Computers & Graphics 37 (2013) 364–375

Contents lists available at SciVerse ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Planar shape interpolation using relative velocity fields ☆

Guiqing Li *, Liang Yang, Shihao Wu, Wenshuang Tan, Xinyu Chen, Chuhua Xian

School of Computer Science and Engineering, South China University of Technology, Guangzhou University City, Guangzhou, China

ARTICLE INFO

Article history:
Received 2 September 2012
Received in revised form
2 March 2013
Accepted 2 March 2013
Available online 28 March 2013

Keywords:
2D shape interpolation
Shape transfer
Shape sequence editing
Relative velocity field
As isometric as possible

ABSTRACT

A novel approach is presented for interpolating two planar shapes using a shape sequence such that one shape is deformed into the other in an as-isometric-as-possible (AIAP) manner. The two shapes can have arbitrarily connected edge soup as long as they have the same connectivity. The interpolation is described by a nonlinear optimization problem that models the deformation energy, which penalizes the non-isometric component of the shape motion, based on the relation between two orthogonal vectors fields, namely, the edge vector field and the relative velocity field. Noticing that the nonlinear optimization includes two types of unknowns and admits a quadratic form with respect to each type, we address the optimization by iteratively solving two linear optimization procedures. One procedure admits a closed form, and the other procedure is associated with a quadratic energy that measures the deviation of the shape sequence from an AIAP motion. To speed up the processing, a local algorithm is devised to reduce the dimensionality of the linear optimization. This algorithm first addresses the AIAP interpolation of individual edges, and then reconstructs the shape sequence, frame by frame, using the generated edge vectors. Furthermore, an efficient initialization strategy is explored to greatly alleviate face-overlapping artifacts that are caused by using linear interpolation to make an initial guess for the shape sequence. Finally, relative velocity fields are employed to explore applications of the AIAP interpolation to shape transfer and sequence editing, in which the relative velocity fields of the given shape sequence are warped and copied onto the shape to be manipulated.

1. Introduction

Planar shape manipulation, such as interpolation, transfer, and sequence editing, plays an important role in 2D animation, computer games and advertisements. Concrete applications include image morphing [1], character animation [2], image warping, video retargeting [3] and editing [4]. Animating two-dimensional shapes is also applied to the simulation of large-scale crowds [5] and the visualization of medical images [6]. Moreover, creating 3D computer animations usually starts from sketching 2D animations, which helps establish a reasonable plan that can make generating the 3D animations more efficient.

Many shape deformation methods are free-form in the sense that they usually modify a shape that is embedded in some parametric forms, such as parametric volumes, barycentric coordinates, moving least squares (MLS), and bounded biharmonic functions. Free-form approaches are typically very efficient because of their linear blending property. However, most of them usually neither focus on searching optimal poses nor explicitly address the rationality of the resulting shapes. It is difficult for these methods to impose global geometric constraints on the deformed object. Some approaches directly force the deformation to fall in specific transformation spaces, but they are usually as-rigid-as-possible (ARAP) [7,8]. However, objects do not always behave in a rigid manner, while isometric motions, as shown in Fig. 1, are also common.

In this paper, we propose a framework that is based on as-isometric-as-possible (AIAP) constraints for the interpolation of 2D shapes, and we further explore the applications of this approach to shape transfer and sequence editing. Our method is an edge-based scheme and can address arbitrary edge 'soup' shapes (2d polygonal contours and 2D meshes are two special cases). The approach can be regarded as an improved 2D counterpart of geometric modeling in shape space [9], but with a different formulation and a set of new features. In summary, our contributions are as follows:

- A new formulation is proposed to model the AIAP interpolation of 2D shapes that are based on the orthogonal relation between the edge vector fields and their corresponding relative velocity fields. This formulation results in an optimization that can be addressed using the block coordinate descent method, in which each block is a linear minimization (Section 3).
- A global algorithm is sketched to tackle the optimization, and a local algorithm is further devised to accelerate the computation of AIAP shape sequences (Sections 4.1 and 4.2).
- An edge vector initialization is introduced. This procedure only needs to compute a sequence of edge vectors for each pair of

☆This article was recommened by J. Jorge

* Corresponding author. Tel.: +86 20 39380288 3511.
E-mail address: ligq@scut.edu.cn (G. Li).

0097-8493/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cag.2013.03.001

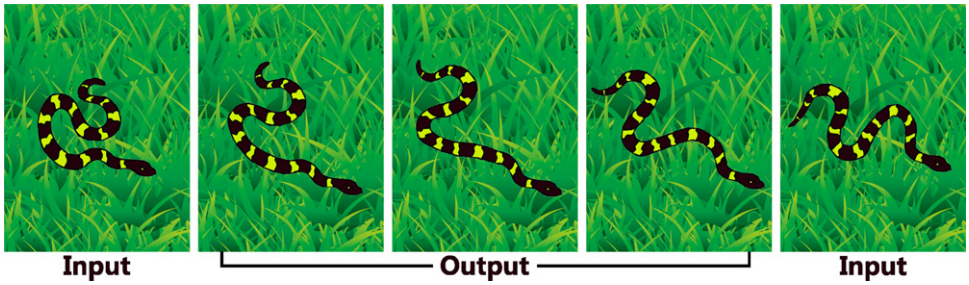

Fig. 1. Using linear interpolation as an initializer.

starting and ending edge vectors by interpolating the affine transformation between the two vectors. This approach naturally enables the propagation of compatible rotational angles among adjacent edges and, therefore, supports the interpolation of two shapes, with one shape being drastically deformed (Section 4.3).

- Relative velocity fields are employed to encode the motion style. Shape transfer and sequence editing are then converted to manipulate these fields (Section 5).

2. Related research

A substantial amount of work has focused on planar shape manipulation from a variety of disciplines. We briefly review the state-of-the-art in the field, according to the quantities that are used to reconstruct the final shapes.

Interpolation based on geometric statistical attributes. An early elaborate treatment for shape interpolation is an intrinsic solution by Sederberg et al. [10,11], which takes two topologically identical polygons as input and computes the edge length and the angle between adjacent edges for intermediate frames by linearly blending the corresponding quantity of the two given shapes. Although this approach produces pleasing intermediate results in most cases, it suffers from area distortion and edge intersection. Shapira and Rappoport [12] alleviated this phenomenon by decomposing the object into a set of star-skeleton representations. By replacing edge angles with face dihedral angles, Winkler et al. generalized the intrinsic solution to construct an interpolation of 3D models in which a hierarchical structure is employed to derive a globally coherent solution [13]. This approach is quite impressive and efficient; however, it is not clear how to migrate the 3D coherent mechanism to a 2D counterpoint.

Free-form and cage-based deformations. Free-form deformation embeds a shape to be modified into a parametric or subdivision volume and changes the shape by editing the control grid of the volume [14,15]. Cage-based methods encode a shape in a cage by using barycentric coordinates and modifying the shape by manipulating the cage [16–19]. More generally, some approaches encode a 2D or 3D space by using a set of geometric primitives, such as points and edges, and modeling a shape by warping the space via moving primitives [20–22]. Surazhsky and Gotsman [23] deployed two shapes to be interpolated into the same convex polygon and then gained the intermediate frames by interpolating the mean value coordinates of the two shapes.

Editing with differential geometric quantities. There is a large body of literature that is related to deformation methods that are based on differential quantities such as Laplacian editing [24–26], gradient-based deformation [27] and interpolation [28]. More details appear in [29]. These methods were initially developed for 3D shape manipulations. In 2D space, the involved differential operators are applicable only for polygonal contours. For example, Weng et al. [30] adapted Laplacian editing to deform 2D shapes by maintaining boundary Laplacian coordinates and forcing the interior area unchanged.

ARAP interpolation and deformation. The ARAP shape interpolation by Alexa et al. [7] reconstructs the shape sequence by blending the rotational and scaling components of the transformation matrix between a pair of starting and ending triangles, respectively. Because this interpolation method could exhibit serious artifacts for large-scale deformations that result from rotational inconsistency among adjacent triangles, Baxter et al. improved the approach by developing a propagation technique to produce compatible rotations [31]. However, both the methods are sensitive to the quality of triangulations because of matrix inversion when establishing the transformation between two triangles. The linear combination of transformation [32] and the steady affine motion [33] focus on exploiting an appropriate method to blend the affine transformation matrices to produce a pleasing trajectory between two shapes. An ARAP deformation approach proposed by Igarashi et al. achieved the ARAP effect by preserving the local frame that is related to each triangle of the shape. Karni et al. [34] established a more general framework to keep deformed objects from having distortion.

Motion-based methods. Geometric modeling in shape spaces [9] considered 3D shape interpolation to be a geodesic path computation in a Riemannian space. Although this approach is very elegant for modeling 3D shapes in an isometric manner, its framework is inefficient because of solving the geodesic path results in a large nonlinear optimization problem. Furthermore, shape sequence initialization with linear interpolation also prevents it from addressing large-scale deformations. Whitted et al. [35] represented a cartoon image as a graph, with each graph edge represented as a stroke. This approach directly specifies a trajectory for each node using logarithmic spirals and interpolates a pair of strokes using the approach in [11]. Ben-Chen et al. introduced a discrete Killing vector field on a 2D-manifold mesh for texture and geometry synthesis, for which the Killing vector field is actually a velocity field that generates isometric motion [36]. Solomon et al. [37] further applied Killing vector fields to 2D shape deformations (AKAP). Instead of specifying the new position of the vertices as handles for the deformed shape, their approach requires prescribing the velocity of these vertices. In addition, their approach minimizes the so-called Killing energy, which penalizes deviations in the speed field for being a Killing field. Huang et al. [38] proposed a physically based interpolation approach that describes 3D deformation as a linear elasticity motion and employs the modal analysis technique to solve the motion equation.

Most of the aforementioned approaches, except for those that introduce rigid constraints, do not explicitly support preserving global or local geometric quantities, such as the volume, area, and edge length, during deformation. Similar to the shape space approach [9], our framework uses a non-linear optimization to directly model the isometric constraints from the viewpoint of motion. Nevertheless, our formulation is more intuitive and easier to understand. In addition, a faster solver is provided for the optimization, and a more

robust initialization technique is presented. Compared to the vertex-based centered AKAP method, our approach also addresses the isometric motion problem but focuses on shape interpolation instead of deformation. Moreover, our formulation is edge-based and is built upon the relative speed between adjacent vertices and penalizes the deviation of the relative speeds from being an isometric motion. Experiments demonstrate that our method, as an approach that supports both the contour and mesh representations, is more robust in preventing area shrinkage than the intrinsic interpolation [11] and is more insensitive to triangulation quality than the ARAP interpolation [7,31].

3. 2D AIAP shape interpolation

Consider an abstract planar shape $M=(I,E)$, in which $I=\{0,1,\ldots,n\}$ is the set of vertex indices, and $E\subseteq I\times I$ is the edge set. Our shapes can be a 2D-connected edge soup with polygons or meshes as special cases, and face information is not a prerequisite for our method unless texture mapping is performed after interpolation. Without loss of generality, we assume that an instance of M is always deformed in a 2D space from time $t=0$ to $t=1$, with the vertex number and connectivity sustained during the process. Additionally, we do not take the translation into account in our formulation.

Let $p_i(t):[0,1]\rightarrow R^2$ be the trajectory of vertex i, and let $\mathbf{e}_{ij}(t)=p_i(t)-p_j(t)$ be the vector of edge (i,j). We further denote the set of vertices at time t by $\mathbf{P}(t)=\{p_i(t)\}_{i=0}^{n}$, which forms a trajectory of shape M, the set of edge vectors by $\mathbf{E}(t)=\{\mathbf{e}_{ij}(t):(i,j)\in E\}$ called *an edge vector field*, and the instance of M with $\mathbf{P}(t)$ by $M(t)$. Furthermore, we use a prime mark to indicate the derivative operation. For example, $p'_i(t)$ represents the velocity of point i, $\mathbf{P}'(t)$ denotes the set of vertex velocities (called *a deformation field* in [9]), and $\mathbf{E}'(t)$ represents $\{\mathbf{e}'_{ij}(t):(i,j)\in E\}$, which we call *a relative velocity field* on $M(t)$, where $\mathbf{e}'_{ij}(t)=p'_i(t)-p'_j(t)$ is the relative velocity of edge (i,j) at time t.

3.1. Isometric deformation energy

To guarantee that the deformation $\mathbf{P}(t)(t\in[0,1])$ is isometric, the length square $\|\mathbf{e}_{ij}(t)\|^2$ of each edge (i,j) should remain constant, namely, $\|\mathbf{e}_{ij}(t)\|^2=const$. Differentiating two sides of the equation with respect to t yields

$$(\mathbf{e}'_{ij}(t))^T\mathbf{e}_{ij}(t)=0,\quad \forall(i,j)\in E, \tag{1}$$

where the superscript T is a transpose operator. Eq. (1) indicates that a relative velocity field is orthogonal to its corresponding edge vector field, as shown in Fig. 2. Our key observation is that, for planar vectors $\mathbf{e}'_{ij}(t)$ and $\mathbf{e}_{ij}(t)$, Eq. (1) implies that the two involved vectors are orthogonal and that there exists a scalar $c_{ij}(t)$ such that

$$\mathbf{e}'_{ij}(t)=c_{ij}(t)\mathbf{R}\left(\frac{\pi}{2}\right)\mathbf{e}_{ij}(t), \tag{2}$$

where $\mathbf{R}(\theta)=\begin{pmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{pmatrix}$ is a rotational matrix. Obviously, the deformation field $\mathbf{P}'(t)(t\in[0,1])$ on $M(t)$ is isometric if and only if Eq. (2) holds for all the edges $(i,j)\in E$. For convenience of description, we collect all the above coefficients into a set: $C(t)=\{c_{ij}(t):(i,j)\in E\}$.

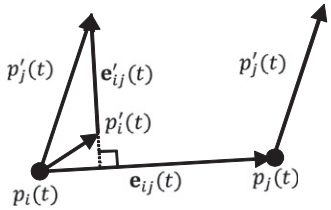

Fig. 2. The velocity difference $\mathbf{e}'_{ij}(t)=p'_j(t)-p'_i(t)$ between vertices i and j is orthogonal to their edge vector $\mathbf{e}_{ij}(t)=p_j(t)-p_i(t)$.

For arbitrary edge vector fields $\mathbf{E}(t)$, we can evaluate how much the corresponding motion approximates an isometric deformation by the following energy:

$$\mathcal{E}_{ISO}(t)=\sum_{(i,j)\in E|}\|\mathbf{e}'_{ij}(t)-c_{ij}(t)\mathbf{R}\left(\frac{\pi}{2}\right)\mathbf{e}_{ij}(t)\|^2\,dt, \tag{3}$$

where $c_{ij}(t)$ is the solution of the minimizer for edge (i,j)

$$\underset{c_{ij}(t)}{\mathbf{argmin}}\|\mathbf{e}'_{ij}(t)-c_{ij}(t)\mathbf{R}\left(\frac{\pi}{2}\right)\mathbf{e}_{ij}(t)\|^2. \tag{4}$$

In fact, energy $\mathcal{E}_{ISO}(t)$ plays the role of a Riemannian metric of the shape space, as described in [9]. This energy measures the residual of projecting the motion into the isometric shape space.

3.2. Optimization formulation

Given two shapes $M(0)=(\mathbf{P}(0),E)$ and $M(1)=(\mathbf{P}(1),E)$, their interpolation is to solve a trajectory $\mathbf{P}(t)$ that passes through $\mathbf{P}(0)$ and $\mathbf{P}(1)$. An interpolation is *as isometric as possible* (AIAP) provided that it minimizes the following isometric energy:

$$\int_0^1\mathcal{E}_{ISO}(t)\,dt. \tag{5}$$

It should be noted that, for an undetermined motion, $c_{ij}(t)$ is unknown. This relationship means that both $\mathbf{P}(t)$ and $C(t)$ must be solved in the variational problem that is defined by Eq. (5).

It is difficult to obtain an analytic solution for $\mathbf{P}(t)$. In practice, we approximate the solution by uniformly sampling the time interval [0,1] by $m+1$ slices such that the corresponding frames form a plausible motion sequence, where m is a number that is specified by users. Let $\mathcal{T}=\{t_k=k/m\}_{k=0}^{m}$ be the set of time slices. We obtain a discrete form for Eq. (5):

$$\underset{\{\mathbf{P}(t_k),C(t_k)\}_{k=1}^{m-1}}{\text{argmin}}\ \sum_{k=0}^{m-1}\mathcal{E}_{ISO}(t_k), \tag{6}$$

with the velocity of each vertex approximated by

$$p'_i(t_k)\approx\frac{p_i(t_{k+1})-p_i(t_k)}{t_{k+1}-t_k},\quad i=0,\ldots,n,\ \ k=0,\ldots,m, \tag{7}$$

where $\{\mathbf{P}(t_k)\}_{k=0}^{m}$ constitutes a discrete trajectory, which is also called a shape sequence or a sequence of shapes.

Accordingly, the minimization in Eq. (6) acts in the role of seeking a geodesic path in the shape space [9]. For conciseness of description, symmetry in discretizing the integral of Eq. (5) has been neglected in Eq. (6). In our implementation, a reverse sampling (from $t=1$ to $t=0$) of the integral with $p'_i(t_k)\approx(p_i(t_{k-1})-p_i(t_k))/(t_k-t_{k-1})$ is also considered in addition to Eq. (6) to guarantee that the results are symmetric with respect to two given shapes.

4. Numerical solutions

A global algorithm is first presented to solve Eq. (6) in this section. On the basis of the global framework, a local algorithm is then designed to improve the efficiency of solving the optimization. Finally, an initialization method is introduced to overcome the drawback of the initialization by linear interpolation.

4.1. Global optimization

Although it is nonlinear, Eq. (6) can be reduced to two quadratic energy optimization problems by using the block coordinate descent method [39]. If the trajectory of each vertex is given, then we can calculate the elements of $C(t_k)$ by solving the linear optimizations of Eq. (4), which admits the following closed form:

$$c_{ij}(t_k)=\frac{(\mathbf{e}_{ij}(t_k))^T\mathbf{e}'_{ij}(t_k)}{(\mathbf{e}_{ij}(t_k))^T\mathbf{R}\left(\frac{\pi}{2}\right)\mathbf{e}_{ij}(t_k)},\quad \forall(i,j)\in E,\ \ k=0,\ldots,m. \tag{8}$$

In contrast, once $\{C(t_k)\}_{k=0}^{m-1}$ are determined, Eq. (6) also degenerates to a linear optimization with respect to the vertices of all the intermediate frames $\{\mathbf{P}(t_k)\}_{k=1}^{m-1}$. Repeating the above two processes alternately leads to Algorithm 1, which is global because all the intermediate frames are solved in an optimization problem.

Algorithm 1. Global Shape Interpolation.

Input: $\mathbf{P}(t_0)$, $\mathbf{P}(t_m)$, m
Output: $\mathbf{P}(t_k)$, $k=1,\ldots,m-1$
for $k=1,\ldots,m-1$
  $\mathbf{P}(t_k)\leftarrow$Initialization($P(t_0),P(t_m),m,k$);
endfor
repeat
  $c_{ij}(t_k)\leftarrow$ Compute By Eq. (8);
    with $\{\mathbf{P}(t_k)\}_{k=0}^{m}$ known;
  $\{\mathbf{P}(t_k)\}_{k=1}^{m-1}\leftarrow$ Solve Eq. (6)
    with $c_{ij}(t_k)$ known;
until the convergent criterion holds

The convergent criterion will be detailed in Section 4.2.

4.2. Local optimization

The above global solver involves $n(m-1)$ vertices of all the intermediate frames in a minimizer, which includes $2n(m-1)$ variables. To further improve the efficiency, we decompose the problem spatially, namely, by performing the iterative procedure that is described in Algorithm 1 on individual starting and ending edge pairs. This process yields the following quadratic energy for the edge (i,j):

$$\mathcal{E}_{EDG}(t_k)=\sum_{k=0}^{m-1}\|\mathbf{e}'_{ij}(t_k)-c_{ij}(t_k)\mathbf{R}\left(\frac{\pi}{2}\right)\mathbf{e}_{ij}(t_k)\|^2, \tag{9}$$

where $\mathbf{e}'_{ij}(t_k)\approx(\mathbf{e}_{ij}(t_{k+1})-\mathbf{e}_{ij}(t_k))/(t_{k+1}-t_k)$.

Our local algorithm works as follows.

First, we can compute a sequence of initial edge vectors for each edge pair $\mathbf{e}_{ij}(t_0)$ and $\mathbf{e}_{ij}(t_m)$ using the initializers that are described in Section 4.3.

Second, we compute $\mathbf{P}(t_k)$ of each t_k by assembling the edge vectors in the same frame together through minimizing the energy

$$\mathcal{E}_{FRA}(t_k)=\sum_{(i,j)\in E}w_{ij}\|p_i(t_k)-p_j(t_k)-\mathbf{e}_{ij}(t_k)\|^2, \tag{10}$$

where $w_{ij}=1/\|\mathbf{e}_{ij}(t_k)\|$. Usually, short edges will lead to larger orientation changes compared with long edges under the same deviation. Therefore, the weights are chosen as the reciprocal of the edge length to balance the orientation change and produce a smoother result. Noting that the solution is unique up to the translation, we fix one vertex for all the frames. In practice, we simply allow the first vertex of the shape to remain unchanged. It is reasonable to do so because each intermediate shape is uniquely determined by all its edge vectors, which are independent of the selection of the fixed vertex.

Third, we employ Eq. (8) to compute $c_{ij}(t_k)$, using edge vectors of $\mathbf{P}(t_k)$ that were obtained in the assembling stage.

Fourth, we minimize Eq. (9) to calculate a more 'isometric' sequence of edge vectors. Repeating the steps from the second to the fourth step leads to Algorithm 2. Because we are minimizing the isometric energy edge by edge, this algorithm is essentially a greedy method.

Algorithm 2. Local Shape Interpolation.

Input: $\mathbf{P}(t_0),\mathbf{P}(t_m),m$
Output: $\mathbf{P}(t_k),k=1,\ldots,m-1$
for $(i,j)\in E$
  $\{\mathbf{e}_{ij}(t_k)\}_{k=1}^{m-1}\leftarrow$ Initialize edge vectors by Eq. (13);
endfor
repeat
  for $k=1,\ldots,m-1$
    $\mathbf{P}(t_k)\leftarrow$ Compute the kth frame from Eq. (10);
  endfor
  for $(i,j)\in E$
    $c_{ij}(t_k)\leftarrow$ Compute by Eq. (8);
    $\{\mathbf{e}_{ij}(t_k)\}_{k=1}^{m-1}\leftarrow$ Compute edge vectors by Eq. (9);
  endfor
until a convergent criterion holds

In our implementation, the maximal distance between the discrete trajectories computed in two subsequent iterations is used to terminate the 'repeat' loop in Algorithm 2. We denote the two subsequent trajectories separately by $\{\mathbf{P}^{old}(t_k)\}_{k=1}^{m-1}$ and $\{\mathbf{P}^{new}(t_k)\}_{k=1}^{m-1}$. Then, the convergent criterion possesses the following form:

$$\max\{\|p_i^{new}(t_k)-p_i^{old}(t_k)\|:i=0,\ldots,n;k=1,\ldots,m-1\}<\tau, \tag{11}$$

where τ is a relative error that is specified by the users. We set $\tau=5\times10^{-4}$ of the diagonal length of the shape bounding box in all the examples.

4.3. Shape sequence initialization

An initial guess on the discrete trajectory is inevitable for estimating $C(t)$, regardless of whether the global or local algorithm is employed. Moreover, a good guess is crucial for both obtaining globally optimal solutions and speeding up the convergence. Linear shape interpolation can provide a rough guess. However, this approach tends to produce initial shapes with overlapped faces and, therefore, yields unreasonable results. An ideal initialization should be able to support a large-scale deformation but be simple enough in the sense that it has linear time complexity.

We present an edge vector interpolation for producing initial shape sequences. In fact, according to Eq. (8), only a guess for the edge vectors instead of edge endpoints of $\mathbf{P}(t_k)$ is sufficient for evaluating $C(t_k)$. This approach causes us to consider edges individually again. Specifically, we find a unique similar transformation

$$\mathbf{S}_{ij}=s_{ij}\mathbf{R}(\theta_{ij}) \tag{12}$$

such that $\mathbf{e}_{ij}(t_m)=\mathbf{S}_{ij}\mathbf{e}_{ij}(t_0)$, where s_{ij} is a scaling factor, and $\mathbf{R}(\theta_{ij})$ a rotational matrix that rotates by angle θ_{ij}. It is easy to determine that

$$s_{ij}=\frac{\|\mathbf{e}_{ij}(t_m)\|}{\|\mathbf{e}_{ij}(t_0)\|},\quad \cos\theta_{ij}=\frac{(\mathbf{e}_{ij}(t_0))^T\mathbf{e}_{ij}(t_m)}{\|\mathbf{e}_{ij}(t_0)\|\|\mathbf{e}_{ij}(t_m)\|},$$

$$\sin\theta_{ij}=\frac{(\mathbf{e}_{ij}(t_0))^T\mathbf{R}\left(\frac{-\pi}{2}\right)\mathbf{e}_{ij}(t_m)}{\|\mathbf{e}_{ij}(t_0)\|\|\mathbf{e}_{ij}(t_m)\|},$$

where $\theta\in[-\pi,\pi]$.

To remedy the incompatibility among the rotational angles of adjacent edges, we adapt the breadth-first propagation algorithm

[31] to our setting, to readjust the rotational angles of the edges. Briefly, we construct a dual graph from M as follows: each edge of M corresponds to a node of the graph, and two nodes of the graph are connected if and only if their corresponding edges in M share a vertex. A breadth-first traversal is then conducted on the graph to check the compatibility between the rotational angles of the parent and child nodes of the graph. If the absolute difference between two angles is greater than π, then the rotational angle of the child node is modified by adding or subtracting a multiple of 2π.

Finally, we can obtain the initial sequence for each edge $(i,j)\in E$ by linearly interpolating the scaling factor s_{ij} and the rotational angle θ_{ij}:

$$e_{ij}(t_k) = (1-t_k + t_k s_{ij}) R(t_k\theta_{ij}) e_{ij}(0), \quad k = 0, 1, \ldots, m. \tag{13}$$

It should be noted that there is some similarity between our initialization and the first stage of the ARAP interpolation [7]. Both the methods produce intermediate frames by interpolating transformation matrices. However, our transformation is defined by two edge vectors, while the ARAP interpolation applies to a pair of triangles. As a result, the ARAP interpolation method is slightly complicated because of the SVD that is involved when decomposing the transformations into rotational and scaling components. Although both the methods admit a closed form, our formulation is more concise.

Fig. 3 shows the results from using our method with different initializers. Fig. 3(a) is related to linear initialization, which exhibits serious distortion. Fig. 3(b) is related to the edge vector interpolation strategy and depicts a smooth transition from the starting shape to the ending shape.

5. Applications

Based on the formulation that was established in Section 3, modeling AIAP shape transfer and sequence editing becomes an easy task. Similar to in Section 4, we also have both global and local strategies to establish our formulation. For succinctness, we directly describe the local approaches here.

5.1. Planar shape transfer

The metaphor of shape transfer is deforming a target reference shape $\widetilde{M}(t_0)$ to create its new pose $\widetilde{M}(t_m)$ by imitating the deformation between given source shapes $M(t_0)$ and $M(t_m)$, which are also called the source reference shapes. Again, we assume that all the involved shapes have the same connectivity and that their vertex correspondence has been established. In addition, $M(t_0)$ is deformed to $M(t_m)$ in an AIAP manner.

The key is to extract quantities that encode the deformation process, which are independent of the geometry of source shapes. Therefore, we first must compute the AIAP shape sequence, interpolating $M(t_0)$ and $M(t_m)$. Then, we must calculate the relative velocity fields $\mathbf{E}'(t_k), k = 0, \ldots, m-1$ from the sequence and, finally, transfer the relative velocity of $\mathbf{e}_{ij}$ onto $\widetilde{\mathbf{e}}_{ij}$, edge by edge.

In fact, we hope that the motion from $\widetilde{M}(t_0)$ to $\widetilde{M}(t_m)$ also obeys the AIAP constraint. Therefore, we need to warp $\mathbf{E}'(t_k)$ before transferring to guarantee that the new relative velocity fields can reflect the edge orientation difference between $\mathbf{E}(t_0)$ and $\widetilde{\mathbf{E}}(t_0)$. To do so, we can guarantee that Eq. (2) still holds for the motion of the target object and that the magnitude of the relative velocity is proportional to the corresponding edge length.

Specifically, our transfer method can be summarized as follows.

First, we perform Algorithm 2 on $\mathbf{P}(t_0)$ and $\mathbf{P}(t_m)$ for evaluating $\mathbf{E}'(t_k)$.

Second, we compute a similar transformation matrix $\widetilde{\mathbf{S}}_{ij} = \widetilde{s}_{ij} R(\widetilde{\theta}_{ij})$ for each edge pair between $M(t_0)$ and $\widetilde{M}(t_0)$, such that $\widetilde{\mathbf{S}}_{ij}\mathbf{e}_{ij}(t_0) = \widetilde{\mathbf{e}}_{ij}(t_0)$ (see Eq. (12) for the computation of $\widetilde{\mathbf{S}}_{ij}$). Next, the relative velocity of edge (i,j) in $\widetilde{M}(t_k)$ can be estimated by

$$\widetilde{\mathbf{e}}'_{ij}(t_k) = \widetilde{\mathbf{S}}_{ij}\mathbf{e}'_{ij}(t_k), \quad k = 0, \ldots, m-1. \tag{14}$$

This step completes the computation of $\widetilde{\mathbf{E}}'(t_k)$.

Third, we determine $\{\widetilde{\mathbf{e}}_{ij}(t_k)\}_{k=0}^{m-1}$ by minimizing the transfer energy for each individual edge

$$\widetilde{E}_{TRA}(i,j) = \sum_{k=0}^{m-1} \|\widetilde{\mathbf{e}}_{ij}(t_{k+1}) - \widetilde{\mathbf{e}}_{ij}(t_k) - \widetilde{\mathbf{S}}_{ij}(t_k)\mathbf{e}'_{ij}(t_k)\|^2 \tag{15}$$

to ensure that $\widetilde{\mathbf{e}}_{ij}(t_0)$ can smoothly move to $\widetilde{\mathbf{e}}_{ij}(t_m)$ in an AIAP manner. Note that, if $\{\widetilde{\mathbf{e}}_{ij}(t_k)\}_{k=0}^{m-1}$ is a solution that minimizes Eq. (15), then $\{\widetilde{\mathbf{e}}_{ij}(t_k) + \mathbf{c}\}_{k=0}^{m-1}$, where $\mathbf{c}$ is a constant vector, will also be a solution. This result indicates that the optimization problem defined by Eq. (15) is undetermined. We solve this problem by assigning a velocity to a vertex of the target reference shape $\widetilde{M}(t_0)$ (set the first vertex to 0 velocity in our implementation). It is easy to observe that this treatment does not influence the final result, up to the translation.

In the final step, we only need to minimize an assembling energy that is similar to Eq. (10) to reconstruct $\widetilde{P}(t_k)$ frame by frame. Fig. 4 shows that the approach can even correctly transfer the details on the bar to the U-shape.

5.2. Shape sequence editing

Sequence editing addresses the shape changing of objects in a clip of a video by altering one or several frames. Suppose that a sequence of source shapes $\{M(t_k)\}_{k=0}^{m}$ has been given, and its $(\kappa+1)$th frame $M(t_\kappa)$ has been deformed to a target shape $\widetilde{M}(t_\kappa)$. Our goal is to generate a target sequence $\{\widetilde{M}(t_k)\}_{k=0}^{m}$ that not only inherits the motion style of the source sequence but also preserves the change to its $(\kappa+1)$th frame. This process can be viewed as an

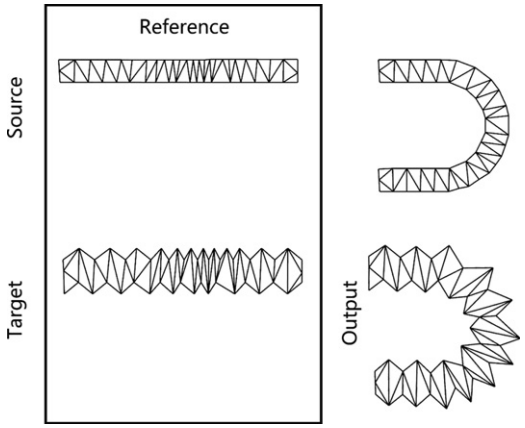

Fig. 4. Shape transfer: two source shapes as well as the target reference are given, and the other target shape (output) is the transfer result by our method.

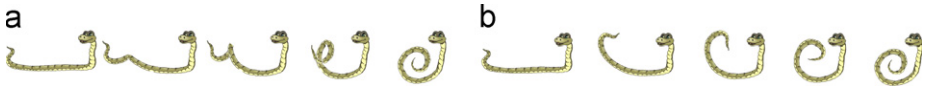

Fig. 3. Initialization comparison: (a) results with linear interpolation as an initializer. The tail of the snake is folded in this example; (b) results with the edge vector interpolation as an initializer. Both the leftmost and rightmost shapes are inputs.

operation of spatial and time signal processing [40] or can be cast into a minimization of a force residual objective [41].

In our scenario, we regard the sequence editing to be a generalized shape transfer problem. The source sequence here corresponds to the source motion that is produced by interpolating the two source poses in the shape transfer, and the target shape $M(t_\kappa)$, accordingly, plays the role of the target reference object in the shape transfer.

The procedure in Section 5.1 can be applied with a slight modification. First, because a sequence of source shapes is given in our setting, the relative velocity fields, $\{\mathbf{E}'(t_k)\}_{k=0}^{m-1}$, of the source shapes can be computed directly. Second, $\widetilde{\mathbf{S}}_{ij}$ is calculated such that $\widetilde{\mathbf{S}}_{ij}\mathbf{e}_{ij}(t_\kappa)=\widetilde{\mathbf{e}}_{ij}(t_\kappa)$. The remaining steps of the approach are completely in accordance with the steps in Section 5.1, namely, applying Eq. (14) to estimate $\{\widetilde{\mathbf{E}}'(t_k)\}_{k=0}^{m-1}$, minimizing Eq. (15) to reconstruct $\{\widetilde{\mathbf{e}}_{ij}(t_k)\}_{k=0}^{m-1}$ for all the edges, and assembling $\widetilde{P}(t_k)$ by using a formulation that is similar to Eq. (10).

6. Results and discussion

The proposed technique has been applied to a variety of drawings and images. This section first presents two examples to illustrate the adaptability of the proposed approach. It then shows a number of comparisons with existing approaches, including the shape space [9] and intrinsic [11] and improved ARAP [31] interpolations, to demonstrate the effectiveness of our approach. Then, we present two examples to show applications for transfer and sequence editing. Next, we show the superiority of our method with respect to the efficiency of the shape space interpolation by timing the examples that are used in this paper. To conclude this section, the limitations of our approach are discussed. More results can be found in the supplemental materials. Without specification, our results are always created by the local approach (Algorithm 2), and our tests are conducted on a PC with an Intel (R) core (TM) i5 with a 2.8 GHZ CPU and 4 GB of memory.

Adaptability. For correct initialization, our approach usually produces good results even though the quality of the initial sequence is unfavorable. Fig. 5 depicts the interpolation between two poses of a goose, in which the initial sequence (Top row) exhibits serious shrinkage, and the result (Bottom row) recovers those shrinking artifacts very well. On the other hand, the complexity of the given meshes appears to have less influence on the quality of the generated sequence, as shown in Fig. 6, in which the first and second rows are, respectively, wireframe and textured results for a pair of coarse meshes, while the third and fourth rows are for a pair of dense meshes. Because deformation usually preserves the shape for dense meshes better than for coarse meshes, our method can correspondingly produce shape interpolation with less distortion for dense meshes. This result can be observed clearly by comparing the textured images of the second and the fourth rows in Fig. 6.

Comparisons. Fig. 7 shows that the shape space interpolation suffers from obvious shrinkage and face flipping (see the zoom-in square regions in Fig. 7(a)), which results from incorrect initialization. In comparison, our approach produces a pleasing shape sequence (see Fig. 7(b)).

As noted by Shapira and Rappoport, the intrinsic method [11] suffers self-intersection and area shrinkage in some cases because it addresses edges and angles in a completely local way [12]. Fig. 8 depicts an example of interpolating 'S'-like and '2'-like shapes that are given as polygonal contours. The intrinsic approach yields intermediate frames with serious shrinkage in this example, while our result looks more reasonable. A new feature of our method is that it accepts a set of edges as input (see Fig. 9), while previous methods either define shapes using polygonal contours or represent models as triangular meshes.

The improved ARAP interpolation by Baxter et al. [31] can successfully address most of the testing examples. However, their approach is sensitive to mesh triangulations in the sense that it usually yields distorted intermediate frames for starting/ending triangulations that contain bad shape triangles, which could incur ill-conditioned transformation matrices. We can observe visible distortion on the head of the yoga actor in Fig. 10(a) by the improved ARAP method, while our interpolating frames look more realistic. Moreover, if flipped triangles exist in the given triangulations, their approach will fail completely (e.g., see Fig. 11). We believe that this result is an attractive feature of our method because generating compatible triangulations of high quality for two shapes is still a difficult task [42].

Shape transfer. In Fig. 12, three source shapes of the doll in the top line are given. Mapping their (warped) relative velocity fields to the Pooh bear (target) shape yields two new shapes, with one shape being squeezed and elongated and the other shape becoming bulged. Note that the head of the bear is slanted to the left, and the right arm of the Pooh bear has been raised. The generated sequence not only can capture the deformation style of the source shape sequence but also can correctly reflect the change of the target shape. This example demonstrates that the proposed transfer mechanism can produce convincing results.

Shape sequence editing. Given the shape sequence as shown in Fig. 13(a), we can generate Fig. 13(b) by dilating the hair of the boxer and enlarging the boxing glove in its middle frame, and we can generate Fig. 13(c) by altering the pose (raising the hand) of the boxer in the same frame.

Timings. Compared to the shape space method [9] by Kilian et al., our acceleration is based on three aspects: linear decomposition of the nonlinear optimization, local approximation of the global algorithm, and a better initializer; the initializer could drastically reduce the number of the iterations.

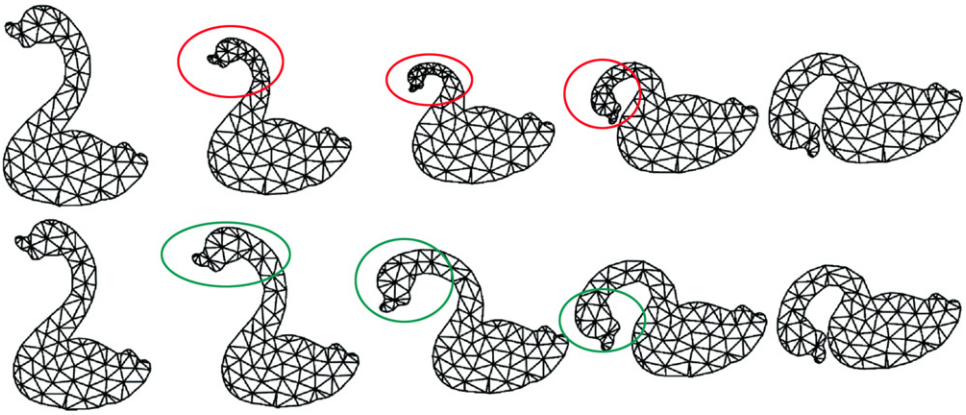

Fig. 5. The proposed approach can yield reasonable interpolated sequences for an initialization that is not very good.

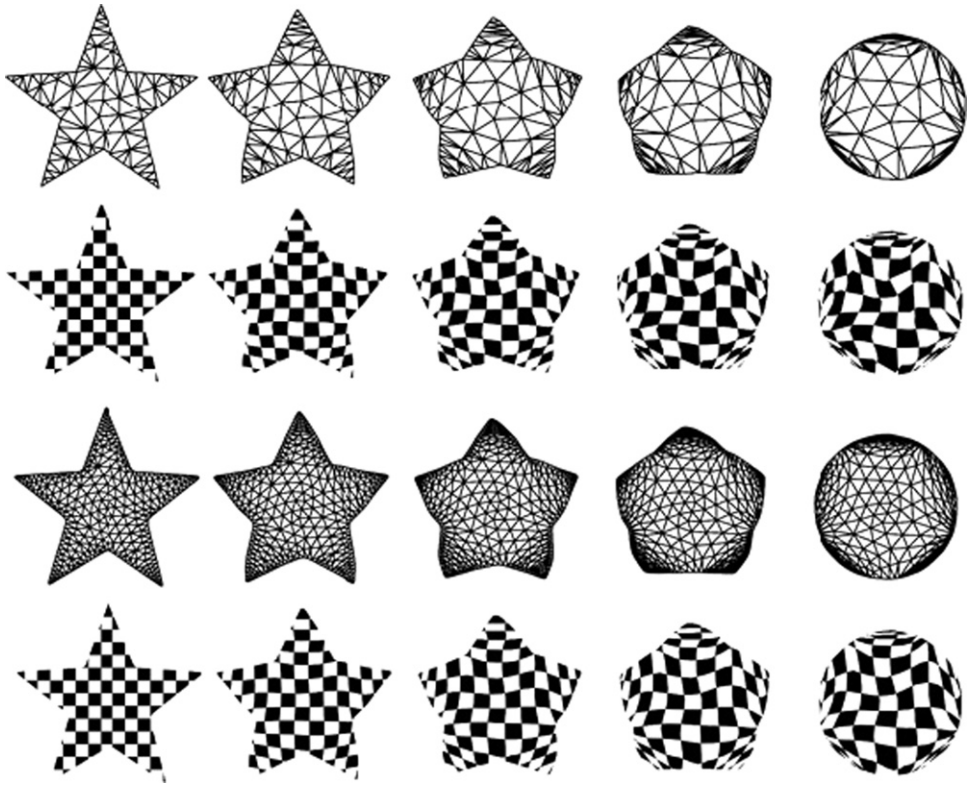

Fig. 6. Our approach can yield almost the same result for both coarse meshes (the first and second rows) and dense meshes (the third and fourth rows). However, the interpolating shapes are usually less distorted because it is easier to yield compatible starting and ending meshes with less distortion for dense meshes. (See the textured images in the second and fourth rows.)

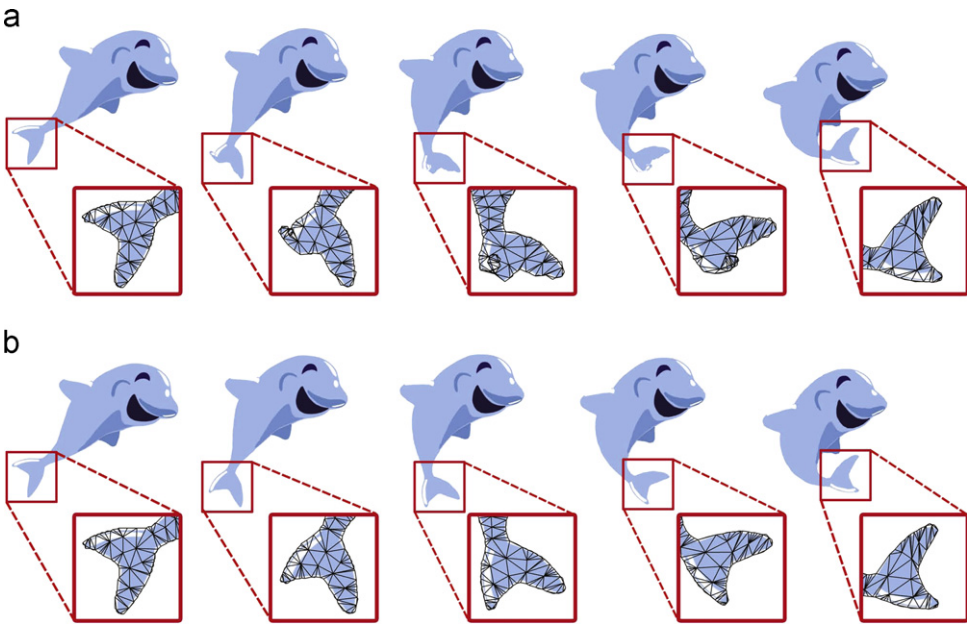

Fig. 7. Interpolation of dolphin images: (a) shape sequence by shape space interpolation; (b) our result.

Shape interpolation in [9] is cast into a non-linear least squares problem with $\Theta(mn)$ unknowns, which we solved using an iterative algorithm, the Gauss–Newton method, in which $m+1$ and $n+1$ are, respectively, the frame number and the vertex number of each frame. Each iteration involves three components: establishing a sparse Jacobian matrix J of size $m|E| \times mn$, where $|E| = \Theta(n)$ is the edge number of each frame; multiplying with its transpose to produce J^{TJ}; and solving the associated sparse linear system of size $\Theta(mn)$. Because the time complexity is $O(N \log N)$ for addressing a sparse linear system with N variables, we know that the third component dominates the complexity of the iteration and costs $O(\zeta_K mn \log(mn))$, where ζ_S denotes the iteration number of the method.

Moreover, our iteration body (see Algorithm 2) also consists of three parts. One part is a simple function evaluation, and the two other parts require solving sparse linear systems. First, evaluating

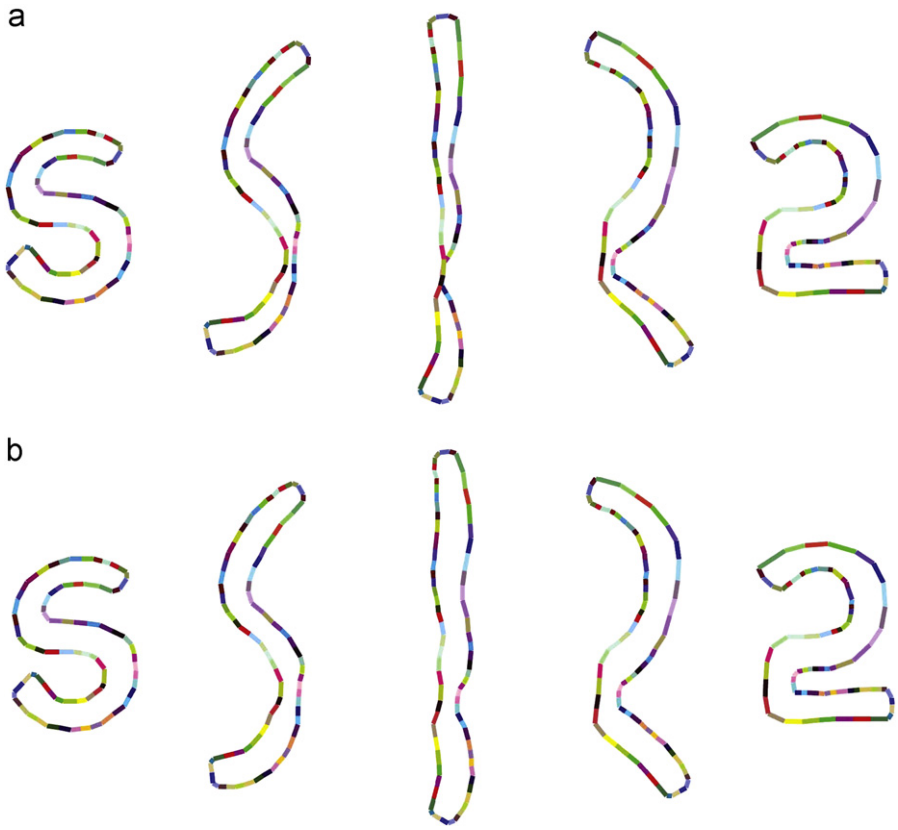

Fig. 8. Interpolation of 'S' and '2' shapes: (a) the intrinsic solution by Sederberg et al. exhibits area shrinkage, while (b) our result does not suffer this problem.

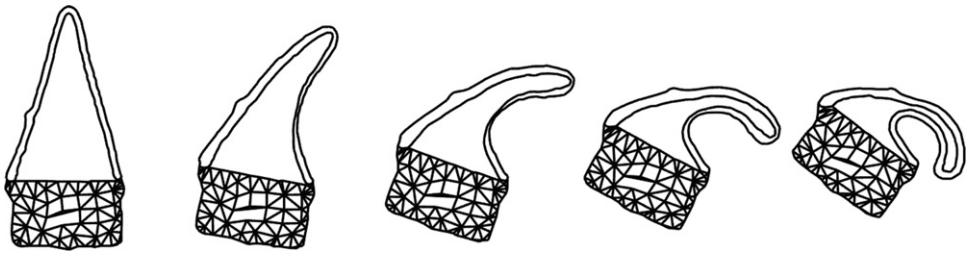

Fig. 9. Our approach can address shapes that are represented as edge soup.

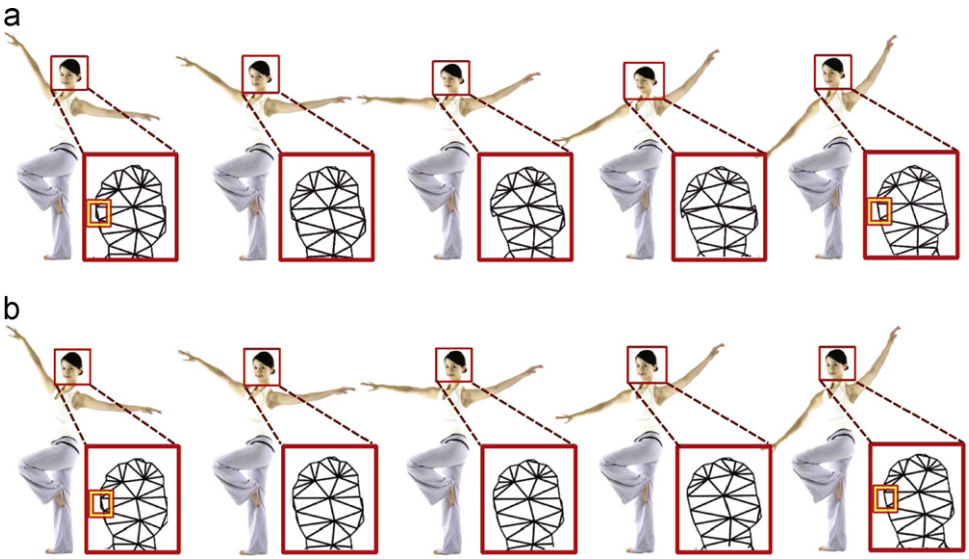

Fig. 10. Yoga act: (a) the improved ARAP method produces distorted intermediate frames (see the texture and meshes in the red squares of the top line) because there are some bad, degenerate triangles (see the small orange square regions in the larger red squares); (b) no similar artifact occurs in our result. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

Fig. 11. Sea horse: (a) because there are flipped triangles in the meshes of this example, the improved ARAP method fails to produce correct intermediate frames (see the tail of the three middle images), while (b) no similar artifact occurs in our result.

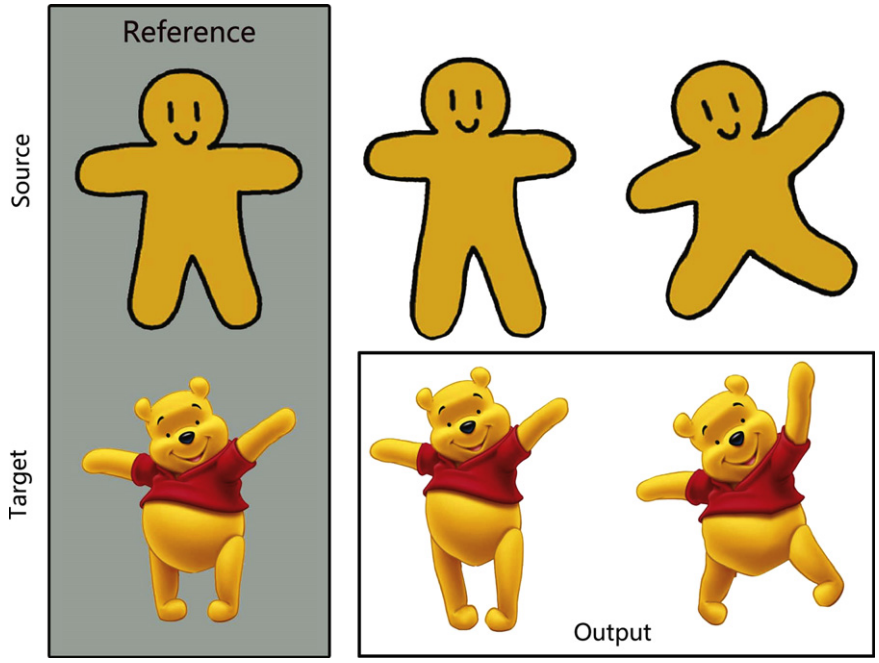

Fig. 12. The doll shape sequence (top line) and the first bear shape (bottom line) are given. The remaining two bear shapes are generated by our method.

Eq. (8) for all the edges requires $O(|E|)$ complexity. Next, solving $|E|$ linear optimization problems of size m separately (defined by Eq. (9)) expends $O(|E|m\log m)$ time. Third, minimizing $m-1$ quadratic energies of size $n-1$ (see Eq. (10)) consumes $O(mn\log n)$. In conclusion, the total complexity of our method is $\zeta_O[O(|E|)+O(|E|m\log m)+O(mn\log n)]=O(\zeta_O mn\log(mn))$, where ζ_O indicates the number of iterations.

The analysis shows that our method has the same asymptotic time complexity as the shape space method. However, the superiority of our approach in terms of the computational cost is supported by Table 1, which indicates that our method speeds up by scaling factors of ten to two hundred. Table 1 also shows that our approach gains increasing acceleration with the increase in the mesh complexity and the number of frames that are interpolated. Here, both the algorithms are terminated based on the criterion that is defined in Eq. (11), such that the generated results have the same interpolating quality.

Numerical analysis of the convergence. All our tested examples demonstrate that the proposed method is convergent in the sense that the convergent criterion defined by Eq. (11) is always satisfied. Here, we further analyze the tendency of the isometric deformation energy, which is defined by Eq. (6), in terms of the

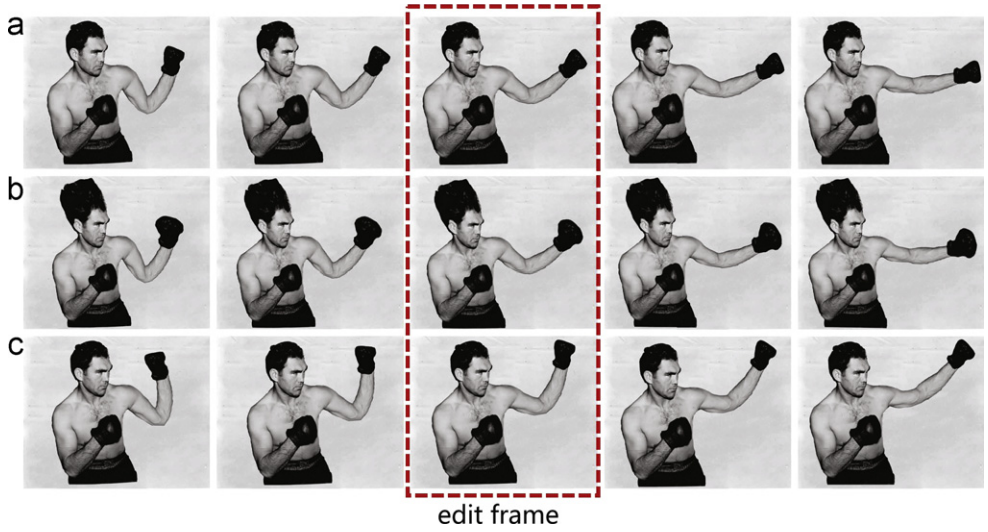

Fig. 13. Boxer sequence editing: (a) source sequence; (b) result after editing the hair and left glove regions; (c) result after raising the left hand.

Table 1
Efficiency comparison between the method of Kilian et al. and our method.

Examples	Vertex num.	Edge num.	Frame num.	Iteration num.		Time (s)	
				Ours	Kilian	Ours	Kilian
Fig. 1	99	211	9	5	17	0.205	1.529
			45	3	35	0.421	18.783
			205	2	50	0.842	137.92
Fig. 3	108	225	9	2	7	0.055	0.643
			45	2	6	0.234	3.299
			205	2	13	0.827	42.407
Fig. 7	467	1072	9	2	8	0.234	3.931
			45	2	16	0.998	168.18
			205	2	34	4.118	1592.4
Fig. 10	272	597	9	2	6	0.25	1.592
			45	2	6	0.562	15.21
			205	2	7	2.324	294.4
Fig. 8	92	198	9	2	13	0.103	1.123
			45	2	29	0.218	16.344
			205	2	50	0.759	150.5
Fig. 15	20	40	9	3	8	0.046	0.327
			45	2	15	0.104	1.607
			205	2	13	0.257	6.155

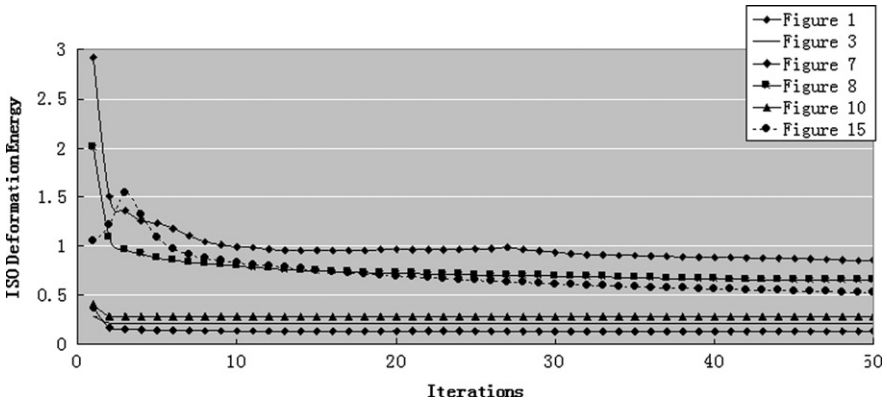

Fig. 14. The isometric deformation energy is computed for several examples used in our experiments. The figure shows that the energy decreases when the number of iterations increases.

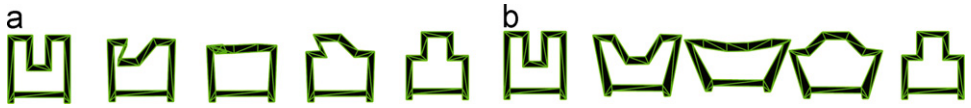

Fig. 15. Initialization comparison: results of our method using (a) edge vector interpolation and (b) linear interpolation as the initializer.

number of iterations. The energy decreases and converges to a minimum for the given examples, as shown in Fig. 14. This trend supports the observation that the proposed algorithm usually finds the optimum (or a suboptimal) solution.

Limitations. Although the edge vector interpolation as an initialization strategy has greatly improved the behavior of our AIAP interpolation compared to linear interpolation, it could yield self-intersection shapes for some special cases, as shown in Fig. 15, because the rotational direction of some boundary edges causes some ambiguity.

7. Conclusions and future work

We have proposed a new edge-based method for interpolating two planar shapes, as well as applications of this method from the viewpoint of motion. Our framework, which models interpolation as a nonlinear optimization, can be viewed as a 2D version of shape space interpolation. However, several new features are introduced. The approach has greatly increased the speed of the AIAP interpolation process. Moreover, it facilitates the correct interpolation of two shapes with large-scale deformations by extending the compatible angle propagation method of Baxter et al. to our setting and embedding it into our initialization stage. Finally, the interpolation technique is applied to formulate shape transfer and shape sequence editing. All these formulations involve the notion of a relative velocity field, which appears to be promising in shape modeling. The implementation of our approach is very simple and straightforward, and a variety of experimental results show that this approach is also efficient and robust.

Future work. We believe that our framework opens a space for investigating 2D shapes from the viewpoint of motion. Future work includes further accelerating local optimization using a GPU and exploiting the theoretical aspects of using relative velocity fields to perform shape deformation. Specifically, it is attractive to interpolate two shapes without creating compatible meshes. It is interesting to apply the idea of using 3D mesh/volume interpolation to improve the robustness and efficiency of shape space interpolation. The key challenge will be to find a good initial guess for the edge vector sequence when interpolating two edge vectors.

Acknowledgment

We want to thank the anonymous reviewers for their valuable suggestions. This work is supported by NSFC (60973084) and NSF of Guangdong, China (915106410-1000106).

Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.cag.2013.03.001.

References

[1] Meng M, Liu L. Sketching image morphing using moving least squares. In: The 3rd Korea-China joint conference on geometric and visual computing, Seoul; 2007. p. 190–7.
[2] Sykora D, Dingliana J, Collins S. As-rigid-as-possible image registration for hand-drawn cartoon animations. In: NPAR 2009; 2009. p. 25–33.
[3] Wang Y-S, Lin H-C, Sorkine O, Lee T-Y. Motion-based video retargeting with optimized crop-and-warp. ACM Trans Graph 2010;29(4):1–9.
[4] Zhou S, Fu H, Liu L, Cohen-Or D, Han X. Parametric reshaping of human bodies in images. ACM Trans Graph 2010;29(4):1–10.
[5] Kavan L, Dobbyn S, Collins S, Zra COJ. Polypostors: 2d polygonal impostors for 3d crowds. In: SI3D; 2008. p. 149–55.
[6] Chen W, Liang X, Maciejewski R, Ebert DS. Shape context preserving deformation of 2d anatomical illustrations. Comput Graph Forum 2009; 28(1):114–26.
[7] Alexa M, Cohen-Or D, Levin D. As-rigid-as-possible shape interpolation. In: SIGGRAPH 2000; 2000. p. 157–64.
[8] Igarashi T, Moscovich T, Hughes JF. As-rigid-as-possible shape manipulation. ACM Trans Graph 2005;24(3):1134–41.
[9] Kilian M, Mitra NJ, Pottmann H. Geometric modeling in shape space. ACM Trans Graph 2007;26(3):64–71.
[10] Sederberg TW, Greenwood E. A physically based approach to 2d shape blending. Comput Graph 1992;26:25–34.
[11] Sederberg TW, Gao P, Wang G, Mu H. 2-d shape blending: an intrinsic solution to the vertex-path problem. Comput Graph 1993;27:15–8.
[12] Shapira M, Rappoport A. Shape blending using the star-skeleton representation. IEEE Comput Graph Appl 1995;15(2):44–50.
[13] Winkler T, Drieseberg J, Alexa M, Hormann K. Multi-scale geometry interpolation. Comput Graph Forum 2010;29(2):309–18.
[14] Sederberg TW, Parry SR. Free-form deformation of solid geometric models. In: SIGGRAPH' 86. ACM Press; 1986. p. 151–60.
[15] Coquillart S. Extended free-form deformation: a sculpturing tool for 3d geometric modeling. Comput Graph 1990;24(4):187–96.
[16] Ju T, Schaefer S, Warren J. Mean value coordinates for closed triangle meshes. ACM Trans Graph 2005;26(3):561–6.
[17] Joshi P, Meyer M, DeRose T. Harmonic coordinates for character articulation. ACM Trans Graph 2007;26(3):71–80.
[18] Lipman Y, Levin D, Cohen-Or D. Green coordinates. ACM Trans Graph 2008;27 (3):1–10.
[19] Weber O, Gotsman C. Controllable conformal maps for shape deformation and interpolation. ACM Trans Graph 2010;29(4):1–11.
[20] Schaefer S, McPhail T, Warren J. Image deformation using moving least squares. ACM Trans Graph 2006;25(3):533–40.
[21] Gain J, Bechmann D. A survey of spatial deformation from a user-centered perspective. ACM Trans Graph 2008;27(4):1–21.
[22] Jacobson A, Baran I, Popovic J, Sorkine O. Bounded biharmonic weights for real-time deformation. ACM Trans Graph 2011;30(4):1–10.
[23] Surazhsky V, Gotsman C. Intrinsic morphing of compatible triangulations. Int J Shape Modeling 2003;16(2):195–212.
[24] Lipman Y, Sorkine O, Cohen-Or D, Levin D, Rossl C, Seidel H-P. Differential coordinates for interactive mesh editing. In: International conference on shape modeling and applications; 2004. p. 181–90.
[25] Sorkine O, Cohen-Or D, Lipman Y, Alexa M, Rossl C, Seidel H-P. Laplacian surface editing. In: Symposium on geometry processing 2004; 2004. p. 179–88.
[26] Lipman Y, Sorkine O, Levin D, Cohen-Or D. Linear rotation-invariant coordinates for meshes. ACM Trans Graph 2005;24(3):479–87.
[27] Yu Y, Zhou K, Xu D, Shi X, Bao H, Guo B, et al. Mesh editing with Poisson-based gradient field manipulation. ACM Trans Graph 2004;23(3):644–51.
[28] Xu D, Zhang H, Wang Q, Bao H. Poisson shape interpolation. Graphical Models 2006;68(3):268–81.
[29] Botsch M, Sorkine O. On linear variational surface deformation methods. IEEE Trans Vis Comput Graph 2008;14(1):213–30.
[30] Weng Y, Xu W, Wu Y, Zhou K, Guo B. 2d shape deformation using nonlinear least squares optimization. Vis Comput 2006;22(9–11):653–60.
[31] Baxter WV, Barla P, Anjyo Ki. Rigid shape interpolation using normal equations. In: Proceedings of NPAR 2008; 2008. p. 59–64.
[32] Alexa M. Linear combination of transformations. ACM Trans Graph 2002;21 (3):380–7.
[33] Rossignac J, Vinacua A. Steady affine motions and morphs. ACM Trans Graph 2011;30(5):1–16.
[34] Karni Z, Freedman D, Gotsman C. Energy-based image deformation. Comput Graph Forum 2009;28(5):1257–68.
[35] Whitted B, Noris G, Simmons M, Sumner R, Gross M, Rossignac J. Betweenit: an interactive tool for tight inbetweening. Comput Graph Forum 2010;29 (2):605–13.
[36] Ben-Chen M, Butscher A, Solomon J, Guibas L. On discrete Killing vector fields and patterns on surfaces. Comput Graph Forum 2010;29(5): 1701–1711.
[37] Solomon J, Ben-Chen M, Butscher A, Guibas L. As-Killing-as-possible vector fields for planar deformation. Comput Graph Forum 2011;30(5): 1543–1552.

[38] Huang J, Tong Y, Zhou K, Bao H, Desbrun M. Interactive shape interpolation through controllable dynamic deformation. IEEE Trans Vis Comput Graph 2011;17(7):983–92.
[39] Tseng P. Convergence of a block coordinate descent method for nondifferentiable minimization. J Optim Theory Appl 2001;109(3):475–94.
[40] Kircher S, Garland M. Free-form motion processing. ACM Trans Graph 2008;27(2):1–10.
[41] Barbic J, Sin F, Grinspun E. Interactive editing of deformable simulations. ACM Trans Graph 2012;31(4) Article 70.
[42] Baxter WV, Barla P, Anjyo Ki. Compatible embedding for 2d shape animation. IEEE Trans Vis Comput Graph 2009;15(5):867–79.