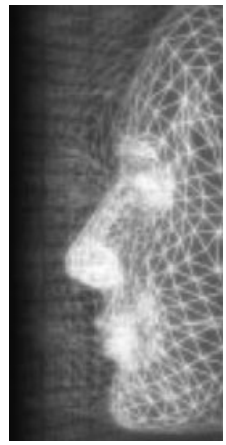
COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2004; **15:** 443–452 (DOI: 10.1002/cav.48)

Morphing based on strain field interpolation



*By Han-Bing Yan, Shi-Min Hu and Ralph Martin**

Strain fields provide a method of deformation measurement based on physics. Using these as a tool, we can analyze deformation of objects in a measurable way. We have developed a new morphing technique based on strain field interpolation. Shape shaking and squeezing, which often happen when using linear interpolation for morphing, do not arise in our approach. We have also developed a new method to create isomorphic meshes from corresponding objects in two images. Meshes generated by this method have much fewer triangles than other methods, which greatly decreases calculation loads in the morphing process. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: morphing; strain field; finite element method; isomorphic mesh

Introduction

Techniques of transforming one image or object into another have gained greatly in importance in recent years, and are known as morphing or shape blending. These techniques involve producing a sequence of intermediate objects or images that smoothly evolve from source to target. Many examples have been produced by these techniques, both in movie and game production, and other fields such as visualization, education, and entertainment. Previous methods include, for image morphing, those in[1–4] and for geometry morphing, those in.[5–9] Image based techniques always deal with objects having similar geometry. Geometry based techniques can handle objects with complex shapes, and generally include the capabilities of image based techniques.

Assuming that the shape is represented as a polygon, the easiest way to perform morphing of the boundary is to perform linear interpolation between corresponding vertices. Sederberg[5] introduced a way to find intermediate shapes between two polygons by minimizing a physical concept of work done during deformation of boundaries. Subsequently, Sederberg proposed a technique which interpolates intrinsic parameters—angles and edge lengths of the polygons,[6] which gives a simple but effective method for most polygon boundary blending problems. Zhang presented a technique using fuzzy methods to create a consistent boundary and applied it to morphing.[7] Such methods based on polygon boundaries have two shortcomings. Firstly, because representing the object by its boundary does not consider the interior, if the objects being interpolated are very different, poor results are often obtained. Secondly, using only a polygon to represent the boundary does not help us map texture from source to target. Shapira and Rappaport[10] suggest using compatible skeletons to represent the interiors of 2D shapes to be morphed, and applies the blend to a parametric description of the skeletons. While this method does take the interior into account, it does not completely solve the problems above.

Work which developed the approach proposed in[8] was presented in[9]. This method is based on determining local non-distorting motions of the triangles in a mesh representation, rather than a global one. They decompose the affine transformation for each triangle into a rotation matrix and a stretch matrix. An optimization method is used to minimize the difference between the desired transformation, and the actual transformation which is applied, taking into account the connectivity constraints on adjacent triangles. This method uses internal shape information, and can map textures during the morphing process. However, it requires high quality meshes to avoid numerical problems. While this method can be used in 2D and 3D, in the latter case, it can only work with tetrahedral meshes. It is very hard to

*Correspondence to: Ralph Martin, School of Computer Science, Cardif University, U.K.
E-mail: ralph@cs.cf.ac.uk

generalize it to handle surface models directly; such models are widely used in 3D graphics.

Hu *et al.*[11] also considered how use internal details of the object to solve morphing problems. They used a physically-based idea which tries to minimize the sum of deformation energies between each pair of adjacent frames. Although this method gives good results in many cases, it has several deficiencies. Firstly, an iterative method is used to find the mesh for each frame. Because the solutions for all frames are found simultaneously, it is not easy to get a good initial value, especially when the differences between the objects is very large. Secondly, as the number of intermediate frames increases, the cost of solving the optimization problem increases cubically.

Lee *et al.*[4] proposed a physically-based thin plate method to solve image morphing problems. Minimization of physically meaningful energy was used to derive one-to-one warping functions. However, warp function interpolation is akin to position interpolation for a selected set of points, so it is again somewhat aribtrary for shapes having large differences, like those in Figure 2.

Most geometric methods mentioned above assume that the objects have the same topology. In,[12,13] morphing techniques for objects with differing topologies are presented.

We aim to describe object deformation in an analytical way and use this to realize natural morphing. Physics provides us with a strong tool for doing this: *strain*. Our paper presents a novel technique for morphing based on strain field interpolation. It allows the morphing of textured objects by firstly decomposing each object into an isomorphic triangular mesh, and then uses strain field interpolation to control the deformation of one mesh into the other. Mapping the texture is straightforward once we are able to map the polygons, e.g. by interpolating values at corresponding barycentric coordinates.

The major contribution of this paper is the new morphing method, so we consider it first, before the new method for producing isomorphic meshes.

Morphing Based on Strain Field Interpolation

Physics of Object Deformation

Object deformation happens in many natural processes, and most people have some intuitive idea of whether a morphing sequence looks natural or not. Real world

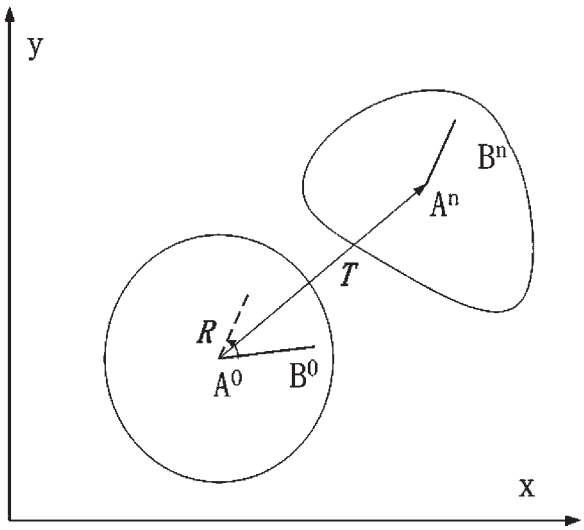


Figure 1. Motion decomposition.

deformation takes place in accordance with physical principles, so it seems sensible to try to apply them to computer generated morphing. Mechanics gives a formulation for the description of the deformation of objects.

In Figure 1, a source object is morphed into a target. A_0 and A_n, B_0 and B_n are two pairs of corresponding points. Physically, the whole transformation can be decomposed (not uniquely) into three parts: a translation, a rotation and a deformation. The translation and rotation are rigid body motions. If only rigid body motion happened, perfect morphing results could be obtained by simply linear interpolating the translation vector and rotation angle. The difficulty arises in dealing with object deformation.

In mechanics, the deformation extent of objects is defined by a strain field, which can be derived from the object's displacement field (or position field). Because displacement in morphing problems is always large, a *large deformation* formula must be used instead of the *small deformation* formula most often used in mechanical engineering. In 2D, the strain field has 3 independent components: ε_x, ε_y and γ_{xy}. At a point p, ε_x gives the local infinitesimal scaling in the x direction, ε_y gives the scaling in the y direction, and γ_{xy} is the shear strain which represents the relative change in angle between lines initially in the x and y directions at p. ε_x is positive when the point is in tension along the x direction, and negative in compression; similarly for ε_y. γ_{xy} is positive when the angle becomes smaller, and negative when the angle becomes larger.

The relationship between the strain field components and the position field is given by:

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}[(\partial x'/\partial x)^2 + (\partial y'/\partial x)^2 - 1] \\ \frac{1}{2}[(\partial x'/\partial y)^2 + (\partial y'/\partial y)^2 - 1] \\ \partial x'/\partial x\, \partial x'/\partial y + \partial y'/\partial x\, \partial y'/\partial y \end{bmatrix} \quad (1)$$

where x and y are the position field component in the x and y directions respectively, before deformation, and x' and y' are components of the position field after deformation. The displacement field is related to the position field by $x' = x + u$, $y' = y + v$, where u is the displacement field component in the x direction, and v is its component in the y direction. The difference between the position field and the displacement field is only a constant initial position field.

In the following, the strain field will be represented in vector form, so ε means $[\varepsilon_x, \varepsilon_y, \gamma_{xy}]^T$.

The strain field and the displacement field can be both used to describe deformation, but they are not the same. Displacement is a macroscopic quantity, which represents object rigid body motion and motion accumulated by deformation. Strain is an infinitesimal quantity, which only represents the deformation at each point. Using the strain field to describe deformation is more convenient than the displacement field or position field. Note that the well known linear interpolation morphing method interpolates the position field.

Motivation

Figure 2 shows morphing results calculated by (a) linear interpolation, and (b) the new method. Clearly the latter is intuitively better. Using the former method, the upper part of the object shrinks abruptly at first and swells quickly later. Visually it seems that needless deformation occurs in this case.

Figure 3 shows how the strain (averaged over the triangle) varies for the marked triangle in Figure 2 during the morphing sequence, computed using equation (1). Note that in the case of linear interpolation, the strain curves for ε_x and ε_y become highly negative at first, which means the triangle is highly compressed, i.e. smaller. The curves then quickly become positive, which means the triangle eventually ends up in tension and hence bigger. This *squeezing* effect is highly undesirable. Using the new method based on strain field interpolation, the strain curves change montonically, as desired, and no squeezing takes place: the shape of the object changes in a monotonic manner.

This example illustrates how strain can be an effective tool to analyze object deformation. The key factor for a smooth deformation process is that the curve of strain against time should not only vary slowly, but also be monotonic (but not necessarily linear, as will be explained later). Based on these observations, we present an object morphing technique using strain field interpolation.

Fixing the Rigid Body Motion

First, we give some definitions. We assume that the time associated with the source frame is $t = 0$, and the time of the target frame is $t = 1$. We suppose that $n - 1$ frames are inserted between the source frame and target frame, and the frames are numbered from 0 to n. The time associated with the i^{th} frame is t_i.

We eliminate the rigid body motion between the source and target frames first. The rigid body motion

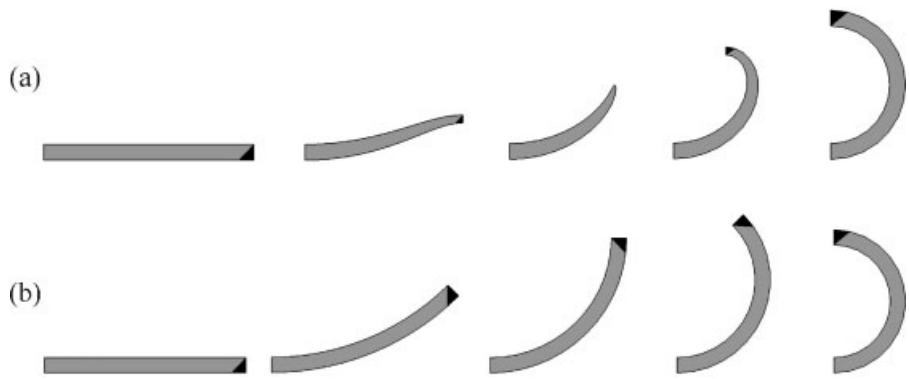


Figure 2. Morphing using (a) linear interpolation, (b) strain field interpolation.

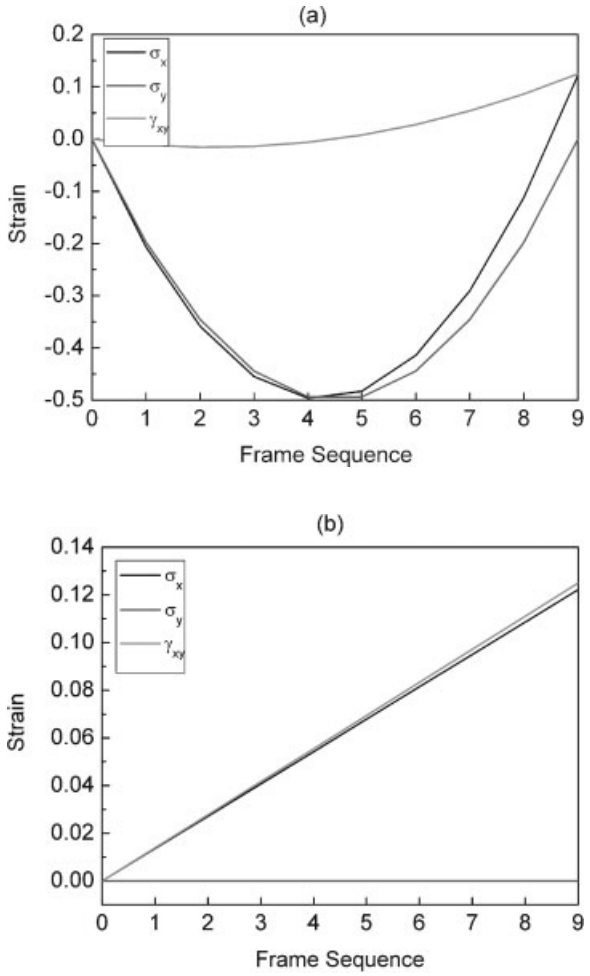


Figure 3. Strain versus time for (a) linear interpolation, (b) strain field interpolation.

does not influence the strain field calculation, but the matrix K_l^t in equation (11) is singular if the rigid body motion is not constrained, whereupon it would not be possible to obtain a unique solution for the position field.

A 2D object has three degrees of freedom. Again refer to Figure 1. Let A^0 be a chosen reference point for translation, and B^0 be a chosen reference point for rotation, which correspond to A^i,B^i at a later time. We define the vector T from A^0 to A^n to be the translation vector from source to target. The angle R from A^0B^0 to A^nB^n is the angle of rotation between the two objects, in the range $-\pi < R \leq \pi$.

Let $R_{A^0B^0}$ be the angle between A^0B^0 and the x axis in an anti-clockwise direction. Let T_{A^0} be the vector from the origin to A^0.

We move the source and target objects so that A^0 and A^n coincide with the origin. We then rotate B^0 around the origin until B^0 is on the positive x axis, and do the same for B^n. The x and y coordinates of A^0 and the y coordinate of B^0, and the corresponding coordinates of A^i and B^i in all other frames, are fixed in subsequent operations until we restore the rigid body motion at the end of the method.

Strain Field Interpolation

The strain field ε^n between the source object and the target object can be calculated from equation (1). To obtain the ideal strain field for each intermediate frame, a simple method which is sometimes adequate is to use linear interpolation:

$$\varepsilon_x^t = (1-t)\varepsilon_x^n, \qquad \varepsilon_y^t = (1-t)\varepsilon_y^n. \tag{2}$$

Good results can be obtained by linear interpolation of strain fields in cases where there are large displacements and small strains, which is often the case—for example, the object bending shown in Figure 2.

If the strain is very large, each intermediate frame calculated by this method will still have a good shape showing little distortion, but the results are usually uneven along the time axis, as in the example in Figure 4.

The reason for this phenomenon is that although the strain field represents the deformation of an object, it is not *linearly* related to the length scaling of a line element in the object.[14] Equation (3) provides a modified scheme which ensures that line length changes in the x and y directions, and angular changes between lines initially in the x and y directions, occur linearly with t; it also constrains the line length change rate in other directions.

$$\begin{aligned} \varepsilon_x^t &= \frac{1}{2}\left[\left[t\sqrt{1+2\varepsilon_x^n}+(1-t)\right]^2-1\right], \\ \varepsilon_y^t &= \frac{1}{2}\left[\left[t\sqrt{1+2\varepsilon_y^n}+(1-t)\right]^2-1\right], \\ \gamma_{xy}^t &= \tan\left(t\arctan\gamma_{xy}^n\right). \end{aligned} \tag{3}$$

Figure 4. Morphing using (a) linear strain field interpolation, (b) modified strain field interpolation.

Experiments show that using equation (3) instead of linear strain interpolation gives much better visual results for cases involving large strains.

Having used the interpolation methods above to find the desired strain field at a particular time, we must now compute the displacement field or the position field.

In fact, physically correct strain field components ε_x^t, ε_y^t, and γ_{xy}^t must be related by the complicated *Compatibility Equation*.[14] Trying to correct the above interpolated strain fields to meet the compatibility equation would be very difficult. Thus, instead, we attempt to estimate a displacement field directly which corresponds to strain fields which are as close as possible to the ones computed by interpolation. We use an optimization method which attempts to minimize W given by

$$W = \frac{1}{2}\int (\varepsilon'^t - \varepsilon^t)^T \cdot (\varepsilon'^t - \varepsilon^t) d\Omega. \tag{4}$$

Here, Ω is the whole domain of source object, ε^t is the desired (interpolated) strain field, and ε'^t is calculated from the position field using equation (1). The position field which minimizes W is chosen as the position field for this intermediate frame. For simplicity, in the following, ε^t is written as ε, and ε'^t written as ε'.

We use the *finite element method* (FEM) to calculate the strain field of an arbitrary object.[15,16] In fact, the meshes used for strain field interpolation method need not be triangular meshes, but could also be quadrilateral meshes, or based on other kinds FEM elements. In FEM, the position of a point inside an element can be expressed using a convex combination of the positions of the element's nodes:

$$x = \sum_{k=1}^{l} N_k x_k, \quad y = \sum_{k=1}^{l} N_k y_k, \tag{5}$$

where N_k is the shape function of the element, and l is the number of vertices of the element. For a triangular element, N_k are barycentric coordinates within the triangle.

By substituting equation (5) into equations (1) and (4), equation (4) can be converted into a finite dimensional problem, whose variables are coordinates of all nodes in the desired intermediate frame. The problem of minimizing W can be solved by any optimization method, but for efficiency, we convert the optimization problem into an equation solving problem, as explained next.

Nonlinear Equation Solver

Only triangular elements are discussed here; similar ideas work for other kinds of elements. From equation (4), we obtain:

$$dW = \sum_{i=1}^{m} \int \left[d\varepsilon'^T \varepsilon' - d\varepsilon'^T \varepsilon \right] d\Omega_i, \tag{6}$$

where m indexes the triangles of the object, and d is the derivative operator.

In FEM, $d\varepsilon$ is usually expressed using the strain matrix B:

$$d\varepsilon^T = (B)^T dV_i, \tag{7}$$

where

$$V_i = \begin{bmatrix} x_1 & y_1 & x_2 & y_2 & x_3 & y_3 \end{bmatrix}^T, \quad B = [B_1\ B_2\ B_3], \text{ and}$$

$$B_{ij}^t = \begin{bmatrix} \partial x'/\partial x \cdot \partial N_j/\partial x & \partial y'/\partial x \cdot \partial N_j/\partial x \\ \partial x'/\partial y \cdot \partial N_j/\partial y & \partial y'/\partial y \cdot \partial N_j/\partial y \\ \partial x'/\partial y \cdot \partial N_j/\partial x + \partial x'/\partial x \cdot \partial N_j/\partial y & \partial y'/\partial y \cdot \partial N_j/\partial x + \partial y'/\partial x \cdot \partial N_j/\partial y \end{bmatrix} \tag{8}$$

Substituting equation (7) into equation (6), and letting the derivative of W equal zero, we can convert the optimization problem into a nonlinear equation solving problem:

$$\phi(V) = \sum_{i=1}^{m} \int (B^T \varepsilon' - B^T \varepsilon) d\Omega_i = 0, \tag{9}$$

where V is the position vector within the whole domain.

This nonlinear equation problem can be solved by the Newton-Raphson method. For this we need to calculate the derivative matrix K^t of ϕ, which is given by:

$$K^t = \sum_{i=1}^{m} \int \left[B^T B + G^T (S' - S) G \right] d\Omega_i, \tag{10}$$

where $G = [G_1 \quad G_2 \quad G_3], \qquad G_i = \begin{bmatrix} \partial N_i/\partial x \cdot I_2 \\ \partial N_i/\partial y \cdot I_2 \end{bmatrix},$

$$S = \begin{bmatrix} \varepsilon_x \cdot I_2 & \gamma_{xy} \cdot I_2 \\ \gamma_{xy} \cdot I_2 & \varepsilon_y \cdot I_2 \end{bmatrix}, \quad S' = \begin{bmatrix} \varepsilon'_x \cdot I_2 & \gamma'_{xy} \cdot I_2 \\ \gamma'_{xy} \cdot I_2 & \varepsilon'_y \cdot I_2 \end{bmatrix}.$$

Figure 5. Morphing between letters.

The appropriate Newton-Raphson method can now be expressed as:

$$K_l^t(V_{l+1}^t - V_l^t) = -\phi(V_l^t) \tag{11}$$

where l represents the iteration count.

The coordinate vector for the source frame is used as the initial value for iterative calculation of the vector for the first intermediate frame, and each subsequent intermediate frame is initialized using the coordinates of the previous frame. For each intermediate frame, we iterate until $\|V_{l+1}^t - V_l^t\| < \delta$, where $\|\cdot\|$ is the Frobenius norm, and δ is a given tolerance.

If the source shape and target shape differ greatly and the number of frames inserted is low, direct use of the Newton-Raphson method can lead to triangle flip. This problem can be avoided by using a standard extension to the Newton method (a continuation method).[17] We first construct a function family which approximates the original function to get a better initial value, then use the classical Newton-Raphson method to get the final result. We use the following replacement for equation (11):

$$K_l^t(V_{l+1}^t - V_l^t) = -\left[\phi(V_l^t) + \left(\frac{l}{N} - 1\right)\phi(V_l^{t-1})\right], \tag{12}$$

for the first $N-1$ iterations: $l = 1, 2, \ldots, N-1$ to provide a better initial value, and then use the original formula in equation (11) for subsequent iterations $l = N, N+1, \ldots$.

A simpler way of solving this problem, obviously, is to insert more frames. The increased computing time which results is much less than proportional to the number of added frames, as with more frames, the initial values provided by the previous frames become more accurate. Adding frames can sometimes even reduce the overall computing time.

Interpolation of Rigid Body Motion

We now have intermediate frames which ignore the rigid body motion. We now need to add the interpolated rigid body motion back. Each frame is rotated around the origin by the angle $R_i = R_{A^0B^0} + t_i \cdot R$,; and each frame is translated by the vector: $T_i = T_{A^0} + t_i \cdot T$.

In summary, the main steps for solving morphing problems using the strain field interpolation method are: (1) Translate and rotate the source and target objects to eliminate rigid body motion. (2) Calculate the final strain field from the source and target objects. (3) Interpolate the strain field for intermediate frames. (4) Solve equation 4 or equation 9 to get the position field for intermediate frames. (5) Translate and rotate the interpolated objects to include the rigid body motion.

Figure 5 is an example of alphabetical morphing: A–B–C. More complex examples can be seen in Figures 7 and 9. In Figure 9, each mesh has 115 vertices. 110 seconds were spent on computing 180 intermediate frames using equation (9) on a 1.7 *Ghz* Pentium 4 machine. In Figure 7, each quadrilateral mesh has 34 vertices. Four seconds were taken to compute 60 intermediate frames with our method, compared to 150 seconds used by the method of[11].

From Objects to Isomorphic Meshes

To be able to apply the above method, we need two isomorphic meshes covering the source and target objects (polygons). This problem has been investigated by[9,18], whose general idea is to mesh both polygons, and then map both polygons to a regular n-gon. The two triangulations over the n-gon are then intersected with each other to give a new triangulation. The main drawback of this method is the number of triangles increases greatly during the intersection step; this greatly increases the computing load when performing the morphing calculations. Praun[19] suggested another approach to create a compatible mesh. He first identifies a relatively small number of feature points on each mesh, then uses a tracing method to join the feature points on one mesh to give a set of patch boundaries, and also forms this patch structure on the other mesh. Then, for

each patch, parameterization and a resampling technique are used to create consistent meshes. However, for 2D meshes, resampling is not easy to use, as it often loses some details at the boundaries, which are very important in 2D.

We introduce a new hybrid method here for creating isomorphic meshes. In outline, first, feature points are identified, and then the shapes are dissected into topologically equivalent patches. For each pair of patches, bijective maps are created between the patch boundaries, and we parameterize the patches on a regular n-gon. We overlay each patch on the n-gon, just keeping the vertices of each patch. We then triangulate the inside of the n-gon using all the mapped vertices. This triangulation is then mapped back to the source and target patches. We now consider the steps in detail.

Choosing Points on Objects

Given a source object and a target object, we select some *inner points* from each polygon, some of which are used to divide both polygons into the same number of more-or-less convex patches, and others of which are internal to the patches and are used to control the mapping behavior. We then identify a set of corresponding points (*anchor points*), which include selected boundary points and inner points. We next select certain *boundary* anchor points, and all *inner* anchor points, as *feature points*. These are used to create the patch structure inside the object. In summary, there are four kinds of points: border points, inner points, corresponding points (anchor points), and feature points.

Triangulation and Patching

The next step is to triangulate each object using the inner points. We use the constrained Delaunay triangulation method[20] to create a triangular mesh, for which the object edges are used as constraints, and all inner and boundary points are used as vertices. After triangulation, a scan line algorithm is used to judge if the centroid of each triangle is inside the object. If it is outside the object, the triangle is deleted from the mesh.

We use the method in[19] to trace paths between feature points, to dissect the source shape and target shape into several topologically equivalent patches. Each pair of patches must be constructed from equivalent anchor points. No patch contains any anchor points inside. Dissecting the polygon into several patches is a very useful tool: it can convert a shape whose genus is not zero into several genus zero patches. For example, in Figure 6, *A* and *B* are both dissected into three patches of genus zero, allowing morphing between them to be performed in a controlled way. More importantly, dissection can split objects with large concavities into several less concave shapes. Large concavities, if not removed, seriously influence mesh generation later.

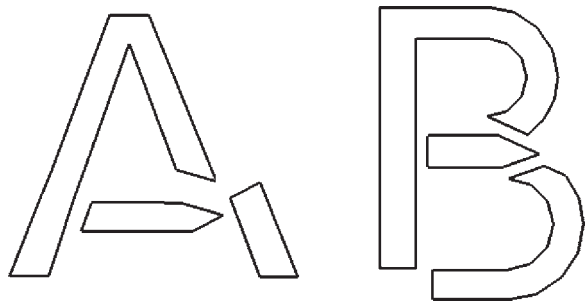


Figure 6. Letters A and B are decomposed into three equivalent patches.

Patch Parameterization

First, we create a consistent boundary for each pair of patches by the method of[5] or otherwise. To avoid problems of triangle flipping, *inner points* are added near any concave angles in the patch. Because new border points are added during the creation of consistent patch boundaries, and new inner points are added here, the patch must be re-triangulated. Then we parameterize the source and target patches onto a regular polygon surrounded by a unit circle. The mean value method proposed in[21] is used as the parameterization method.

We now merge the *points* from the source and target patch that have been parameterized on the n-gon. We ignore the *edges* of the triangulations. If the distance between two inner points is smaller than a threshold, we merge them to avoid producing small triangles after triangulation. We then triangulate the merged set of points.

Mapping Back

We next map the merged triangulation back into the source patch and target patch. Flipped triangles (with reversed orientation) may result, but this is usually because no or inadequate inner points were added near any concavities of the patches. We try to perform a flipping operation first to see if it can reverse the orientation and ensure the orientations in both patches

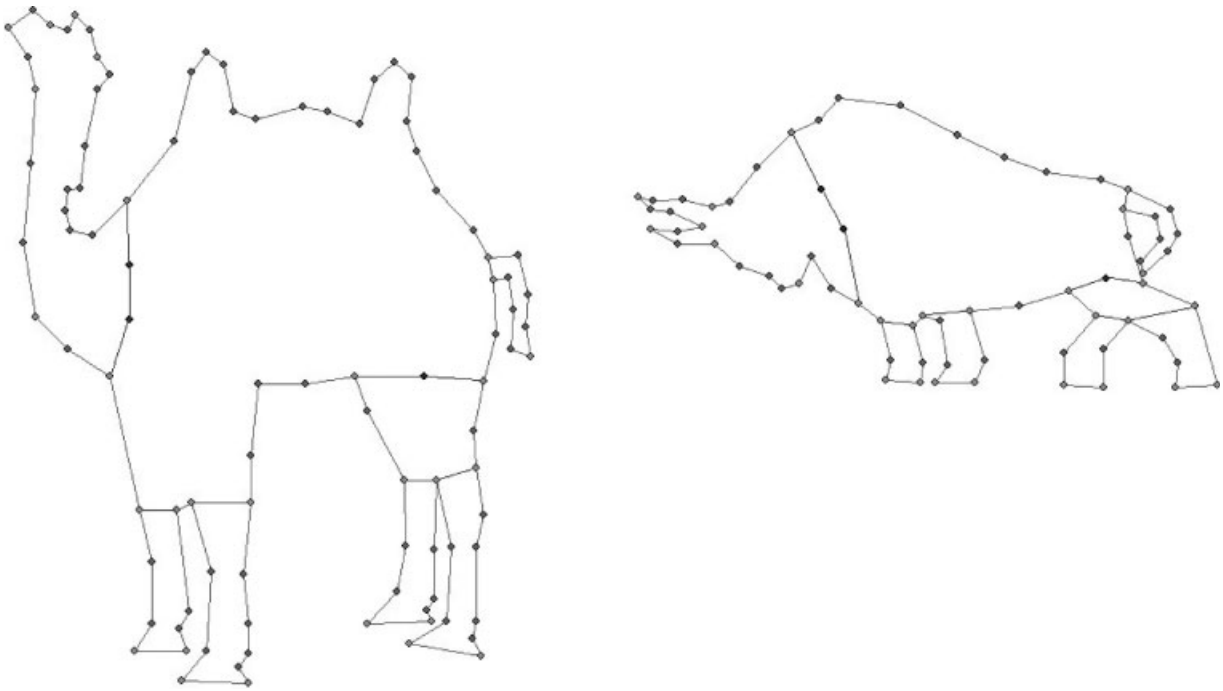


Figure 7. Morphing of a gymnast using quadrilateral meshes: (a) and (f) are source and target frames, (b)–(e) are intermediate frames.

are positive. Usually this is effective. If this does not work, more inner points must be added, and the updated patch re-triangulated. We then must repeat patch parameterization.

Merge all Patches and Optimization

Next, each pair of corresponding patches is consistently meshed. We merged the meshes from all patches, giving

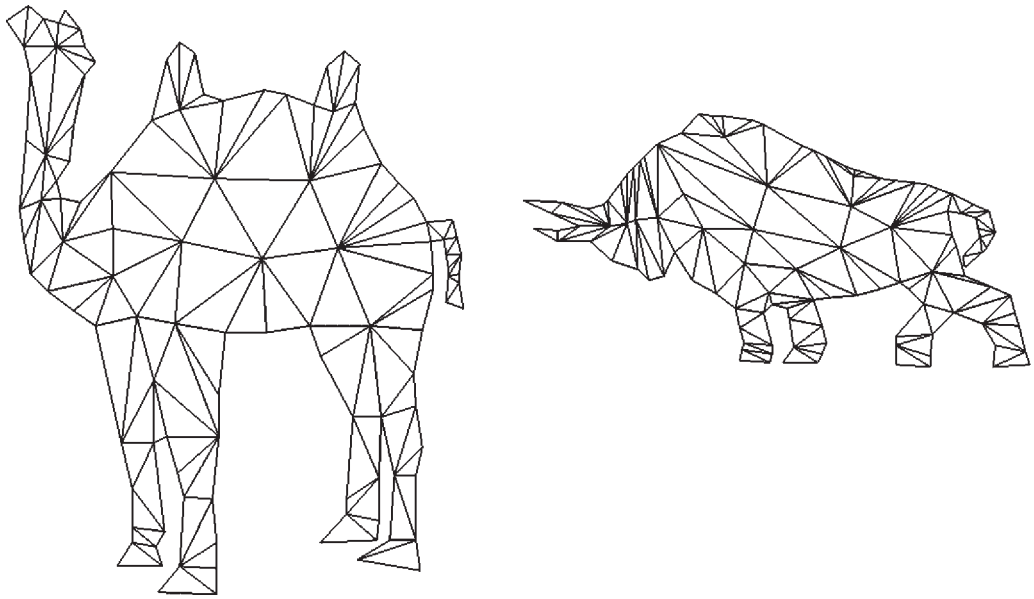


Figure 8. Isomorphic meshes.

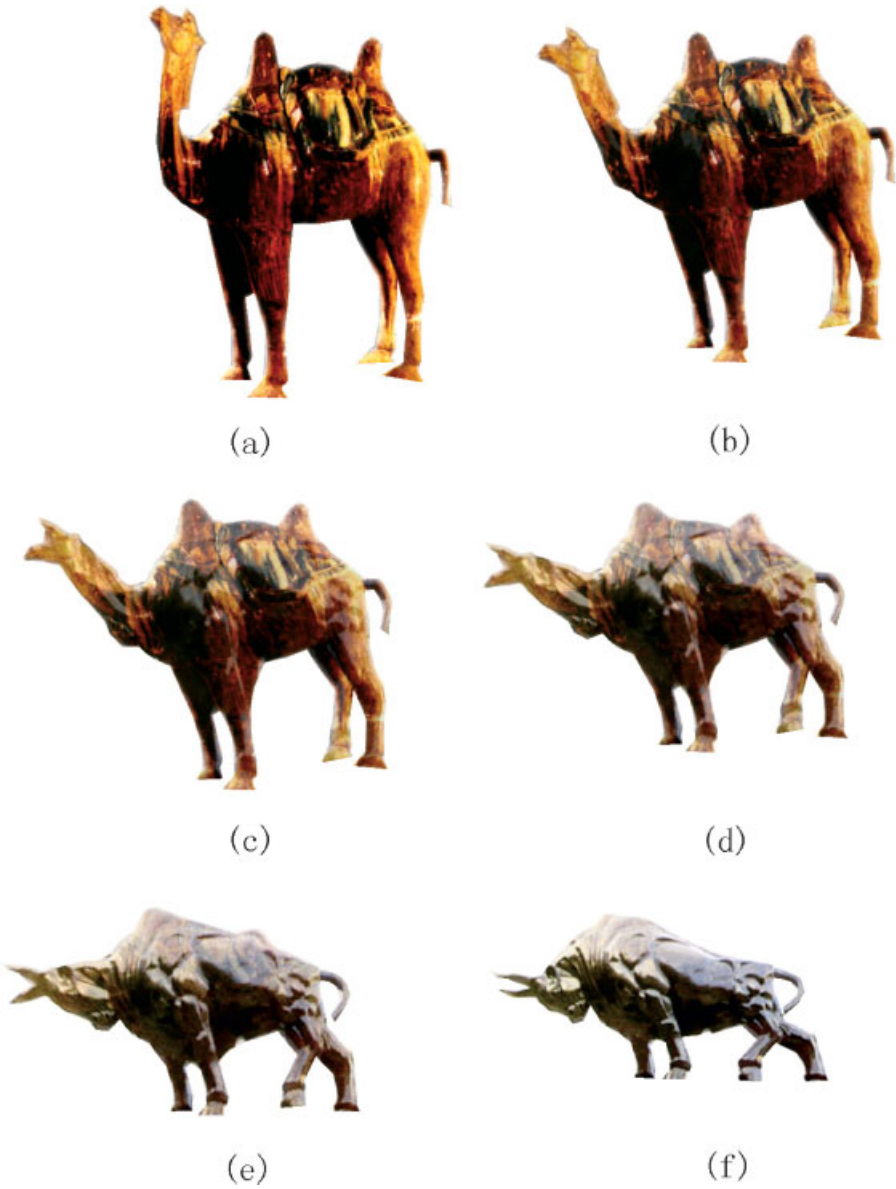


Figure 9. From camel to ox: (a) and (f) are source and target frames, (b)–(f) are intermediate frames calculated by our method.

isomorphic meshes for the source and target shapes. However, the quality of the merged triangulations still can be improved. We use the same operations as[9] to improve triangle quality: moving interior vertices and flipping interior edges.

The isomorphic meshes created by our new method contain many fewer triangles than produced by the method in,[9] becuase edge intersection is not used. To ensure that all triangles generated are valid, techniques of shape dissection and adding points near concavities are used.

If the meshes produced have many long thin triangles, more inner points can be added to improve the triangles' quality. Examples show that some long thin triangles do not, however, influence our later calculation seriously.

Conclusion and Future Work

In this paper, we presented new methods for natural morphing. The first contribution is a method to create

isomorphic meshes using many fewer triangles than previous methods based on edge intersection. Meshes with fewer triangles greatly decrease the computing time required in subsequent calculations. The second, and main, contribution is to use the concept of strain fields from physics to control object deformation in an intuitive way. We perform morphing based on strain field interpolation. Results show that our method is fast and also robust. Further examples are shown in Figure 7 illustrating morphing of a gymnastic figure, and Figures 8 and 9 showing morphing of a camel into an ox.

Although the examples provided in this paper are all two dimensional, this technique may very easily be extended to 3D using the same theory, based not only on tetrahedral volume meshes, but also on surface meshes. We intend to investigate this next.

ACKNOWLEDGEMENTS

We thank Chen Shaohua for sharing his knowledge of large deformations, and Zhang Wei for kind help. This work was supported by the Natural Science Foundation of China (Project Number 60225016), the Specialized Research Fund for the Doctoral Program of Higher Education (Project Number 20020003051) and the National Basic Research Project of China(Project Number 2002CB312100).

References

1. Wolberg G. Image morphing: a survey. *The Visual Computer* 1998; **14**(12): 360–372.
2. Bier T, Neely S. Feature-base image metamorphosis. *Computer Graphics* 1992; **26**(2): 35–42.
3. Lee SY, Wolberg G, Chwa KY, Shin SY. Image metamorphosis with scattered feature constraints. *IEEE Transactions on Visualization and Computer Graphics* 1996; **2**(4): 337–354.
4. Lee SY, Chwa KY, Hahn J, Shin SY. Image morphing using deformable surfaces. *Proceedings of Computer Animation'94*, 1994; 31–39.
5. Sederberg TW, Greenwood E. A physically based approach to 2D shape blending. *Computer Graphics* 1992; **26**(2): 25–34.
6. Sederberg TW, Gao P, Wang G, Mu H. 2-D shape blending: an intrinsic solution to the vertex-path problem. *Computer Graphics* 1993; **27**: 15–18.
7. Zhang Y. A fuzzy approach to digital image warping. *IEEE Computer Graphics and Applications* 1996; **16**(6): 34–41.
8. Shoemake K, Duff T. Matrix animation and polar decomposition. *Proceedings of Graphics Interface'92*, 1992; 258–264.
9. Alexa M, Cohen D, Levin D. As-rigid-as-possible shape interpolation. *Siggraph2000 proceedings*, 2000; 157–165.
10. Shapira M, Rappoport A. Shape blending using the star-skeleton representation. *IEEE Computer Graphics and Applications* 1993; **15**: 44–51.
11. Hu SM, Li CF, Zhang H. Actual morphing: a physical-based approach for blending. *ACM Symposium on Solid Modeling and Application' 2004*. http://cg.cs.tsinghua.edu.cn/en/research.htm, 2-19-2004.
12. DeCarlo D, Metaxas D. Shape evolution with structural and topological changes using blending. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1998; **20**(11): 1186–1205.
13. Liang Y, Bao H, Zhou W. A morphing with geometric continuity between two arbitrary planar polygons. *Pacific Graphics 2002 Proceedings*, 2002; 448–449.
14. Wilhelm F. *Tensor Analysis and Continuum Mechanics*. Springer-Verlag, c1972.
15. Zienkiewicz OC. *The Finite Element Method*. McGraw-Hill, 2000.
16. Belytschko T, Liu WK, Moran B. *Nonlinear Finite Elements for Continua and Structures*. John Wiley, 2000.
17. Ortega JM, Rheinboldt WC. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, 1970.
18. Aronov B, Seidel R, Souvaine D. On compatible triangulations of simple polygons. *Computational Geometry: Theory and Applications* 1993; **3**: 27–35.
19. Praun E, Sweldens W, Schroder P. Consistent mesh parameterizations. *Siggraph2001 Proceedings*, 2001;179–184.
20. http://www.cgal.org. 3-14-2004.
21. Floater MS. Mean value coordinates. *Computer Aided Geometric Design* 2003; **20**: 19–27.