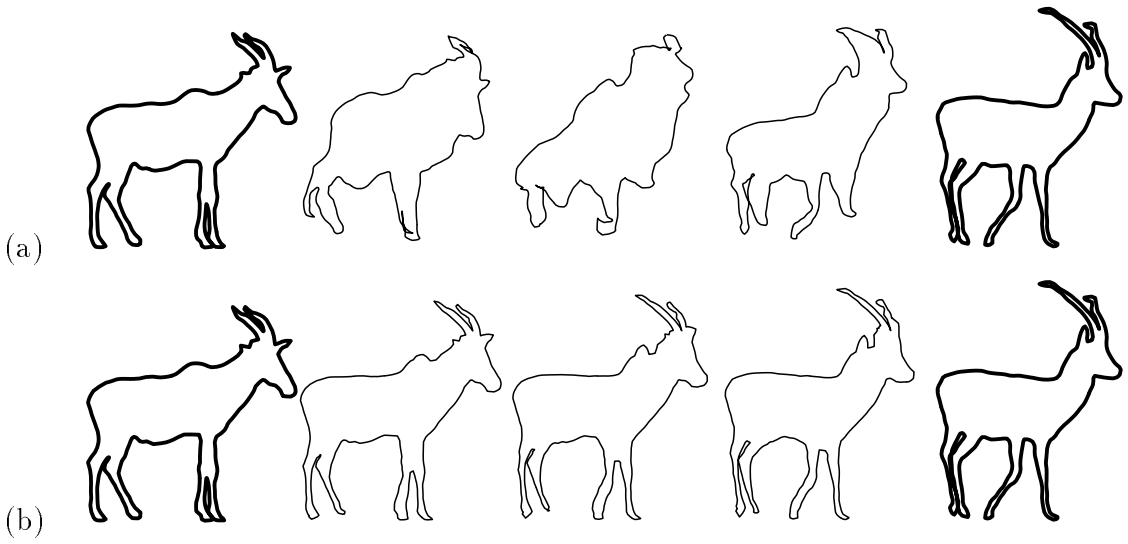


Figure 2: (a) shows a naive linear blend of the outlines of a topi (left) and gazelle (right) that were created by the outline extraction stage. In (b), the automatic procedure of the matching algorithm has been applied to reparametrize one of the curves (i.e., a gazelle), yielding a much better result that preserves features such as the legs, the horns, etc.

for simplicity, we employ $T_{ik}(t)$ instead of the Gaussian image. In [10], the correspondence between the two curves is established by deriving a reparametrization to one of the two curves, $r(t)$, by maximizing the following function,

$$\max_{r(t)} \int_0^1 \langle T_{1k}(t), T_{2k}(r(t)) \rangle \, dt, \tag{1}$$

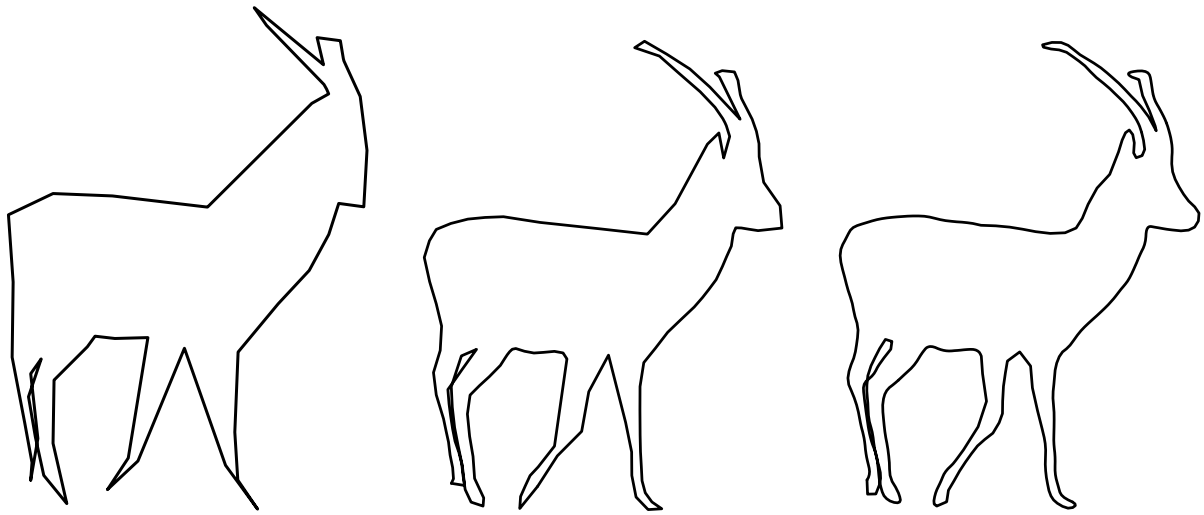
subject to the end conditions of $r(0) = 0$ and $r(1) = 1$, and $r(t)$ is an allowable change of parameter (i.e., monotone). $\langle T_{1k}(t), T_{2k}(r(t)) \rangle$ can assume values between -1 and 1 and therefore Equation (1) is maximized when the two tangent fields are identical throughout.

Finding a solution to the continuous optimization problem presented in Equation (1) is difficult. Instead, the problem can be discretized by sampling m points on each of the two curves and, as in [10], the optimization problem which needs to be solved is:

$$\max_{j(i)} \sum_{i=0}^{m-1} \left\langle T_{1k}^{i}, T_{2k}^{j(i)} \right\rangle, \tag{2}$$

subject to $j(0) = 0$, $j(m-1) = m-1$, and $j(i) \leq j(i+1)$.

Using dynamic programming, the global optimum is found to Equation (2) in quadratic time, $O(m^2)$. The solution to Equation (2) provides the optimal match between the two discrete sets of the tangents of curves $C_{ik}(t)$, $i = 1, 2$. This discrete solution, $j(i)$, converges to the solution of the continuous problem of Equation (1) as m becomes larger. In [10], $j(i)$ is employed to reparameterize one of the two curves so that the two curves share a regular parameterization, an approach we exploit here as well. Figure 2 shows one example of the output of this matching algorithm between a topi and a gazelle.



(a) (b) (c)

Figure 3: Piecewise linear samples of the outline of the gazelle, once the matching has been established. (a) shows the results of sampling with 50 points. (b) employs 150 points and (c) employs 500 points

.

While the matching algorithm is completely automatic and yields quite satisfactory results, it can also match undesired features. For example, in Figure 2 (b), the back of topi is matched to the ear of the gazelle. Manual pinpointing of the ear of the topi can recover the desired situation. This minimal and optional intervention by the user is feasible by applying the matching procedure between the prescribed domains only.

The next stage of the algorithm involves the derivation of a compatible triangulation, and requires that the piecewise linear outlines of the two shapes have the same number of samples. Hence, this stage ends with a sampling at the desired resolution of the two sets of Bspline curves into two sets of piecewise linear closed curves. Figure 3 shows the gazelle sampled at 50, 150 and 500 points along its outline, after the correspondence has been established. The sample with 500 points in Figure 3 (c) is more than sufficient while the sample with 150 points in Figure 3 (b) is almost as good.

2.3 Compatible Triangulation

At this stage we apply a compatible triangulation algorithm. Given two polygons P_1 and P_2, each with n vertices, their compatible triangulation is a joint labeling of the vertices of the polygons and of some of their internal points, such that a triangulation of one polygon admits a triangulation in the other polygon which is labeled compatibly. Compatible triangulation algorithms have been mainly used for three–dimensional morphing, as in [16, 17, 28]. They were less utilized in polygon morphing or in image morphing, though algorithms have been devised in computational geometry.

It is shown in [3] that there always exists a compatible triangulation, where $k = O(n^2)$ Steiner points are allowed. Algorithms for constructing compatible triangulations of simple polygons are presented in [3, 15, 18]. When the polygons have holes, it is shown in [4] that

a compatible triangulation of size $O(n^2)$ always exists, and an algorithm is presented.

We follow [3], and extend its algorithm to work for polygons with holes. Though our algorithm is less general then [4], its implementation is much easier, and it is less prone to degenerate cases. In fact, the algorithm worked quite well for all our examples, as will be shown in Section 3. Handling holes is important since it allows for better control over internal features such as the eyes and the mouth while avoiding vanishing and bleeding effects of the textures of those features into other regions of the objects. We will first describe the algorithm for simple polygons, and then its extension to polygons with one or more holes.

Given two simple polygons P_1 and P_2, a triangulation T_1 (respectively, T_2) is first found for P_1 (P_2), using any triangulation algorithm (e.g., [9, 13, 24]). The triangulation T_1 is then mapped into a triangulation T_1' of P, where P is a simple polygon with n vertices. Similarly, T_2 is mapped into a triangulation T_2' of P. Overlaying T_1' and T_2' on P yields a convex subdivision, which can be easily triangulated. Every triangle in this refined triangulation is fully contained in some triangle of T_1' and in some triangle of T_2'. Hence, It can be mapped back into T_1 and T_2 to obtain a compatible triangulation of P_1 and P_2.

If the polygon has a hole then the goal is to remove the hole by adding a *bridge* between the hole and the outer boundary. To construct a bridge, a segment connecting two vertices, one on the hole and the other on the outer boundary, is first added. The segment should intersect the boundary of the polygon only at the segment's vertices. A bridge is represented by two such segments, having opposite orientations. Thus, adding a bridge removes the hole from the polygon, and transforms the polygon into a simple polygon. For our problem, the goal is to find a compatible bridge, i.e., a bridge whose endpoints have the same labeling in both polygons.

Given a a pair of polygons P_1 and P_2, each with h holes, the holes are removed one by one. Topologically, the outer boundary of the polygon is no different from the boundaries of the holes. Thus, a hole can be removed either by adding a bridge from that hole to the outer boundary or by adding a bridge to another hole. After a single hole is removed, the problem is reduced to the problem of finding a compatible triangulation for a pair of polygons with $h-1$ holes. Repeating this procedure h times yields a simple polygon. Moreover, the algorithm can be applied recursively to holes which are nested within other holes.

For selecting bridges we employ the following greedy procedure. Every possible bridge in P_1 gets a score. The score might reflect, in addition to the Euclidean distance, some other properties, such as the preferred outlines for connection. Some of the candidate bridges found in P_1 do not yield a compatible bridge in P_2 since the compatible segment intersects the boundary of P_2 in an internal point of that segment. The more similar P_1 and P_2 are, the higher the probability that candidate bridges found in P_1 indeed have compatible bridges in P_2. In fact, in practice, there are many feasible bridges. Of all the candidate bridges, we select the bridge with the joint best (minimum) score. In the extreme case, where simple compatible bridges cannot be found, a more complex polygonal path needs to be constructed in P_2. This can always be done by selecting a path which is $\epsilon-$away from the boundary, for instance, similarly to the way proposed in [4].

The compatible triangulation algorithm is applied to the outer polygon (after the bridges are constructed) as well as to each of holes, recursively. For instance, the algorithm can be applied to the outer boundaries of the animals, recursively to their eyes, and if desired, recursively to the their pupils. This ensures that each distinctive feature is morphed into its compatible feature in a way that preserves the feature's properties.

The results of the algorithm are illustrated in Figure 4. In Figure 4(a)-(b), the objects

are triangulated ignoring the holes of the eyes. In the resulting morph sequence, the eye disappears in order to re-appear in a different location. However, in Figure 4(c)-(d), the objects are triangulated after compatible bridges have been constructed between the eyes and the outer boundaries of the animals. In Figure 4(e)-(f), a zoomed version of the compatible triangulation of the eyes is shown. In this case, the eye of the gazelle is mapped to the eye of the topi throughout the morph sequence, and all bleeding effects of the vanishing and the reappearance of the eye are avoided.

2.4 Texture Mapping

We are now given two lists of l triangular polygons, P_{ij}, $i = 1, 2$, $j = 0, ..., l-1$, and need to compute the in-between triangles. Assume an intermediate blending value t between zero and one, and let the three vertices of triangle P_{ij} be V_{ij}^k, $k = 1, 2, 3$. For each pair of triangles $\{P_{1j}, P_{2j}\}$, the intermediate triangle, $P_j(t)$, is computed as,

$$P_j(t) = \left\{ tV_{2j}^k + (1-t)V_{1j}^k \right\}, \ k = 1, 2, 3,$$

blending the vertices of the two compatible polygons.

Given $P_j(t)$, all the pixels covering $P_j(t)$ are now visited and scan converted. For each pixel $p \in P_j(t)$, the three barycentric coordinates, β_k, $k = 1, 2, 3$, are computed so that

$$p = \sum_{k=1}^{3} \beta_k \left(tV_{2j}^k + (1-t)V_{1j}^k \right).$$

The barycentric coordinates are applied back to the three vertices of P_{1j} and to the three vertices of P_{2j}, to compute the two corresponding pixels on the two original images, $\mathcal{I}_i$, $i = 1, 2$.

$$p_i = \sum_{k=1}^{3} \beta_k V_{ij}^k, \ i = 1, 2.$$

Then, given the colors of the pixels p_1 and p_2, denoted as $c(p_1)$ and $c(p_2)$ respectively, the final color of pixel p, $c(p)$, is set to be

$$c(p) = tc(p_2) + (1-t)c(p_1).$$

3 Examples

The example we have employed throughout this paper of a two-dimensional shape blending between a topi and a gazelle yields the result shown in Figure 5. These two animals were extracted from images that were down loaded from the WWW [1]. The final background of Figure 5 is completely different from the backgrounds of the two original images and is taken from of a whole different image. This is possible since the objects were morphed regardless of the images they were extracted from.

Another similar example of a metamorphosis between two animals, this time quite different, is shown in Figure 6. Herein, a giraffe and a warthog are similarly matched and blended together.

In both examples, that of the topi and the gazelle and that of the giraffe and the warthog, key interior features of the animals vanish. Most noticeable, the eyes disappear, only to

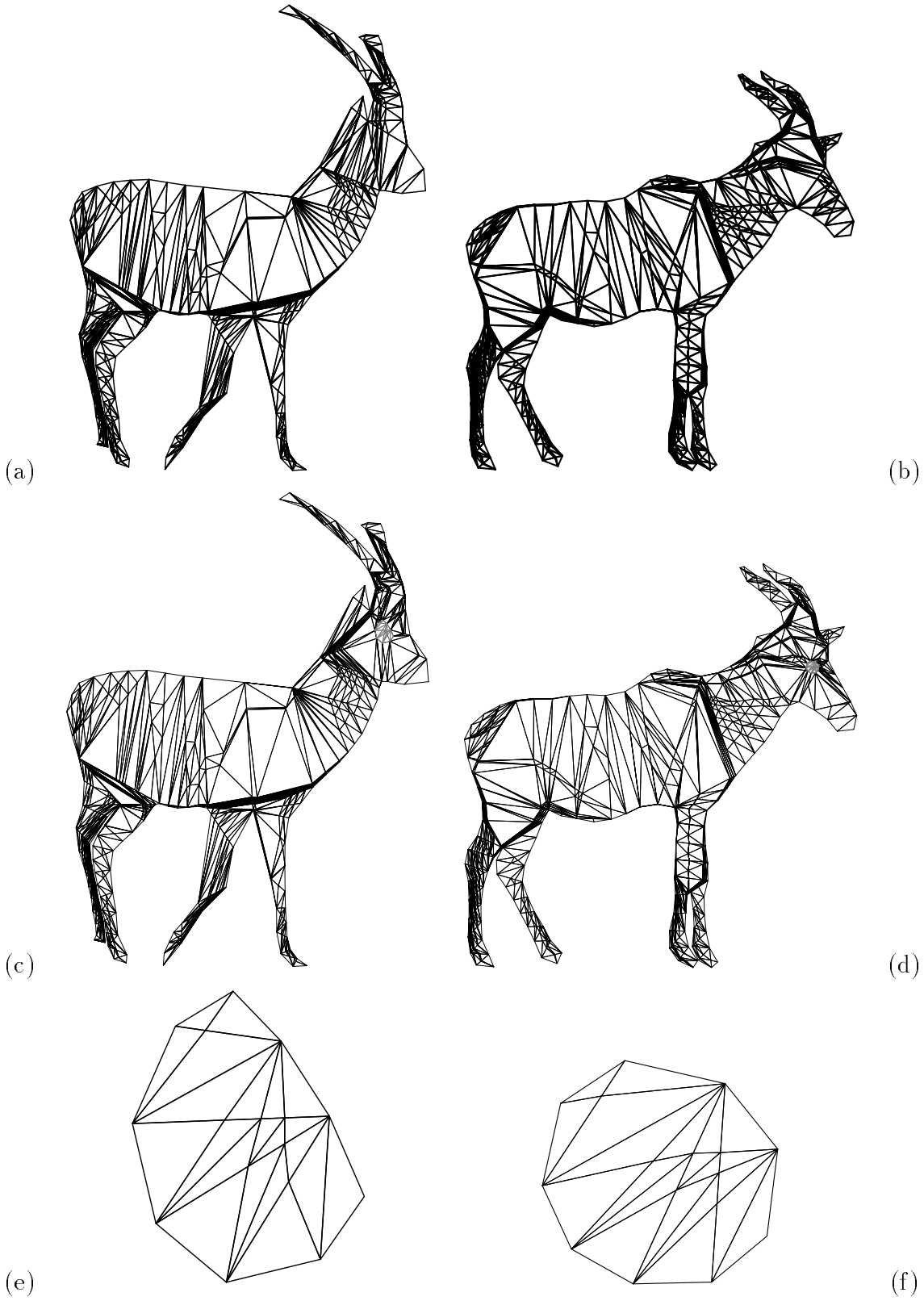


Figure 4: The compatible triangulation established between the gazelle and the topi for 150 samples along the outline. (a) and (b) show the compatible triangulation without the eye taken into consideration. (c) and (d) show the compatible triangulation with the eye taken into consideration as a hole. Finally, (e) and (f) show the two zoomed up regions of the eyes.

Figure 5: The final metamorphosis between the gazelle and the topi. The original gazelle is shown at the top left corner whereas the original topi is shown at the bottom right. Note the stationary background throughout the metamorphosis sequence.

reappear at a different location. Figure 7 shows a zoomed section from Figure 6 that focuses on the eyes.

The eyes are a crucial organ that is very distinctive and its preservation can vastly improve the quality of the blend. As explained before, we are able to preserve any key interior feature by adding interior contours as holes. When a second contour around the eye is extracted and the algorithm described in this work is applied to the outline of the animal with a *hole* in the form of the eye, the resulting morph can greatly improve. Figure 8 shows the result of applying a new contour to the eyes of the two animals.

Finally, because the metamorphosis is conducted on a well isolated domain bounded by an exact outline of the shape, one can compose animations of various objects, much like clip arts in two-dimensional drawing and/or paint packages. In Figure 9, two gazelles are blended with two topis. This scene was created by pasting together two animated clip arts of the metamorphosis sequence presented in Figure 5, at different locations on the image.

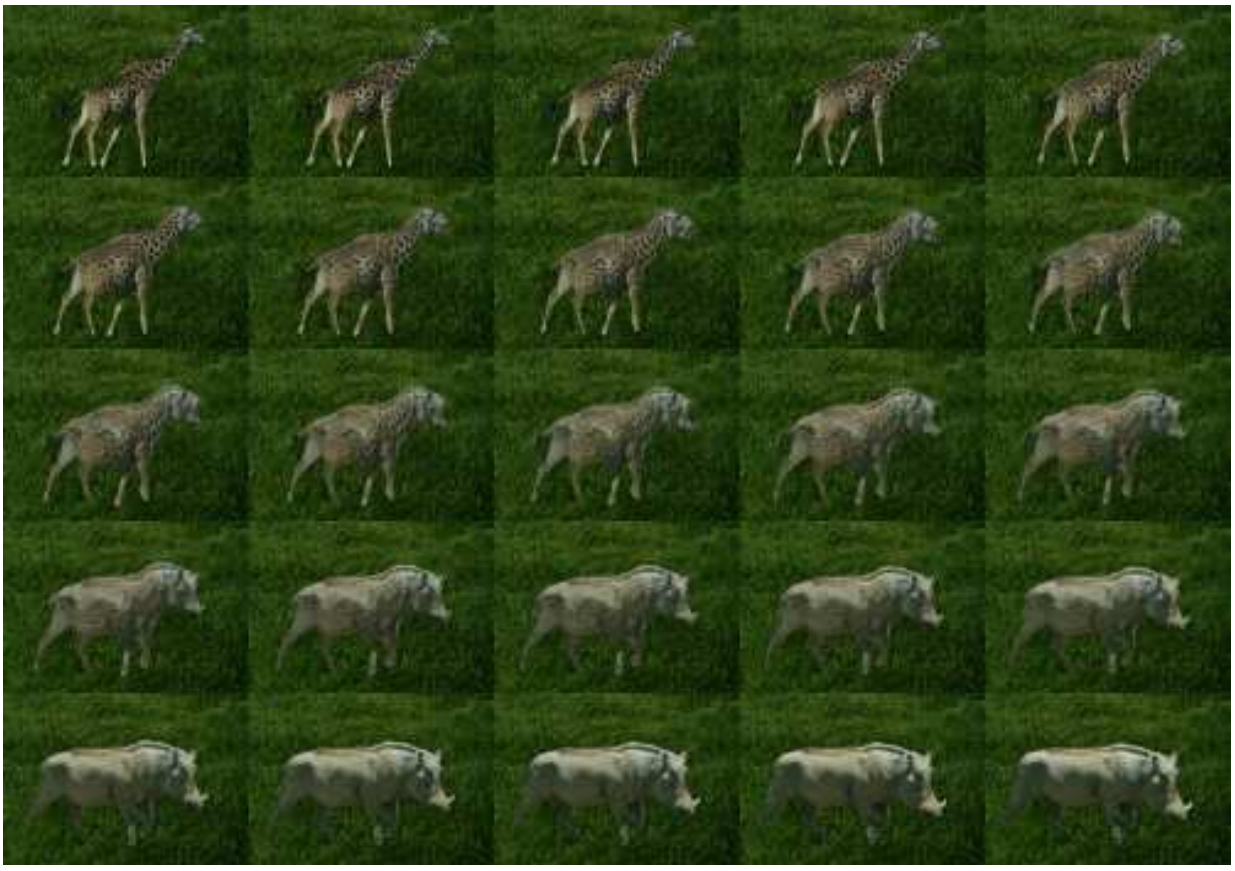


Figure 6: A metamorphosis of a giraffe (top left) and a warthog (bottom right).

4 Conclusions

This work explored the metamorphosis of isolated two-dimensional shapes within images, which might be nested within each other. We presented an algorithm that allows one to capture and morph arbitrary number of such shapes in a single image independently, and without any bleeding of one shape into another, thus allowing one to preserve the main features of the objects. Moreover, the presented approach can be exploited for putting together dynamic clip arts in animated sequences.

The algorithm consists of four stages: outline extraction, establishment of correspondence, construction of a compatible triangulation, and texture mapping. Each of the these stages forms an interesting and intriguing problem by itself. We chose a single solution for each of these subproblems. However, other techniques devised for the various components can be sought and easily integrated into our general algorithm.

Acknowledgments: We thank Daniel Bricker for implementing the compatible triangulation algorithm.

This work was partially supported by the Fund for Promotion of Research at the Technion, Haifa, Israel.

Figure 7: A zoomed up section of the eyes of Figure 6. The eyes of the giraffe disappears during the metamorphosis whereas the eyes of the warthog appear at a totally different location.

Figure 8: A zoomed up section of the eyes of Figure 6. The eyes during the blend are corrected with the aid of an interior contour around the eyes. Compare with Figure 7.

Figure 9: Because the metamorphosis is conducted on a well isolated and exact domain of the object, in the image, one can compose arbitrary number of such animated metamorphosis sequences, much like the static clip arts used in two dimensional drawing and/or paint packages. This figure shows two animated sequences composed together at different locations in the image. Compare with Figure 5.

References

[1] African Studies WWW (U. Penn).
http://www.sas.upenn.edu/African_Studies/
Wildlife_GIFS/menu_Wildlife.html.

[2] N. Arad, D. Reisfeld. Image warping using few anchor points and radial functions. *Computer Graphics Forum* 14(1), pp 35–46, 1995.

[3] B. Aronov, R. Seidel, D. Souvaine. On compatible triangulations of simple polygons. *Computational Geometry: Theory and Applications* 3, pp 27–35, 1993.

[4] M. Babikov, D. Souvaine and R. Wenger. Constructing Piecewise Linear Homeomorphisms of Polygons with Holes. *Proceedings of the 9th Canadian Conference on Computational Geometry*, 1997.

[5] T. Beier and S. Neely. Feature–based image metamorphosis. In *Proceedings of SIGGRAPH '92*, Vol 26(2), pp 35–42, June 1992.

[6] Canny, J.F. (1986). A Computational Approach to Edge Detection, *PAMI* 8(6), pp.679-698.

[7] E. Carmel and D. Cohen-Or. Warp–guided object–space morphing. *The Visual Computer,* 13:(9+10), pp 465–478, 1997.

[8] K. H. Chan and R. W. Lau. Counter-Based Warping Graphical Models and Image Processing, Vol 60, N0 5, pp 331-348, September 1998.

[9] B. Chazelle. Triangulating a simple polygon in linear time. *roc. 31st IEEE Symp. on Foundations of Computer Science,* pp 220–230, 1990.

[10] S. Cohen, G. Elber, and R. Bar Yehuda. Matching of Freeform Curves. CAD, Vol 29, No 5, pp 369-378, 1997.

[11] D. Cohen-Or, D. Levin and A. Solomovici. Contour blending using warp–guided distance field interpolation. In *Proceedings of Visualization '96*, pp 165–172, October 1996.

[12] D. Cohen-Or, D. Levin and A. Solomovici. Three dimensional distance field metamorphosis. In *ACM TOG*, April 1998.

[13] M. R. Garey, D. S. Hohnson, F.P. Preparata and R.E. Tarjan. Triangulating a simple polygon. *Information Processing Letters*, 7(4) 175–179, 1978.

[14] E. Goldstein and C. Gotsman. Polygon morphing using a multiresolution representation. *Graphic Interface '95*, 247–254, 1995.

[15] H. Gupta and R. Wenger. Constructing piecewise linear homeomorphisms of simple polygons. *J. Algorithms*, 22, 142–157, 1997.

[16] T. Kanai, H. Suzuki and F. Kimura. 3D geometric metamorphosis based on harmonic maps. *Proceedings of Pacific Graphics '97*, 97–104, October 1997.

[17] J.R. Kent, W.E. Carlson, and R.E. Parent. Shape transformation for polyhedral objects. *Computer Graphics*, 26(2):47–54, July 1992.

[18] E. Kranakis, and J. Urrutia. Isomorphism triangulations with small number of Steiner points. *Proceedings of the 7th Canadian Conference on Computational Geometry*, 291–296, 1995.

[19] S-Y. Lee K-Y. Chwa, S. Y. Shin and G. Wolberg. Image Metamorphosis Using Snakes and Free-Form Deformations, *SIGGRAPH '95*, pp 439–448, 1995.

[20] P. Litwinowics and L. Williams. Animating images with drawings, *SIGGRAPH '94*, pp 409–412, 1994.

[21] A. Lerios, C. D. Garfinkle, and M. Levoy. Feature-based volume metamorphosis. In *Proceedings of SIGGRAPH '95*, pp 449–456. ACM, August 1995.

[22] E. N. Mortensen and W. A. Barrett. Interactive Segmentation with Intelligent Scissors. Graphical Models and Image Processing, Vol 60, No 5, pp 349-384, September 1998.

[23] T. Nishita. T. Fujii and E. Nakamae. Metamorphosis using Bezier Clipping. First Pacific Conference on Computer Graphics and Applications, pp 162–173, 1993.

[24] J. O'Rourke. Computational Geometry in C. Cambridge University Press, 1994.

[25] D. Ruprecht and H. Müller. Image warping with scattered data interpolation *IEEE Computer Graphics and Applications*, pp 37–4315(2), 1995.

[26] T.W. Sederberg, P. Gao, G. Wang, and H. Mu. 2–D shape blending: An intrinsic solution to the vertex path problem. In *Proceedings of SIGGRAPH '93*, Vol 27, pp 15–18. ACM, August 1993.

[27] M. Shapira and A. Rappoport. Shape blending using the star–skeleton representation. *IEEE Computer Graphics and Applications*, 15:44–50, March 1995.

[28] A. Shapiro and A. Tal. Polyhedron Realization for Shape Transformation. *The Visual Computer*, to appear.

[29] S. M. Seitz and C. R. Dyer. View Morphing: Synthesizing 3D Metamorphosis Using Image Transforms In *Proceedings of SIGGRAPH '96*, pp 21–30, 1996.

[30] G. Wolberg. Digital Image Warping. *IEEE Computer Society Press*, 1990.