Ya-Shu Liu, Han-Bing Yan, Ralph R Martin As-Rigid-As-Possible Surface Morphing. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY Vol.(No.): 1–end Mon. Year

As-Rigid-As-Possible Surface Morphing

Ya-Shu Liu[1](刘亚姝), Han-Bing Yan[2*](严寒冰), and Ralph R Martin[3]

[1]*Department of Computer Science, Beijing University of Civil Engineering and Architecture, Beijing 100044, China*

[2]*National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China*

[3]*School of Computer Science & Informatics, Cardiff University, Cardiff, UK*

E-mail: ly_s8020@163.com; yanhb02@mails.tsinghua.edu.cn; ralph@cs.cf.ac.uk

Received month day, year.

Abstract This paper presents a new morphing method based on the 'as-rigid-as-possible' approach. Unlike the original as-rigid-as-possible method, we avoid the need to construct a consistent tetrahedral mesh, but instead require a consistent triangle surface mesh and from it create a tetrahedron for each surface triangle. Our new approach has several significant advantages. It is much easier to create a consistent triangle mesh than a consistent tetrahedral mesh. Secondly, the equations arising from our approach can be solved much more efficiently than the corresponding equations for a tetrahedral mesh. Finally, by incorporating the translation vector in the energy functional controlling interpolation, our new method does not need the user to arbitrarily fix any vertex to obtain a solution, allowing artists automatic control of interpolated mesh positions.

Keywords Morphing, Simplex, Transformation, Interpolation

1 Introduction

Morphing, also called metamorphosis or shape blending, is a technique used to smoothly transform one graphical shape into another. Many 2D morphing methods have been developed to assist 2D animation making [1]. In recent years, with the introduction of 3D cartooning techniques, 3D morphing has increased in importance in games and animation production.

The most popular type of model used in 3D graphics is a surface triangle mesh. In this paper, we propose a new morphing method that works well even when the two input 3D triangle meshes have very different shapes. Many successful approaches to morphing use a framework based on

Regular Paper

∗Corresponding Author

firstly creating *consistent* meshes for the original source and target models, i.e. source and target meshes having one to one correspondences between their vertices, edges and faces. Given these, a path may then be determined for each vertex to follow during the morphing process. Much work has considered the first issue [2, 3, 4]; this paper focuses on the trajectory problem.

The approach used in this paper is based on the *as-rigid-as-possible warping* method [5] for generating a smooth morph between two quite different shapes. However, that method requires 3D objects to be represented as *volume, tetrahedral* meshes. Unfortunately, as is well known, meshing complex solids is not easy, and creating consistent tetrahedral meshes is notoriously difficult. Indeed, we are unaware of any satisfactory general solution to that problem.

Our new method has the following merits compared to the original as-rigid-as-possible method, while still providing very good 3D morphing results:

- Our method works directly with *surface triangle* meshes instead of requiring tetrahedral meshes representing the interior of each object.
- Creating consistent triangle meshes is a much *easier* problem to solve than creating consistent tetrahedral meshes.
- The number of vertices in consistent triangle meshes is far *fewer* than in corresponding tetrahedral meshes of the same objects at the same resolution, resulting in greatly *reduced* computing time.
- By incorporating the translation vector into our energy functional, we do *not* need to fix some arbitrary vertex when computing the solution; the original method needs user selection of at least one fixed vertex.

The inputs to our method are a source mesh and a target mesh, in the form of consistent triangle meshes, and the desired number of intermediate frames. The output is a sequence of intermediate meshes forming a morphing sequence.

Figure 1 illustrates results produced by linear interpolation, Alexa's original as-rigid-as-possible volume warping method [5], and our new surface-mesh-based method. Our method produces almost identical results to Alexa's volume-based morphing method, but at much lower computational cost, and without the need to produce volume meshes and make them consistent.

We note that [6] uses ideas somewhat similar to ours to solve the deformation problem. Nevertheless, there is a significant difference. In our method, we incorporate the translation vector into the error function, avoiding the need for the user to fix the location of any vertex in the solution process. Appropriate choice of the vertex to fix is not a simple task, so our approach simplifies the user interface for the user. Nevertheless, in the degenerate case, our error function corresponds to the form not using a translation vector, allowing the artist to fix any vertex if desired, giving the artist the freedom to choose how much explicit control to use.

2 Related Work

Methods for 3D morphing typically take one of two approaches. The first blends simpler volumes

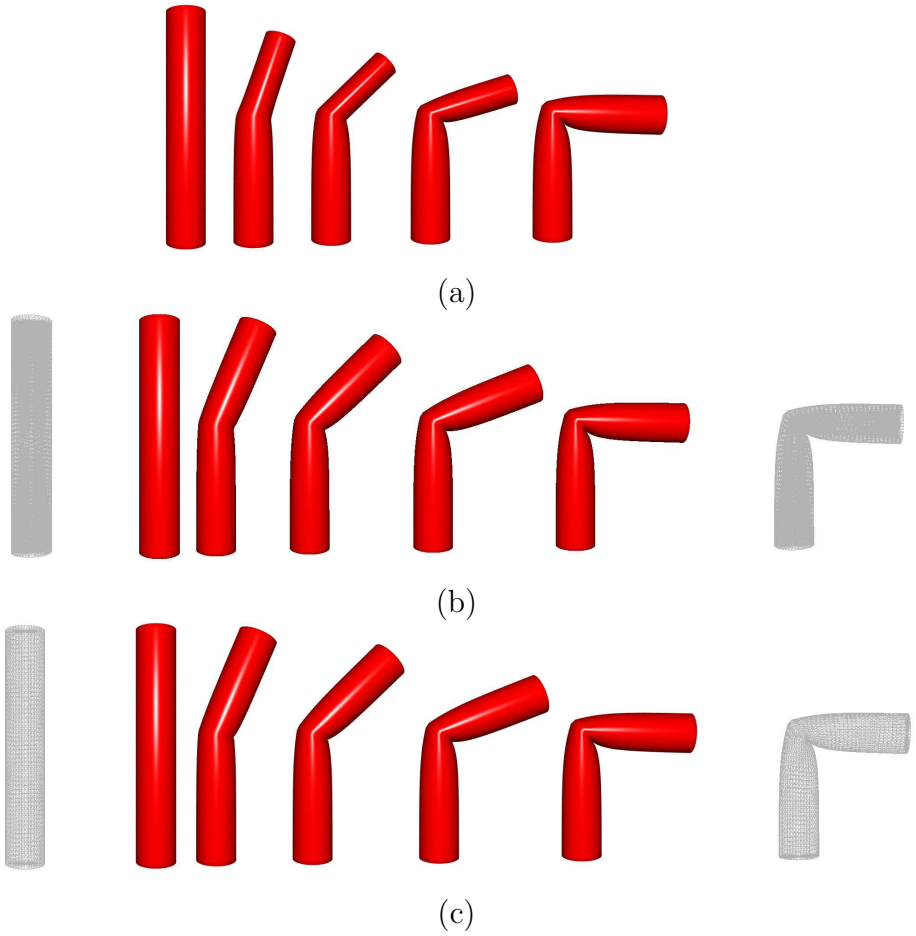


Fig 1: Morphing comparison using a total of 5 frames: (a) morphing using linear interpolation, (b) morphing using Alexa's method [5],(c) morphing using our method. In (b), initial and final tetrahedral meshes are shown in gray, while in (c) initial and final triangle meshes are shown.

into which the initial and final shapes have been embedded [7, 8]. The second directly manipulates an explicit geometric object representation, typically a surface mesh or volume mesh [2, 5]. The first class of approach has the advantage of being able to morph objects having different topologies. However, mesh-based methods typically produce better results—often, shape boundaries resulting from use of embedding methods are not smooth enough. Thus, the focus of morphing has shifted towards mesh-based approaches in recent years. Our method is based on use of surface triangle meshes.

As noted earlier, explicit surface or volume mesh morphing typically uses two main steps: creating *consistent* meshes, and determining vertex *trajectories*. We first briefly review the former, then consider the latter in more detail as it is the main focus of this paper.

Various work has considered how to create *consistent* meshes for pairs of shapes of genus zero, using a topological merging method. A frequent approach is to first dissect the source and target shapes into several pieces [9], then to construct a

local parameterization for each piece, and finally to perform merging [4, 10] or remeshing [2, 11] to create the consistent mesh. Praun [12] gives a tracing method which can dissect source and target shapes automatically.

Many of the above papers concentrate on the problem of creating consistent meshes, and only use simple linear vertex coordinate interpolation methods to find vertex trajectories. For source and target objects with very similar shapes, linear interpolation is sufficient to produce simple visual effects. However, if objects undergo large deformations, especially bending, linear interpolation always leads to visually unacceptable shrinkage of intermediate shapes—see, for example, Figure 1(a).

Using global Euclidean coordinates at each vertex does not capture local shape information and vertex connectivity information directly. Some work thus tries to interpolate other intrinsic representations based on *local* shape information, including barycentric coordinates, Laplacian coordinates, and other quantities which represent local intrinsic attributes of the mesh, as we now discuss.

Floater and Gotsman [13] used barycentric coordinate interpolation to find suitable paths for convex 2D shapes. This method has been extended concave shapes by embedding them into a convex shape [14]. Ju et al. [15] used mean value coordinates (a generalisation of barycentric coordinates) to control 2D shape deformation. The same principles can also be used for 3D morphing. Mean value coordinates are invariant under translation and rotation but are not invariant under scaling, restricting their usefulness.

Alexa et al. [16] suggested interpolation of Laplacian coordinates to determine morphing trajectories and discussed how to control morphing locally. Laplacian coordinates are invariant under translation but are not under rotation and scaling. Later work [17, 18] has considered how to modify interpolated Laplacian coordinates to provide better morphing results. Recently, Hu [19] extended this approach by interpolating the curvature flow Laplacian operator.

Sheffer and Kraevoy [20] introduced so-called pyramid coordinates into mesh editing and morphing. While these coordinates are rotation-invariant, reconstruction of Cartesian coordinates from them requires time consuming non-linear optimization. Lipman [21] proposed rotation invariant differential coordinates by considering tangential and normal components relative to the surface. Their method requires the solution of two linear systems for each intermediate frame, and hence roughly twice the amount of work of other similar methods.

Yu et al. [22] introduced a technique which interpolates gradients of the surface mesh; reconstruction of in-between surfaces requires the solution of a Poisson equation. These gradients are again not invariant under rotation, so must be modified when rotation is involved, using a similar method to that used for Laplacian coordinates. Xu et al. [23] generalised this technique for use in morphing. They calculate intermediate surface gradients by quaternion interpolation, and reconstruct surfaces by solving a Poisson equation. This method produces good results in many cases.

Another important morphing method is the *as-rigid-as-possible* approach, which can handle source and target shapes with large differences.

Shoemake [24] proposed the use of a global affine transformation to represent distortion between source and target shapes. By decomposing the transformation matrix into separate rotational and irrotational parts, and interpolating them independently, the overall transformation matrix and shape of each intermediate frame can be calculated. Alexa [5] developed this idea to determine the local transformation separately for each mesh element (interior triangles for 2D morphing or interior tetrahedra for 3D morphing) instead of the whole shape, then used optimization to minimize the difference between the desired transformation, and the actual transformation which is applied, taking into account the connectivity constraints on adjacent mesh elements. This method uses a tetrahedral *volume* mesh to solve 3D morphing problems. However, most models used in animation are represented using *surface* triangle meshes. It is a difficult problem to generate a corresponding solid tetrahedral mesh from a triangle mesh, especially one with well-shaped elements. Obviously, it is even more difficult to create consistent meshes for source and target shapes, making Alexa's method very hard to use in practice for 3D morphing. Simplex transformation has also been used with surface triangle meshes to perform deformation learnt from existing examples [25].

Hu et al. [26] presented a method based on minimization of deformation energy, which is novel in that it does not use interpolation, but is a global optimization method. Yan et al. [27, 28] proposed the use of strain fields from mechanics for smooth interpolation. This method can provide very uniform results even if the source and target meshes are quite different, but it requires the solution of a nonlinear equation and again, it is not easy to create consistent tetrahedral meshes in 3D. Bao et al. [29] also used a similar idea for point cloud morphing.

Although many morphing methods use different algorithms, they are typically guided by the same physical principle—that of keeping the shape as rigid as possible as it changes. This concept has also been used in geometry editing and parametrization [30, 31, 32].

In this paper, we present a new method based on Alexa's *as-rigid-as-possible* method [5]. Our modification allows this method to be applied to surface meshes directly without the need for construction of consistent tetrahedral meshes. As a further consequential benefit, our method requires much less computation than Alexa's original method. Our method can be used in cases involving large rotations, and no arbitrary vertex needs to be fixed by the user in order to compute a solution to the morphing problem.

We explain our method in Section 3 and demonstrate results in Section 4, finally drawing conclusions in Section 5.

3 Surface Morphing

Our approach to the morphing problem builds on the work of Alexa [5]. We presume that *consistent* source and target surface triangle meshes have already been computed. To determine trajectories, the basic approach is as follows: we calculate the transformation matrix and translation vector relating the initial and final disposition of each mesh element. The transformation matrices are further decomposed into rotational and irrotational components, each of which is interpolated

separately. Finally, we reconstruct each intermediate mesh from the interpolated transformation matrices and translation vectors. Our method differs from that of Alexa [5] in two ways. First, we do not need to create consistent tetrahedral meshes, but calculate each transformation matrix simply using information from the surface triangle meshes (we generate a set of surface tetrahedra, as we explain shortly). Secondly, we use a different error energy function, taking into account the translation, so no vertex needs to be fixed in our method.

3.1 Transforming Triangles

We now consider transformation of mesh triangles. An n-simplex is the simplest possible n-dimensional polytope in n-dimensional Euclidean space: in 2D and 3D, these are triangles and tetrahedra respectively. Given two such n-simplices S_1 and S_2 in nD space, there exists a unique transformation that changes S_1 into S_2. If u_i are the vertices of S_1, and v_i are the corresponding vertices of S_2, this can be written:

$$v_i = Mu_i + T, \tag{1}$$

where M is an affine transformation matrix representing rotation, scaling and shearing, and T is a translation vector. M and T are determined by the vertex coordinates of S_1 and S_2, as follows:

$$M = VU^{-1}, \tag{2}$$

where in 2D,

$$V = \begin{bmatrix} v_1 - v_3 & v_2 - v_3 \end{bmatrix}, \tag{3}$$
$$U = \begin{bmatrix} u_1 - u_3 & u_2 - u_3 \end{bmatrix},$$

and in 3D,

$$V = \begin{bmatrix} v_1 - v_4 & v_2 - v_4 & v_3 - v_4 \end{bmatrix}, \tag{4}$$
$$U = \begin{bmatrix} u_1 - u_4 & u_2 - u_4 & u_3 - u_4 \end{bmatrix}.$$

Having computed M, it can be substituted into Eqn. 1 to find T.

Our morphing method adds a vertex per mesh triangle to create a single tetrahedron for each initial and final triangle mesh face. The transformation matrix relating each such pair of corresponding tetrahedra can then be calculated. After decomposing the transformation into rotational and irrotational parts, we interpolate these two parts separately to get a desired transformation matrix for each intermediate frame. Finally, we use an optimization method to ensure the connectivity of adjacent triangles.

We use similar ideas to those in Sumner [33] to construct these surface tetrahedra. For each mesh triangle of the source and target meshes, we add a fourth vertex, and connect it to the triangle to give a tetrahedron. The fourth vertex is added at a certain distance along the normal direction over the triangle's centroid. Let v_1, v_2, v_3 be the vertices of a triangle on the mesh. The fourth vertex is placed at

$$v_4 = \frac{(v_1 + v_2 + v_3)}{3} + \frac{(v_2 - v_1) \times (v_3 - v_2)}{\sqrt{|\,(v_2 - v_1) \times (v_3 - v_2)\,|}}. \tag{5}$$

By putting the new vertex over the centroid, and by letting the distance from the new vertex to the triangle centroid be proportional to the square root of the triangle area, we create a tetrahedron that is as similar as possible to a regular tetrahedron. This ensures that the calculation of the transformation between a source and target tetrahedron is as well-conditioned as possible. It also ensures the tetrahedron changes shape as little as possible, in accordance with the spirit of the as-rigid-as-possible concept.

From these tetrahedra, we can calculate the transformation matrix M for each pair of corresponding tetrahedra using Eqn. 2, and afterwards the translation vector T using Eqn. 1.

3.2 Interpolation

Given the initial and final transformations for each pair of tetrahedra, and thus for the underlying mesh triangles, we now interpolate the transformation matrix M and the translation vector T for each tetrahedron to get the desired transformation matrix and the desired translation vector for each vertex of each triangle, for each intermediate model.

We interpolate the transformation matrix using the method in [5]. First the transformation is decomposed by singular value decomposition giving

$$M = PDQ, \tag{6}$$

which may also be written

$$M = P(QQ^T)DQ = RS, \tag{7}$$

where $R = PQ$ and $S = Q^T DQ$. R is an orthogonal matrix representing rotation, and S represents the irrotational part of the transformation. We now interpolate the rotation matrix R and irrotational matrix S separately. The latter is interpolated by simple linear interpolation to give $\widetilde{S}(t)$, the desired irrotational matrix at time t (where t is normalised to the range 0 to 1):

$$\widetilde{S}(t) = I\,(1-t) + S\,t. \tag{8}$$

Linear interpolation is appropriate for the shears and scalings represented by S.

We use quaternions to interpolate the rotation matrix R, a widely used approach. A quaternion may be written $Q = [w, x, y, z]$. A unit magnitude quaternion $w^2 + x^2 + y^2 + z^2$ can also be written as $[\cos(\theta/2), \mathbf{n}\sin(\theta/2)]$ and represents a rotation with axis $\mathbf{n}$ through an angle θ. Spherical interpolation of unit quaternions is an effective tool for performing rotation interpolation. Details of quaternion interpolation, and conversion between rotation matrices and unit quaternions can be found in [34]. We transform the rotation matrix R to a unit quaternion Q, then interpolate between a quaternion $Q_0 = [1, 0, 0, 0]$ representing no rotation at time $t = 0$, and a quaternion Q representing the final rotation at time t. The result is $\widetilde{Q}(t)$ which we convert back to a rotation matrix $\widetilde{R}(t)$.

Thus, the desired transformation matrix at time t is

$$\widetilde{M}(t) = \widetilde{R}(t)\widetilde{S}(t), \tag{9}$$

The desired translation vector at time t is simply interpolated by linear interpolation, representing a constant velocity of motion.

$$\widetilde{T}(t) = T\,t, \tag{10}$$

3.3 Optimization to Find Mesh Vertices

We now show how to compute the new mesh at time t from the desired translation vectors and transformation matrices.

Remember that a different transformation is determined for each triangle independently. As a result, gaps or overlaps would occur between triangles if each triangle were simply transformed independently according to its own interpolated transformation matrix $\widetilde{M}(t)$ and translation vector $\widetilde{T}(t)$. The key to retaining connectivity of adjacent triangles is that each vertex belongs to more

than one triangle but can have only a single transformation. We thus determine transformations for vertices so that each triangle has a transformation as close as possible to that determined by its interpolated transformation matrix and translation vector. We do so using our optimization method from [35].

Let $\widetilde{M}_i(t)$ and $\widetilde{T}_i(t)$ represent the interpolated transformation matrix and translation vector of the i^{th} triangle. An error function taking into account the difference between the actual state and the desired interpolated state is defined as follows:

$$E = \sum_{i=1}^{n} A_i(\|M_i^* - \widetilde{M}_i\|_F^2 + \alpha\|T_i^* - \widetilde{T}_i\|_2^2) \quad (11)$$

where n is the number of simplices in the mesh, M_i^* is the actual transformation matrix for the i^{th} triangle, and T_i^* is the actual translation vector of the i^{th} triangle. F denotes the Frobenius norm. A_i is the area of the i^{th} triangle: large triangles are more visible so should provide a greater contribution to the error function. α is the square of the reciprocal of the diagonal length of the scene bounding box, which is used to ensure both terms are comparable, and have the same units of measurement.

We minimize the error energy E to get the best intermediate shape while ensuring that adjacent triangles remain connected. The variables being optimised over in Eqn. 11 are the vertex coordinates for the intermediate shape.

As noted in [33], such a quadratic optimization problem can be decomposed into 3 independent optimization problems for x, y and z. Each subproblem can be transformed into a linear system by setting the gradient of E to zero, giving three independent linear systems:

$$KX = b_x, \quad KY = b_y, \quad KZ = b_z, \quad (12)$$

where X, Y and Z are the x, y and z coordinate vectors of the interpolated mesh; these have size $m + k$, where m is the number of vertices in the mesh, and k is the number of triangles. K is a sparse matrix of size $(m+k) \times (m+k)$, and b_x, b_y and b_z are vectors of size $m + k$. This decomposition into 3 smaller linear systems allows the solution to be found more efficiently. We use direct Cholesky decomposition and back substitution to do so. By now, we get a tetrahedra mesh sequence. When rendering, we only keep the surface triangle where ignore the fourth vertex located on each surface triangle.

An important difference between our method and that in [5] is that we incorporate the translation vector as the second term in the error function in Eqn. 11, in addition to the shape error represented by the first term. Without this extra term, the matrix K in Eqn. 12 would be singular, but adding this term ensures that the linear system has a unique solution. Alexa's original method uses an alternative method to avoid singularities, which is to fix the positions of one or more vertices. However, this requires user input, and it is not always easy for the user to make appropriate choices.

We use the coefficient α to control the relative importance of the position error. By making the position error small compared to the shape error, it weakly restricts each triangle shape, while allowing it to decide the global mesh position. Experiments show that if the normalising factor α is not used, morphing results may be poor if displacements are large. On the other hand, α should not be too small, to avoid the equations becoming

ill-conditioned. We have performed many experiments showing that our choice of α above consistently provides satisfactory results. Our experiments show that interpolated shapes determined by our method differ little from those found by Alexa's approach of fixing a suitable arbitrary vertex, and mainly differ in model positions—see, for example, Figure 2.

In more detail, in Alexa's original method [5], the location of (at least) one vertex must be determined by the user, in each frame. In the simplest case, its position may be found in intermediate frames by interpolating its position in the initial and final frames. Different choices of this fixed vertex may lead to differing positions of the model in intermediate frames, and poor choices may produce unsatisfactory animation results, leading in bad cases to unacceptable wobbling and jittering of the model's perceived location. Our method has no such problems, as no vertex needs to be fixed; the source *smoothly* changes globally into the target both in shape *and* in position. Figure 2(a) shows a morphing result based on Alexa's approach of fixing the position of a vertex in each frame (marked by the blue sphere in the first image). Figure 2(b) shows the corresponding morphing result if instead the translation vector is incorporated. In Figure 2(a), it can be seen that the hand appears to move backwards in the intermediate frames and then forwards again (which is why the hand shape appears smaller in the intermediate results). In Figure 2(b) the intermediate results differ little in shape from those in Figure 2(a) but are more stable in position. (The viewpoint is the same for all frames.)

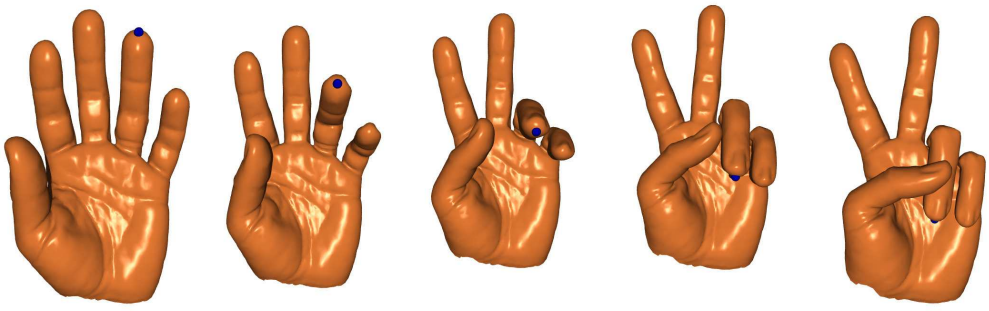
Nevertheless, we note that in cases where the artist wants to explicitly control the intermediate mesh positions, instead of relying on an automatic method, we can easily set $\alpha = 0$ in Eqn. 11, and allow the artist to fix any vertex as desired, as in Alexa's method.

4 Results

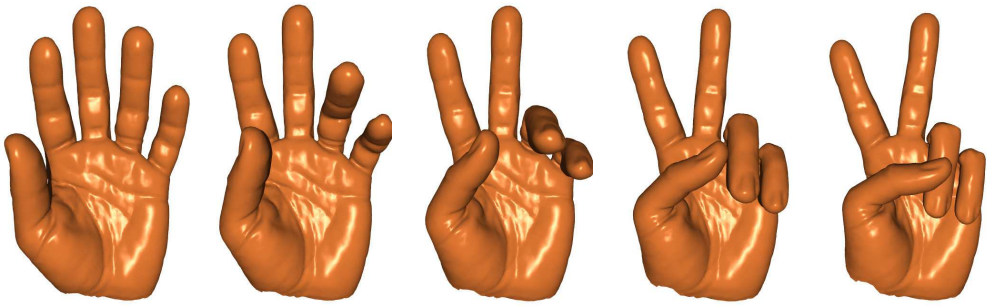
We have applied our method to various models, both morphing between two different objects, and performing morphing corresponding to various fundamental deformations of a single object. Typical experimental results are shown in Figures 2–6. In each figure, the first and last images show the given source and target meshes, while intermediate images show morphing results produced by our method. All experiments were performed on a PC with a 3.2GHz Pentium 4 CPU with 1GB memory. Table 1 shows the times taken per frame to compute the intermediate models illustrated in this paper.

Experiments show that the morphing results generated by our method are smooth and natural, and that no shape jittering or wobbling occurs, which may be a problem for other morphing methods relying on a fixed vertex.

Our method can also be easily modified for use in the solution of $2D$ polygon morphing problems, turning polygon edges into virtual triangles in a manner analogous to the 3D case. After adding a virtual vertex on each edge, the transformation relation between each pair of virtual triangles is calculated, and the $3D$ equations of this paper can be directly replaced by $2D$ equaivalents in an obvious manner. Figure 7 shows a $2D$ polygon morphing result produced by our method, in which each polygon had 196 vertices.



(a)



(b)

Fig 2: Hand: (a) not using translationg vector(letting $\alpha = 0$ in Eqn. 11) and one vertex is fixed(blue sphere), (b) using translation vector, no vertex need to be fixed.

	Hand	Armadillo	Bunny to Rabbit	Camel to Horse	2-Torus to Vase
Number of vertices	12782	165954	14221	8431	13998
Time per frame	1.28s	20.34s	1.45s	0.79s	1.30s

Table 1: Timing information

5 Conclusions

This paper has shown how to make Alexa's morphing method much more practical, requiring only surface meshes rather than volume meshes. Our approach not only avoids the complex process of consistent tetrahedral meshing, but also lowers the computation time, since surface triangle meshes have many fewer vertices than corresponding volume tetrahedral meshes. Secondly, unlike many other morphing methods using intrinsic representations, our approach does not need to fix any vertex during the solution process, and provides

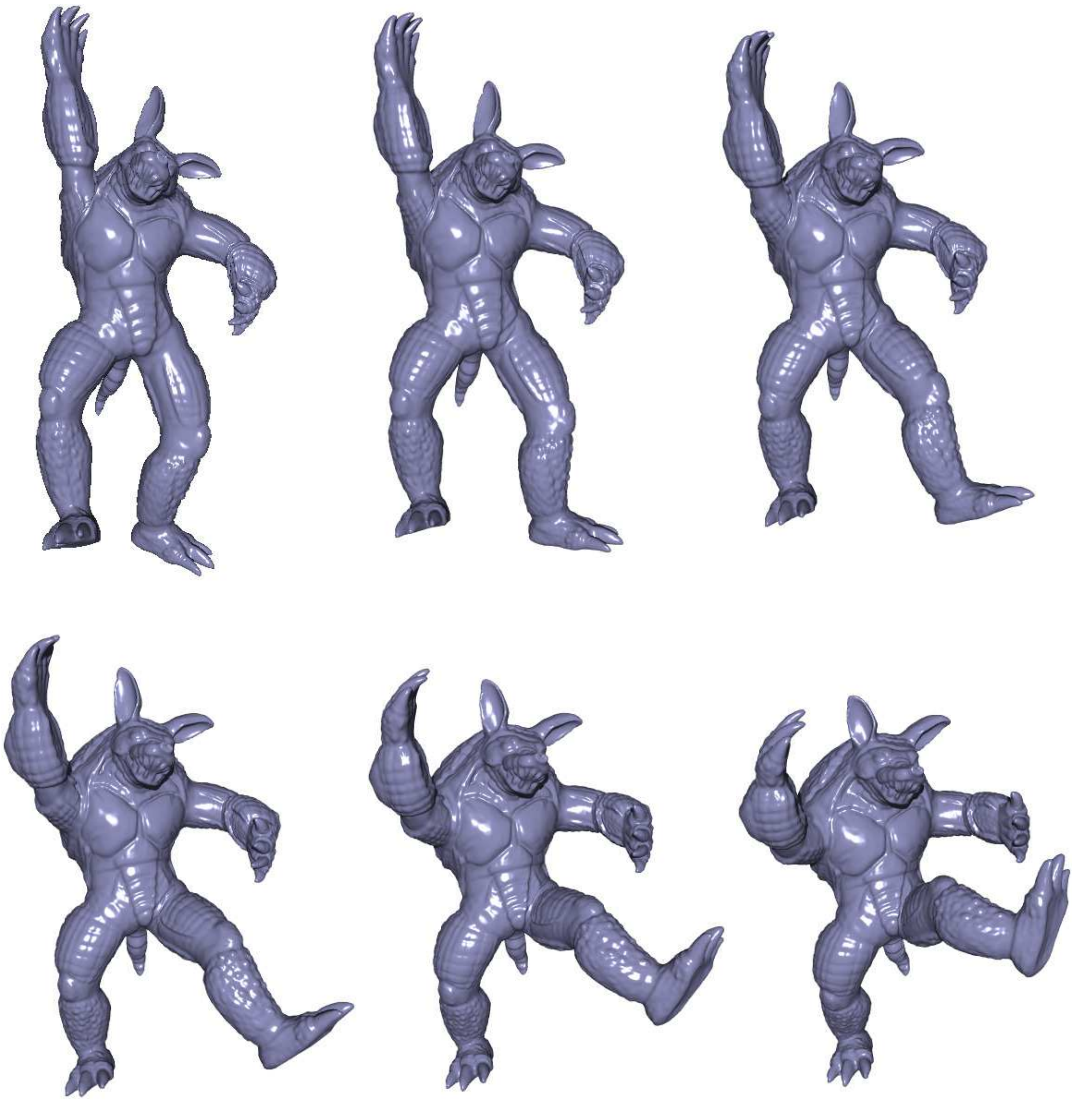


Fig 3: Armadillo.

automatic control of mesh position. Our method requires the solution of 3 linear systems which can be done very rapidly using direct Cholesky decomposition. Our method can also be easily modified for use in the solution of $2D$ polygon morphing problems. In summary, our method is of wide applicability.

Except compared with Alexa's mothod [5] as in figure 1, our method is also compared with Xu's method as in figure 8. It is obvious that our method has very similar morphing results compared to Alexa and Xu's methods if proper vertex is fixed in the those two methods. Compared to Alexa's method, our method highly increase the solving efficiency and does not need complex solid mesh. Though Xu's method and ours has similar solving efficiency, our method does not need to fix any vertex and much simplified the artist opera-

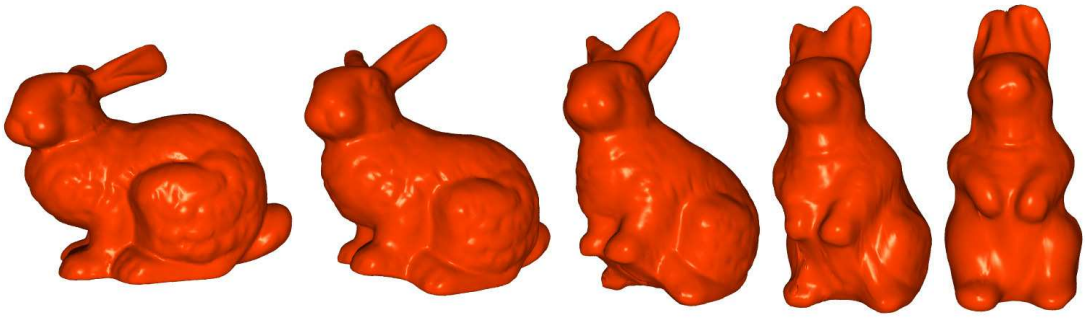


Fig 4: Bunny into Rabbit.

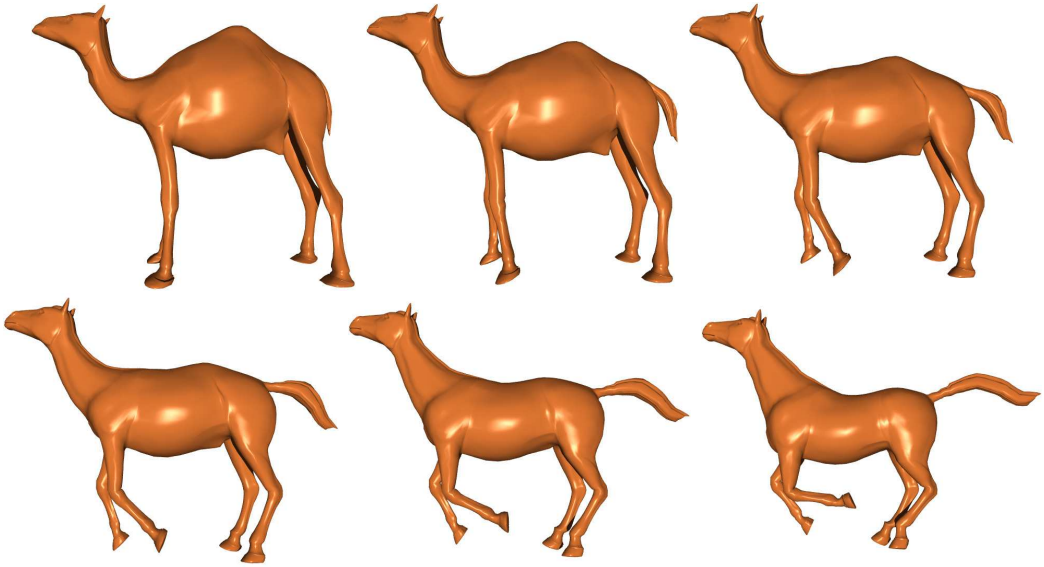


Fig 5: Camel into Horse.

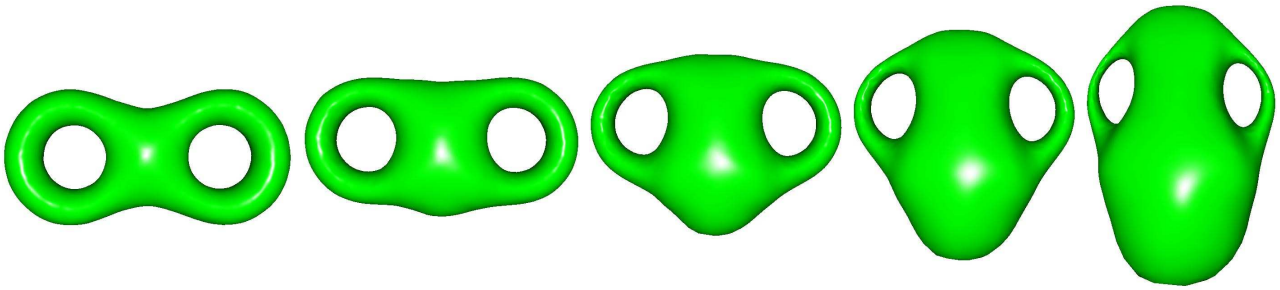


Fig 6: 2-Torus into Vase.

tion, while Xu's method need to fix at least one vertex and may suffer from the fixed vertex selection problem as in figure 2.

The main limitation of our work is inherited from [5] and shared by [23]: the largest rotation angle may not exceed π, due to the use of quater-

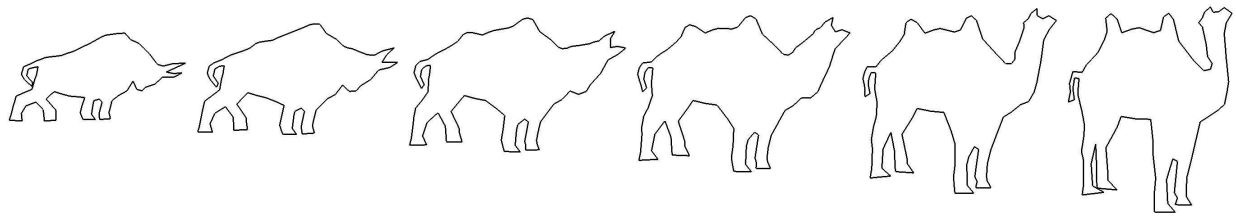
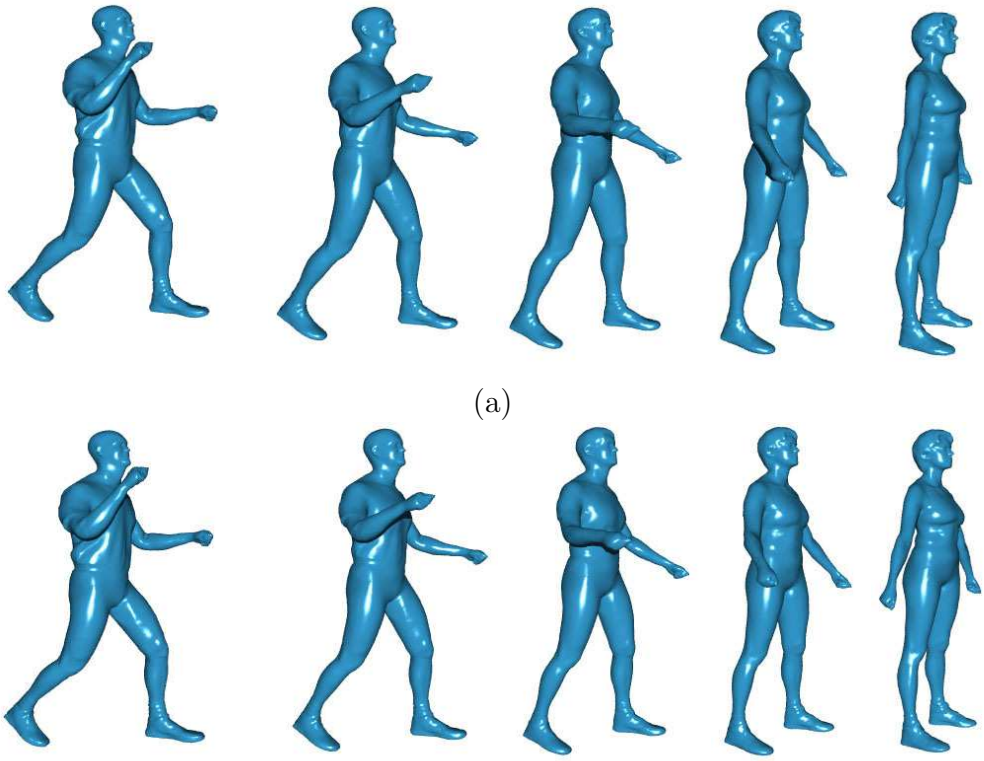


Fig 7: 2D polygon morphing: Ox into Camel.

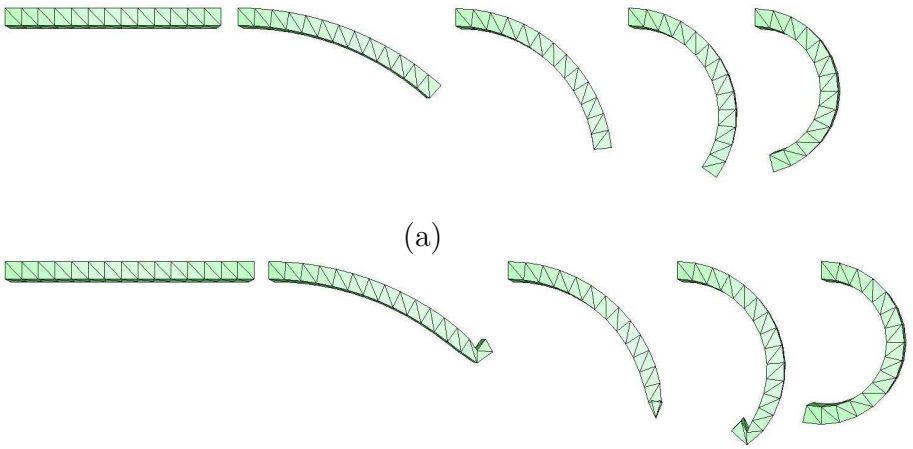


(a)

(b)

Fig 8: Man into Woman.(a) Morphing using our method, (b) Morphing using Xu's method [23]. The results of the two methods are very similar.

nion interpolation. Figure. 9 give 2 similar 3D examples but the largest rotation angle is a little different. In Figure. 9(a), our method works well where the largest rotation angle is a little less than π. In Figure. 9(b), the largest rotation angle is a little larger than π, and the morphing result is in error since the quaternion interpolation cannot distinguish angle larger or smaller than π. A simple way to overcome the analogous problem in 2D is given in [36, 37], but it cannot be extended to 3D. The morphing method in [28] can be used in cases where the rotation angle exceeds π, but it requires the solution of a nonlinear system, which is less efficient. We intend to further investigate



(a)

(b)

Fig 9: (a) The Largest Rotation Angle Less Than π, (b) The Largest Rotation Angle Larger Than π.

removing this restriction.

Acknowledgements The authors would like to thank AIM@SHAPE, Prof. Tong-Yee Lee, Dr. Dong Xu, Prof. Hong-Xin Zhang, Prof. David Xian-Feng Gu and Prof. Xin Li for providing the models used in the paper. This work was supported by the National Science Foundation of China(Project Number 61003132), the EPSRC Travel Grant, the Technology Porject of MOUHURD of China(Project Number 2010-K9-25) and the Development Project of BMCE(Project Number KM200710016001).

References

[1] George Wolberg. Image morphing: a survey. *The Visual Computer*, 14(8/9):360–372, 1998.

[2] Hujun Bao and Qunsheng Peng. Interactive 3d morphing. *Computer Graphics Forum*, 17(3):23–30, 1998.

[3] Marc Alexa. Merging polyhedral shapes with scattered features. *The Visual Computer*, 16(1):26–37, 2000.

[4] Tong-Yee Lee and Po-Hua Huang. Fast and intuitive metamorphosis of 3d polyhedral models using smcc mesh merging scheme. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):85–98, 2003.

[5] Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid as-possible shape interpolation. In *Proceedings of SIGGRAPH 2000*, pages 157–164, 2000.

[6] Robert W. Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popovi'c. Mesh-based inverse kinematics. *ACM Transaction on Graphics*, 24(3):488–495, 2005.

[7] Daniel Cohen-Or, Amira Solomovici, and David Levin. Three dimensional distance field metamorphosis. *ACM Transaction on Graphics*, 17(2):116–141, 1998.

[8] Xiang Fang, Hujun Bao, Pheng-Ann Heng, Tien-Tsin Wong, and Qunsheng Peng. Continuous field based free-form surface modeling and morphing. *Computer and Graphics*, 25(2):235–243, 2001.

[9] Takashi Kanai, Hiromasa Suzuki, and Fumihiko Kimura. Three-dimensional geometric metamorphosis based on harmonic maps. *The Visual Computer*, 14(4):166–176, 1998.

[10] Arthur Gregory, Andrei State, Ming C. Lin, Dinesh Manocha, and Mark A. Livingston. Feature-based surface decomposition for correspondence and morphing between polyhedra. In *Proceedings of the Computer Animation 98*, pages 64–71, 1998.

[11] Jin-Bey Yu and Jung-Hong Chuang. Consistent mesh parameterizations and its application in mesh morphing. In *Computer Graphics Workshop 2003*, 2003.

[12] Emil Praun, Wim Sweldens, and Peter Schroder. Consistent mesh parameterizations. In *Proceedings of SIGGRAPH 2001*, pages 179–184, 2001.

[13] Michael S. Floater and Craig Gotsman. How to morph tilings injectively. *Journal of Computational and Applied Mathematics*, 101(1–2):117–129, 1999.

[14] Craig Gotsman and Vitaly Surazhsky. Guaranteed intersection-free polygon morphing. *Computers and Graphics*, 25(1):67–75, 2001.

[15] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. *ACM Transaction on Graphics*, 24(3):561–566, 2005.

[16] Marc Alexa. Local control for mesh morphing. In *Proceedings of the International Conference on Shape Modeling and Applications*, pages 209–215, 2001.

[17] Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rossl, and Hans-Peter Seidel. Differential coordinates for interactive mesh editing. In *Proceedings of the 2004 ACM symposium on Solid and physical modeling*, pages 181–190, 2004.

[18] Olga Sorkine, Daniel Cohen-Or, and Yaron Lipman. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 179–188, 2004.

[19] Jianwei Hu, Ligang Liu, and Guozhao Wang. Dual laplacian morphing for triangular meshes. *Computer Animation and Virtual Worlds*, 18(4–5):271–277, 2007.

[20] Alla Sheffer and Vladislav Kraevoy. Pyramid coordinates for morphing and deformation. In *The 2nd International symposium of 3D Data Processing, Visualization and Transmission*, pages 68–75, 2004.

[21] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. Linear rotation-invariant coordinates for meshes. *ACM Transaction on Graphics*, 24(3):479–487, 2005.

[22] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and

Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Transaction on Graphics*, 23(3):644–651, 2004.

[23] Dong Xu, Hongxin Zhang, Qing Wang, and Hujun Bao. Poisson shape interpolation. In *Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 267–274, 2005.

[24] Ken Shoemake and Tom Duff. Matrix animation and polar decomposition. In *Proceedings of the conference on Graphics interface '92*, pages 258–264, 1992.

[25] Robert W. Sumner and Jovan Popovi'c. Deformation transfer for triangle meshes. *ACM Transaction on Graphics*, 23(3):399–405, 2004.

[26] Shi-Min Hu, Chen-Feng Li, and Hhui Zhang. Actual morphing: A physical-based approach for blending. In *Proceedings of the 2004 ACM symposium on Solid and physical modeling*, 2004.

[27] Han-Bing Yan, Shi-Min Hu, and Ralph Martin. Morphing based on strain field interpolation. *Journal of Visualization and Computer Animation*, 15(3–4):443–452, 2004.

[28] Han-Bing Yan, Shi-Min Hu, and Ralph Martin. 3d morphing using strain field interpolation. *Journal of Computer Science and Technology*, 22(1):147–155, 2007.

[29] Yunfan Bao, Xiaohu Guo, and Hong Qin. Physically based morphing of point-sampled surfaces. *Computers Animation and Virtual Worlds*, 16:509–518, 2005.

[30] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Eurographics Symposium on Geometry Processing*, pages 109–116, 2007.

[31] Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. A local/global approach to mesh parameterization. In *Eurographics Symposium on Geometry Processing*, pages 1495–1504, 2008.

[32] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. *ACM Transaction on Graphics*, 23(3):1134–1141, 2005.

[33] R.-W Sumner and J Popovic. Deformation transfer for triangle meshes. *ACM Transaction on Graphics*, 23(3):399–405, 2004.

[34] Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of SIGGRAPH 1985*, pages 245–254, 1985.

[35] Han-Bing Yan, Shi-Min Hu, Ralph Martin, and Yong-Liang Yang. Shape deformation using a skeleton to drive simplex transformations. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):693–706, 2008.

[36] Hongbo Fu, Chiew-Lan Tai, and Oscar Kin-Chung Au. Morphing with laplacian coordinates and spatial-temporal texture. In *Proceedings of Pacific Graphics 2005*, pages 100–102, 2005.

[37] William Baxter, Pascal Barla, and Ken-ichi Anjyo. Rigid shape interpolation using nor-

mal equations. In *Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*, pages 59–64, 2008.

Ya-Shu Liu obtained her Master degree from the Department of computer science and technology, Daqing Petroleum Institute, China in 2003. Now she is an assistant professor at Beijing University of Civil Engineering and Architecture in China. Her research interests include computer graphics, computer supported corporative work, network database.

Han-Bing Yan obtained his Ph.D. degree from the Department of computer science and technology, Tsinghua university, China in 2006. He is now working in the National Computer network Emergency Response technical team/Coordination Center of China. His research interests include computer graphics, computer animation, computer network security and information security.

Ralph R Martin received the PhD degree from Cambridge University in 1983. He is currently a professor at Cardiff University. He has published more than 200 papers and 10 books, covering such topics as solid and surface modeling, intelligent sketch input, geometric reasoning, reverse engineering, and various aspects of computer graphics. He is on the editorial boards of Computer Aided Design, Computer Aided Geometric Design, Graphical Models, and the International Journal of Shape Modelling.