Active Comicing for Freehand Drawing Animation

Fukusato, Tsukasa
Graduate School of Advanced Science and Engineering, Waseda University : Doctral Program

Morishima, Shigeo
Waseda Research Institute for Science and Engineering : Professor

出版情報：MI lecture note series. 64, pp.6-15, 2015-09-18. 九州大学マス・フォア・インダストリ研究所
バージョン：
権利関係：

MATHEMATICAL PROGRESS IN EXPRESSIVE IMAGE SYNTHESIS 2015
Sep 25-27, 2015, Nishijin Plaza Kyushu University, Fukuoka

Active Comicing for Freehand Drawing Animation

Tsukasa Fukusato*
Waseda University

Shigeo Morishima†
Waseda Research Institute for Science and Engineering

Abstract This paper presents *Active Comicing*, a prototype sketching system that provides enhanced frame interpolation capability for freehand drawing animation. In this system, the user draws several 2D freeform strokes interactively on multiple frames, and the system automatically constructs stroke-to-stroke interpolation frames. To compose a comprehensive and coherent least-distorting interpolation, we assume input stroke has ghost points, which are additional points defined on stroke edges, and define affine transformations. In addition, the system semi-automatically guides the template motion of each stroke. For example, if the user draws an *arrow*, the system assigns the stroke moves in the direction of the arrow. To assign template motion, we compute the stroke similarity between the user's input and stroke information from a database. With this method, it is possible to generate stroke animation on each frame without stroke interpolation. By combining these techniques, the user can generate freehand animations easily and quickly.

Keywords: As-Rigid-As-Possible Stroke Interpolation, Stroke Matching, Interactive Drawing

1 Introduction

2D freehand animation enables viewers to intuitively experience artistry and feeling. Among these techniques, GIF animation (e.g., LINE's stamp and Twitter icon) has attracted worldwide attention in social networks. To present a worldview using 2D freehand animation, anime-like techniques such as flip books and motion comics are employed. However, the creation of 2D freehand animations has always been a time-consuming and skill-demanding process. Software such as *Adobe Photoshop* provides some assistance by creating animations from a small number of key frames and generating in-between frames automatically. For example, animation software might deform a shape using handles or transform simple geometric primitives. However, creating the numerous key-frames of animation, such as those in a flip book, requires significant skill and time.

Conversely, the cartoon animation industry tends to shift traditional hand-drawn techniques to a pipeline using parameterized 3D models. Although a 3D model technique reduces production costs, this approach comes at the expense of well-established cartoon animation values, such as character and expression. These models may diminish freedom, expressiveness, and the artist's commitment to the characters. It is difficult to parameterize the freedoms of pencil and paper. In short, many amateur animators, including children, find it difficult to create freehand drawing animation.

*tsukasa@moegi.waseda.jp
†shigeo@waseda.jp

Our goal is to create new shape from freehand drawing 2D shapes, sketched onto a drawing interface. In this paper, we present Active Comicing, a sketch-based interface that allows users to interpolate freehand drawing strokes without a skeleton (i.e., graffiti animation). Figure 1 shows a freehand stroke animation on a display-integrated tablet. Specifically, we have implemented a freehand stroke interpolation method, based on *As-Rigid-As-Possible Stroke Interpolation,* which does not require editing commands or special interaction modes. To address the vertex correspondence problem, in which correspondence between input strokes must be established, our system constructs a simple layer structure. To reduce the distortion of the in-between shapes, we compute the orientation information for stroke vertices (*stroke triangulation*). Moreover, this system allows users to edit the animation path of each stroke using a template motion database. As a result, we can easily create a simple 2D freehand drawing animation.

The reminder of this paper is organized as follows. Related works are reviewed in Section 2. We discuss the user interface in Section 3, and describe the main ideas underlying the proposed method's algorithms in Section 4. In Section 5, we describe the implementation details of our prototype system. We conclude the paper and discuss limitations and future work in Section 6.

Figure 1: *Prototype drawing system with Active Comicing, on a display-integrated tablet.*

2 Related Works

Recently, 3D computer graphics researchers have proposed generating 3D models from 2D drawings. In particular, Teddy [1] inflates the stroke region surrounded by a silhouette. River et al. [2] propose a method to create 2.5D models, which are hand-drawn illustration style models. Although their method can automatically estimate the depth information of each polygon section, users must configure Z-ordering and create each section from scratch. It is difficult to parameterize freehand drawings created with pencil and paper.

Drawing is simple tool that reflects the artist's creative sense. Pencil lines or brush coloring can express rich emotions or subtle charms. Live2D [4] can create rich animations based on standard linear interpolation (point-to-point interpolation) while keeping the original charms intact. However, it is necessary to determine the character pose on the frame or redraw some or all of the strokes in the model manually. In addition, significant time is required to create a mesh structure and edit mesh deformations. Cambell [5] and Baxter [6] propose intuitive approaches, whereby line drawings are interpolated in a pose-space with reduced dimensions. This method enables the subspace of a pose to be browsed. Unfortunately, it is limited to line drawings with the same number of lines, and may give unnatural results because curves are linearly interpolated.

While physically based simulations such as the mass-spring model [7] can also be used for this purpose, it tends to be slow and produces unstable shapes. Wang's approach [8] enables image deformation based on meshless rigid shape matching [10]. Moreover, Sykora et al. [12] apply this approach to elasticity-inspired character registration. With this method, it is possible to register images undergoing large free-form deformations and appearance variations. Unfortunately, they cannot directly obtain pixel or sub-pixel precision, because they embed the image into a coarse lattice. Although this method can apply a multi-scale extension, increasing the number of squares makes the overall iterative process ineffective.

In shape interpolation research, *As-Rigid-As-Possible Shape Interpolation* approaches (ARAP) have been studied [13][14][15][16][17][18]. These methods enable the volume of the stroke's interior to be maintained and produce more plausible animations by using triangle mesh structures. Furthermore, Baxter [19] has extended this method to examples-way rigid interpolation. However, this system uses polygonal boundary-based triangulation; that is, it focuses only on similarity shape morphing. In addition, it is difficult to edit the mapping of each stroke. Alexa [23] propose laplacian coordinate for shape interpolation; however, the morphing results have shirinkage. Sederberg [26] exploits intrinsic blending on a basis of interpolating the respective vertex angles and edge length. Moreover, Whited [27] develops BetweenIT, a technique for stroke interpolation from two key frames. This technique combines stroke motions constructed from logarithmic spiral vertex trajectories with stroke deformations based on curvature averaging and twisting warps. This system provides a context in which the user can guide the system in a natural manner to produce quality results efficiently. However, this system only focuses on tight in-betweens, which are drawn between two key frames that are very similar in shape.

Other approaches have processed more general shapes by considering deformations of a template model. For example, Igarashi's Spatial Keyframing [28] animates 3D objects composed of skeletons. Moreover, applying motion capture data to a single character image based on a skeleton has been studied [29][30]. However, the range of deformation is limited with these approaches. In addition, these approaches do not specify how handles should be interpolated to achieve plausible interpolations. In contrast, Sumner's [31] Mesh Inverse Kinematics system interpolates between multiple meshes. However, a non-linear inverse kinematics approach is not browsed directly.

To summarize, previous animation techniques and tools have restrictions on the types of input strokes that can be used for similarity stroke morphing, and it is necessary to redraw some or all of the strokes in the model. Therefore, we propose a method to interpolate freehand-drawing strokes. The proposed method is of great value, and can create a simple animation interactively.

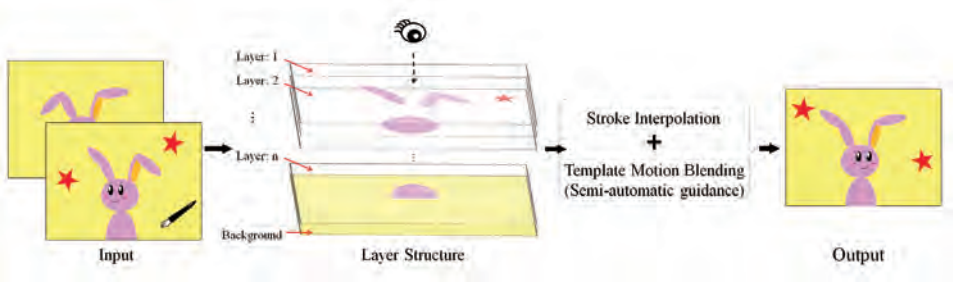

Figure 2: *System overview.*

3 User Interface

Active Comicing's physical user interface utilizes traditional 2D input devices such as a standard mouse and pen tablet. Figure 2 shows an overview of our system. In drawing mode, a user draws several 2D freeform strokes interactively on some key frames. This system has various user drawing functions such as image (e.g., jpg image file) loading functions and key frame copy function. The user can also redraw some or all of the strokes on each frame. Moreover, using onion skinning, the user can make decisions on how to create key frames based on the previous key frames in the sequence.

A stroke consists of a sequence of points on the plane, which we call stoke vertices. The stroke vertices are interpolated using a centripetal Catmull-Rom spline. In editing mode, the user can transform the strokes to edit the xy-coordinate of each stroke vertex; this translation is performed by dragging the mouse. The system automatically assigns labeling numbers (or layer number) to the strokes on each key frame based on the stroke order. The input strokes on one frame spatially correspond to those on another frame based on the labeling number; stroke-to-stroke correspondences are defined. Moreover, the user can edit the layering order by dragging and dropping layers with the mouse.

The user can easily generate a freehand stroke animation as a GIF image, using the provided animation timeline in animation mode.

4 Algorithm

In this section, we introduce the algorithm that creates the stroke animation between frames. *As-Rigid-As-Possible Stroke Interpolation* is described in Section 4.1; template motion blending is discussed in Section 4.2.

4.1 Stroke Interpolation Method

To interpolate two frames, the corresponding strokes have to have the same number of stroke vertices. The source strokes are first resampled to the number of target stoke vertices n equidistantly. Let $P = (\vec{p}_0, \cdots, \vec{p}_n)$ be the source stroke and $Q = (\vec{q}_0, \cdots, \vec{q}_n)$ be the target stroke.

For 2D interpolation technique, Sederberg [25] proposes a solution to the vertex correspondence problem, and the vertex path problem is dealt with in Sederberg [26] method, which interpolates the edge lengths and the angles between consecutive edges of polygonal curves. To ensure these blended strokes are closed without local self-intersection, they set to an equality constraint of the end positions by tweaking the edge length only; however, the final morphing results are dependent on the computation order of dihedral angels and edge length. Moreover, they cannot add some constraints, and extend this method to an invariant interpolation under similarity transformation, i.e., rotation and scale. Most shape interpolation and deformation studies have focused on 2D or 3D triangle mesh because the affine transformation of each triangle polygon can be computed easily. However, this approaches focus only on triangle mesh structure and do not determine a vertex path for stroke interpolation. Baxter et al. [19] apply a Delaunay triangulation to 2D stroke vertices, and then deform the Delaunay triangles based on ARAP. Unfortunately, the Delaunay triangulation approach is focused only on a closed stroke, and is less intuitive for interpolations between closed strokes. Specifically, it is difficult to define an affine transformation based on the source and the target stroke vertices only.

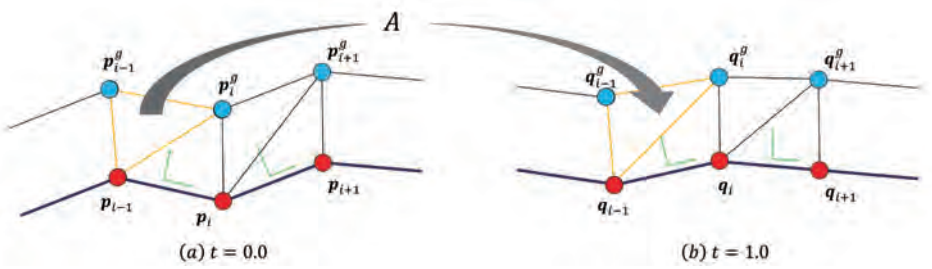

Figure 3: *Configuration of stroke, vertex (red) and ghost vertex (cyan).*

Therefore, we assume that a each stroke vertex $\vec{v_i}$ has a ghost vertex $\vec{v_i}^g \in \mathbb{R}^2$, which is placed on a certain distance along normal direction of each adjacent edge (as shown in Figure 3). Representing the ghost vertex is mainly inspired by Umetani's ghost point approach [32] and Sumner's surface tetrahedra [33]. Using the ghost vertex, we generate the triangles of the source and the target strokes in order to compute a unique affine transformation of each edge. For the corresponding triangle in each stroke shape, the system first computes the ghost vertex $\vec{v}^g$ of each stroke vertex (R_{90} denotes rotation matrix by 90 degrees):

$$\vec{v_i}^g = \vec{v_i} + R_{90}\left(\frac{\vec{v_{i+1}} - \vec{v_{i-1}}}{2}\right) \quad (1)$$
$$where \quad R_{90} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

As the result, the source and the target stroke can consist of a chain of triangles. Then, we focus on ARAP interpolation of local and global linear transformation. An affine mapping represented by matrix A transforms the source into the target triangle. The matrix A is parameterized by time $t \in \mathbb{R}$ such that $A(0.0) = I$ (identity matrix) and $A(1.0) = A$. We next deal with the 2D interpolation of the entire input strokes (the source and the target triangles). To compute a global transformation $B_i(t)$ based on local translation $A_i(t)$, we use Kaji's local error function using the polar decomposition and the exponential map [16] as follows:

$$A_i(t) = R^t_\theta \cdot \exp(t \log S) \quad (2)$$

$$E^R_i(A_i(t), B_i(t)) = \min_{s,\delta \in \mathbb{R}} \|sR_\delta A_i(t) - B_i(t)\|^2_F \quad (3)$$

where R_δ is a rotation matrix, and s is scale value. This equation measures how different $A_i(t)$ and $B_i(t)$ are as affine transformations of ith triangle. With the local error functions for each triangle, we combine them into a single global error function. If a local affine transformation can be formed by reflections, we exclude an error value of local triangle distortion from the global error function. The global error function is a positive definite quadratic form. Instead, in-between parameter t requires the solution of a linear system of equations.

In addition, using the ghost vertex, we can unify the stroke's global orientation into a counterclockwise orientation. We compute a sign area S as follows:

$$S = \frac{1}{2}\sum_i (v_{ix}v_{(i+1)y} - v_{(i+1)x}v_{iy}) \quad (4)$$

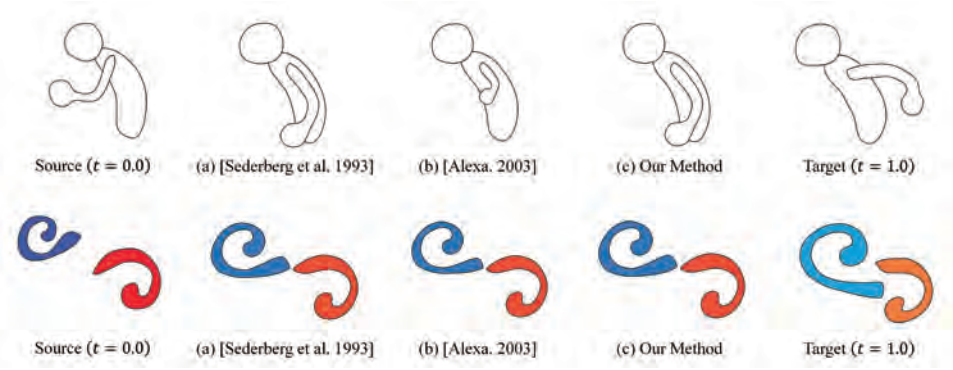

Figure 4: *Comparison. (a) Intrinsic shape interpolation [26] with* $t = 0.5$*. (b) Laplacian morphing [23]. (c) Our method.*

This computation gives a positive signed area S for a simple stroke (non-self-intersecting polygon) ($S > 0$) when the vertices are oriented counterclockwise around the polygon, and negative ($S < 0$) when oriented clockwise. However, this equation cannot be applied to complex strokes (self-intersecting polygon). It is necessary to split the complex strokes into several simple strokes.

For interpolating the line weight and RGBA color information of each stroke, we used standard linear interpolation. The system allows us to set the in-between parameter t for linear and interactive interpolation of each stroke shape. The acceptability of the computation time depends on the shape and the desired application. Figure 4 illustrates the resulting transformations from a source to a target shape. For comparison, Figure 4(a) shows Sederberg's method [26] of each vertex coordinate with $t = 0.5$, and Figure 4(b) shows laplacian morphing method [23]. Our transformation (ARAP with ghost vertex) is depicted in Figure 4(c). The results show that we have successfully reduced distortions in stroke shape transformations. In addition, we can incorporate some constraints into the global error function.

4.2 Template Motion Blending

In this section, we describe a method to animate strokes based on template effects, such as those in *Microsoft PowerPoint*. The template effects consist of affine transformations (e.g., translation matrix T, scaling matrix S, and rotation matrix R) and alpha blending.This system is formed with the origin at the centroid vertices of each stroke. By setting the stroke motion matrix (e.g., the animation path), we can interactively edit the results of the stroke animation.

In addition, we attempt to synthesize the template effects automatically. For example, when the user draws an *arrow* shape, our system assigns the stroke moves in the direction of the arrow. First, we assume that the user assigns the same effect to similar shape strokes. Therefore, we propose a technique to compute the similarity between strokes, and recommend optimum template effects based on the stroke database, which contains sets of stroke and template effect. We compute the degree of similarity between the input stroke $\vec{v}$ and the database stroke $\vec{d}$, and present template effects of highly similar strokes from the database. In image processing research, stroke similarity has been proposed. Ip et al. [34] propose affine-invariant stroke features based on stroke area and angle information. Because a single shape signature (in the form of a nineteenth-dimension histogram) records the stroke area and angle information, their similarity between stroke shapes can be computed efficiently using a signature difference. However, this approach, known as histogram

intersection, does not consider stroke's global orientation, and has difficulty representing complex stroke shapes. Therefore, to compute stroke similarities, we use rigid shape matching [8] [10] based on the centroid position of the input stroke $\vec{v_{cm}}$ and the database stroke $\vec{d_{cm}}$. We define the quadratic error function between the stroke vertices $\vec{p_i}(= \vec{v_i} - \vec{v_{cm}})$ and $\vec{q_i}(= \vec{d_i} - \vec{d_{cm}})$ as follows:

$$E = \sum_i |\vec{p_i} - sR \cdot \vec{q_i}|^2 \tag{5}$$

$$R = A_{pq}S^{-1} = A_{pq}(\sqrt{A_{pq}^T A_{pq}})^{-1}$$

$$A_{pq} = \sum_i \vec{p_i} \cdot \vec{q_i}^T$$

where n is the number of stroke vertices, s is the normalized value ($s = \sum_i |\vec{p_i}|/|\vec{q_i}|$). The optimal rotation R is the rotation portion of $A_{pq} = RS$; we compute rotation matrix $R = A_{pq}S^{-1}$, where the symmetric portion is $S = \sqrt{A_{pq}^T A_{pq}}$. The output value E provides the dissimilarity value because the value tends to be smaller if two signatures are more similar. However, it is essential to work to have the same number of vertices in a stroke. In this paper, the database strokes are resampled to the number of input stoke vertices during pre-processing.

To evaluate the performance of our similarity for stroke retrieval, we perform an experiment. The experiment is carried out to retrieve relevant strokes based on the users' sketching of the desired stroke with a stroke database of 20 strokes. The evaluation of the approach is based on retrieval accuracy (precision rate). The stroke retrieval results are shown in Table 1. For comparison, we use Ip's affine-invariant histogram approach [34]. These results show that our similarity can determine highly accurate animation template motions. Moreover, we add an editing function for relearning moving guidance. By adding $(k + 1)$'s editing data (a set of stroke and template motion) to the stroke databases, it is possible to obtain a more suitable optimum stroke motion.

Table 1: Stroke Classification Results

Number of strokes	Our Approach (%)	Ip et al. 2002 (%)
10	90.0	50.0
15	75.0	40.0
20	70.0	35.0

5 Implementation

Our prototype system is written using openFrameworks, an open source C++ toolkit. A 64bit Windows PC (Intel®Core^{TM} i7-3770 CPU@3.40GHz 8GB RAM; NVIDIA GeForce GT 620M 1GB) is used. By drawing freehand strokes on some frames ($max number = 4$), users can easily generate animation. Although all results are generated at over 30.0 fps, it is difficult to accurately measure the performance of each computation. Our results are presented in Figures 5 and 6.

The display-integrated tablet version of Active Comicing has been used to create different 2D freehand animations, mainly by computer graphics researchers and students. Our prototype system was also evaluated by users who provided individual feedback: One user stated that the hand drawing animation capabilities were more impressive and expressive than normal CG animation techniques. Other users commented that they wanted to upload information results to LINE or Twitter, and that the system could benefit from a more elastic function for editing stroke shapes. In the future, we

Figure 5: *Selected examples of deformable stroke animation, 'muscle training,' produced by our technique. (a)* $t = 0.0$ *(b)* $t = 0.25$ *(c)* $t = 0.5$ *(d)* $t = 0.75$ *(e)* $t = 1.0$.

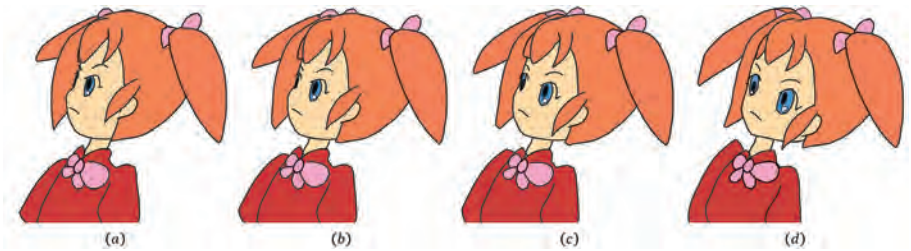

Figure 6: *Selected examples of deformable stroke animation, 'girl's looking back motion,' produced by our technique. (a)* $t = 0.0$ *(b)* $t = 0.25$ *(c)* $t = 0.5$ *(d)* $t = 1.0$.

plan to include shape deformation functions such as physical simulation in the user interface to create richer animations.

6 Conclusions and Future works

We have presented a method to interpolate and animate freehand drawing strokes. The prototype system, *Active Comicing,* enables the easy creation of simple 2D GIF animations. Moreover, the results of the stroke animation can be edited according to user preferences by template effects. It is assumed that the stroke similarity technique could also be applied to character recognition to help users find and review required freehand information. We intend to apply the proposed approach to character recognition.

In future work, we plan to increase the number of key-frame, e.g., multi-stroke morphing, and focus on color interpolation, e.g., color model or gradient color. Moreover, we recognize that sampling technique based on stroke shape will help users to create richer animations more efficiently. Therefore, we intend to study these functions. Such functions are applicable to a wide range of situations in anime production.

Acknowledgements

This project was funded in part by grants from the Japanese Information-Technology Promotion Agency (IPA) and by Research Fellowship for Young Scientists of Japan Society for the Promotion of Science (JSPS).

References

[1] T. Igarashi, S. Matsuoka, and H. Tanaka: Teddy: a sketching interface for 3D freeform design. In *ACM SIGGRAPH 2007 courses*, ACM, 21, 2007.

[2] A. Rivers, T. Igarashi, and F. Durand: 2.5D cartoon models. *ACM Transactions on Graphics (TOG)* 29, 4, 59, 2010.

[3] C.-K. Yeh, P. Song, P.-Y. Lin, C.-W. Fu, C.-H. Lin, and T.-Y. Lee: Double-sided 2.5D graphics. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 19, 2, 225-245, 2013.

[4] Live2D
`http://www.live2d.com/en/`

[5] N.D. Campbell, and J. Kautz: Learning a manifold of fonts. *ACM Transactions on Graphics (TOG)* 33, 4, 91, 2014.

[6] W. Baxter, and K. Anjyo: Latent doodle space. *Computer Graphics Forum (Proc. Eurographics)* 25, 3, 477-485, 2006.

[7] W.-C. Ma, Y.-H. Wang, G. Fyffe, B.-Y. Chen, and P. Debevec: A blendshape model that incorporates physical interaction. *Computer Animation and Virtual Worlds* 23, 3-4, 235-243, 2012.

[8] Y. Wang, K. Xu, Y. Xiong, and Z.-Q. Cheng: 2D shape deformation based on rigid square matching. *Computer Animation and Virtual Worlds* 19, 3-4, 411-420, 2008.

[9] S. Setaluri, Y. Wang, N. Mitchell, and L. Kavan: Fast grid-based nonlinear elasticity for 2D deformations. In *Proceedings of the 13rd ACM SIGGRAPH/Eurographics symposium on Computer animation*, 67-76, 2014.

[10] M. Müller, B. Heidelberger, M. Teschner, and M. Gross: Meshless deformations based on shape matching. *ACM Transactions on Graphics (TOG)* 24, 3, 471-478, 2005.

[11] M. Müller, and N. Chentanez: Solid simulation with oriented particles. *ACM Transactions on Graphics (TOG)* 30, 4, 92, 2011.

[12] D. Sỳkora, J. Dingliana, and S. Collins: As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*, ACM, 25-33, 2009.

[13] M. Alexa, D. Cohen-Or, and D. Levin: As-rigid-as-possible shape interpolation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 157-164. 2001.

[14] T. Igarashi, T. Moscovich, and J.F. Hughes: As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics (TOG)* 24, 3, 1134-1141, 2005.

[15] W. Baxter, and K. Anjyo: Rigid shape interpolation using normal equations. In *Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*, ACM, 59-64, 2008.

[16] S. Kaji, S. Hirose, S. Sakata, Y. Mizoguchi, and K. Anjyo: Mathematical analysis on affine maps for 2D shape interpolation. In *Proceedings of the 11th ACM SIGGRAPH/Eurographics conference on Computer Animation*, ACM, 71-76, 2012.

[17] R. Chen, O. Keren, and M. Ben-Chen: Planar shape interpolation with bounded distortion. *ACM Transactions on Graphics (TOG)* 32, 4, 108, 2013.

[18] Z. Levi, and C. Gotsman: Smooth rotation enhanced as-rigid-as-possible mesh animation. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 21, 2, 264-277, 2015.

[19] W. Baxter, P. Barla, and K. Anjyo: N-way morphing for 2D animation. *Computer Animation and Virtual Worlds* 20, 2-3, 79-87, 2009.

[20] S. Schaefer, T. McPhail, and J. Warren: Image deformation using moving least squares. *ACM Transactions on Graphics (TOG)* 25, 3, 533-540, 2006.

[21] A. Jacobson, I. Baran, J. Popovic, and O. Sorkine: Bounded biharmonic weights for real-time deformation. *ACM Transactions on Graphics (TOG)* 30, 4, 78, 2011.

[22] J.P. Lewis, and K. Anjyo: Direct manipulation blendshapes. *IEEE Computer Graphics and Applications* 4, 42-50, 2010.

[23] M. Alexa: Differential coordinates for local mesh morphing and deformation. *The Visual Computer*, 19, 2, 105-114, 2003.

[24] J. Hu, L. Liu, and G. Wang: Dual laplacian morphing for triangular meshes. *Computer Animation and Virtual Worlds (Proceedings of CASA)*,18, 4-5, 271-277, 2007.

[25] T.W. Sederberg, and E. Greenwood: A physically based approach to 2-D shape blending. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, 25-34, 1992.

[26] T.W. Sederberg, P. Gao, G. Guijin, and H. Mu: 2-D shape blending: an intrinsic solution to the vertex path problem. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, 15-18, 1993.

[27] B. Whited, G. Noris, M. Simmons, R.W. Robert, M. Gross, and J. Rossignac: BetweenIT: an interactive tool for tight inbetweening. *Computer Graphics Forum (Proc. Eurographics)* 29, 2, 605-614, 2010.

[28] T. Igarashi, T. Moscovich, and J.F. Hughes: Spatial keyframing for performance-driven animation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, 107-115, 2005.

[29] A. Hornung, E. Dekkers, and L. Kobbelt: Character animation from 2D pictures and 3D motion data. *ACM Transactions on Graphics (TOG)* 26, 1, 1, 2007.

[30] J. Pan, and J.J. Zhang: Sketch-based skeleton-driven 2D animation and motion capture. In *Transactions on edutainment VI*, Springer, 164-181, 2011.

[31] R.W. Sumner, M. Zwicker, S. C. Gotsman, and J. Popovic: Mesh-based inverse kinematics. *ACM Transactions on Graphics (TOG)* 24, 3, 488-495, 2005.

[32] N. Umetani, R. Schmidt, and J. Stam: Position-based elastic rod. In *Proceedings of the 13rd ACM SIGGRAPH/Eurographics symposium on Computer animation*, 21-30, 2014.

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

[34] H.H.-S. Ip, A.K. Cheng, and W.Y. Wong: Affine invariant retrieval of shapes based on hand-drawn sketches. In *Proceedings of the 16th International Conference on Pattern Recognition*, IEEE, 794-797, 2002.

[35] A.M. Namboodiri, and A.K. Jain: Retrieval of on-line hand-drawn sketches. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04)*, 642-645, 2004.

[36] K. Santosh, C. Nattee, and B. Lamiroy: Spatial similarity based stroke number and order free clustering. In *Proceedings of the 12th International Conference on Frontiers in Handwriting Recognition*, 652-657, 2010.

[37] S. Parui, and A. Mittal: Similarity-invariant sketch-based image retrieval in large databases. In *Computer Vision–ECCV 2014*, Springer, 398-414, 2014.

[38] G. Wolberg: Image morphing: a survey. *The Visual Computer* 14, 8, 360-372, 1998.

[39] H. Ochiai, and K. Anjyo: Mathematical basics of motion and deformation in computer graphics. In *ACM SIGGRAPH 2014 Courses*, ACM, 19:1-19:47, 2014.