Vol. 10, No. 3, pp. 87 – 97

Skeleton-based Adaptive In-between Generation for Hand-drawn Key Frames

Fumihito Kyota[†] **Eiji Sugisaki**[‡1]
Hock Soon Seah[‡] **Masayuki Nakajima**[†]

[†] Tokyo Institute of Technology [‡] Nanyang Technological University
[†]{kyota, nakajima}@img.cs.titech.ac.jp [‡]{SEIJI, ASHSSEAH}@ntu.edu.sg

Abstract
For improving the efficiency of 2D animation production, this paper presents a method to create in-between frames based on hand-drawn key-frames. The outlines of characters or objects on two key-frames are used as inputs. First, a skeleton linkage of the target object is automatically constructed by rasterizing each of input key-frames and then applying a pixel-based skeleton extraction method. Secondary, a pair of skeleton linkages having corresponding structure between the current key-frame and the next key-frame is constructed by applying the stroke matching algorithm. After these processes, motion transitions between the skeleton linkages are generated based on our simulation model. When the in-between frames are created only in the 2D plane, the outlines at in-between frames can be generated by a 2D deformation. In case that the in-between transitions are containing a rotation around an axis which is no perpendicular to the drawing plane, however, a 3D structure is required. For achieving such in-between transitions, our method constructs a 3D structure by inflating 2D mesh based on the input outlines. Finally, the contours from the view-point for the created 3D structure are projected onto the 2D plane during in-between transitions. In our method, we adopt the Photic Extremum Lines (PEL) to extract the 2D contours from the obtained 3D shape. In this way, we achieve the in-between creation containing spatial rotation such as hand-flipping, which has not been achieved by general ways of in-between creation method.

Keywords
2D Animation, In-betweening, 3D Model, Physical Simulation, Line Drawing

[1] He is currently working for Digital Magic Ltd., China

1 Introduction

Creating in-between frames is a fundamental process for 2D animation production. At the same time, drawing in-between frames is considered as one of the most labor-intensive procedures. For producing a smooth animation sequence, key-frame animators (they are usually more skilled than in-between animators) first draw key-frames and in-between animators then draw in-between frames according to the drawn key-frames. In the animation production pipeline, the in-between creation occupies a large proportion, approximately 60%, in total labor [1]. Although the expanding of digitization in the pipeline has progressed, the digitization is just partially replacing the traditional ways. In addition, the hand-drawing process is still considered as the typical style of 2D production pipeline and has developed the most efficient way to achieve the quality [2]. If in-between frames can be automatically generated by computing, or even partly, a huge amount of labor and time can be saved and it provides opportunity to redeploy human resources for more creative activities. From this point of view, we propose an automatic in-between creation method based on hand-drawn key-frames, which can handle dynamic structural changes.

To achieve this, our method requires animators to draw at least two key-frames and the outlines of a character or an object in each layer is used as inputs to our method. First, the skeleton linkage in each key-frame is automatically constructed based on the drawn outlines. The input contour lines are rasterized into a binary image, and then applied the skeleton extraction method [3]. Next, a linkage structure is constructed from the extracted skeleton at each key-frame. The pair of skeleton linkages constructed from two key-frames must have the same topology and the same number of joints. Therefore, we solve the inconsistency of the extracted skeletons by using the stroke matching algorithm [4].

In the next step, skeleton's transitions during in-between frames are generated by applying our simulation model [5]. In fact the simulated motion transitions of skeletons are considered as in-betweens. When the in-betweens are achieved only in the 2D plane, the outlines during in-between frames can be generated by using several 2D deformation algorithms such as [6]. However, if the in-betweens contain a rotation around z-axis which is not perpendicular to the drawing plane, a 3D structure is required for creating convincing animation otherwise the motion transition during in-betweens collapses. Therefore, we create a 3D structure by inflating 2D mesh constructed by the input outlines. The pixels of the rasterized image are converted into vertices of 3D triangle mesh. A depth value of each vertex is calculated by using the distance from the pixel to the edge of the shape. Animators can control the shape of the 3D mesh by adjusting the depth function. The 3D mesh is divided into individual parts corresponding to each link of the skeleton linkage for simulating motions appropriately.

Finally, the contours from the view-point for the created 3D mesh are projected onto the 2D plane during in-betweens. In fact, the deformed 3D mesh based on the simulation result is illustrated as 2D line drawings. In our method, the contours are extracted by using the Photic Extremum Lines (PEL) [7]. Thus, we achieve the in-between creation such as hand-flipping.

2 Related Work

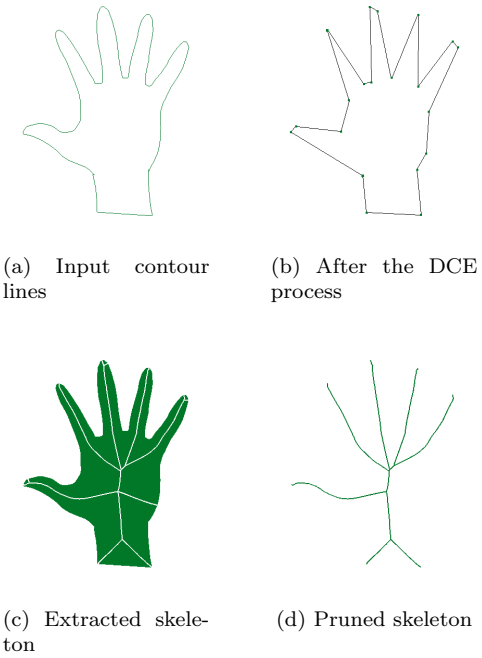
We start with describing the overview of the computer graphics techniques in cartoon animation, and then mentioning related work on simulation and motion creation.

Lasseter [1] is likely the first researcher to describe the basic principles of traditional 2D hand-drawn animation and their application to 3D computer animation. In the paper, he clearly described what cartoon animation is and what it requires of an animator. Witkin and Kass [8] also described the principles of how to create character animation using physics properties. They achieved features of the traditional animation such as anticipation, squash-and-stretch, follow-through and timing.

Recently, 3D computer graphics techniques are used in the Japanese "Anime" industry (e.g. the movie "Ghost in the Shell" and "Innocence"). The movie Appleseed is a landmark anime movie featuring hyper realistic imagery and a hybrid 2D and 3D style [9] [10]. Rademacher [11] proposed a method for a 3D structure used in cel animation. The reference hand-drawn image of an object or a character often contains various view-dependent distortions that cannot be described with conventional 3D models. Therefore, given discretionary view-dependent models, they interpolate the key-deformations specific to the new viewpoint. They thus capture the view-dependent inconsistencies of the reference drawing. We referred their concept accordingly.

Chen et al. [12] achieved a polished method to create animation automatically by in-betweening according to several 2D hand-drawing inputs. Zhou et al. [13] proposed a method to apply non-rigid and exaggerated deformations of 2D cartoon characters to 3D meshes. Kondo et al. [14] directably animate elastic objects. Their framework successfully provides realistic deformable animation and gives animators controllability and usability. Additionally, the main concept of this paper has been presented in [15].

Based on the previous work and their idea, we have proposed a simulation-based in-between creation for hair motion [5]. However, the previous simulation-based method has several limitations. First, the method is specified to hair strands and assumes that the skeletons at two key-frames have the same topology. Therefore, the method cannot handle dynamic structural changes between key-frames. On the other hand, we solve the inconsistency of the extracted skeleton by using the stroke matching algorithm [4]. Secondly, motions which can be generated by the previous simulation-based method are constrained in the 2D plane. If the motions contain 3D rotation such as y-axis rotation, the 2D deformation fails and generated in-between frames most likely collapse. Our proposed method therefore needs to have a 3D structure and it is constructed from 2D outlines to be able to generate such a 3D motion. Our method proposed in this paper can achieve the in-between creation such as hand-flipping, which has not been created by the above mentioned related work.



(a) Input contour lines

(b) After the DCE process

(c) Extracted skeleton

(d) Pruned skeleton

Fig. 1: Skeleton extraction process.

3 Skeleton Linkage Construction

Animators have to draw at least two key-frames; a source key-frame and a target key-frame. Our method creates a pair of skeleton linkages from each key-frame. For in-between creation, these skeleton linkages are required for having the same topology and the same number of joints.

3.1 Skeleton Extraction

First, a set of points on contour lines of a target object is obtained from hand-drawing key-frames. Note that hand-drawn strokes which depict the character's outline are converted into vectors represented by piecewise cubic Bezier curves in our method. Therefore, the points on the strokes can be selected automatically to create skeleton linkages. The input outlines are rasterized into a binary image, and then a pixel-based skeleton extraction method is applied to the rasterized image.

To avoid generating a too complicated skeleton, we adopt an automatic skeleton pruning technique proposed in [3]. The rasterized binary image is considered as a polygon whose vertices are boundary pixels, and then the polygons are simplified by removing vertices iteratively. This process is called Discrete Curve Evolution (DCE) [16]. After the DCE process, the contour line is partitioned into several sections by the remaining vertices. Figure 1(a) and 1(b) show a rasterized binary image and a simplified polygon by DCE, respectively.

The Euclidean Distance Transform (EDT) to the binary image for a given shape is computed by [17]. The skeleton is grown recursively by adding points that lie on dividing ridges of the EDT. Each skeleton point is corresponding to two boundary pixels which belong to different contour sections. Therefore, we establish a correspondence between contour sections and skeleton paths. Figure 1(c) shows an extracted skeleton and a reconstructed shape from the skeleton.

Finally, the pruning method called Discrete Skeleton Evolution (DSE) [3] iteratively removes skeleton end branches with smallest relevance. Figure 1(d) shows the final pruned skeleton pixels. In this way a linkage structure of these skeleton pixels are constructed.

3.2 Skeleton Graph Matching

The skeleton extraction process mentioned above is applied to the two key-frames. If the structures between these obtained skeleton graphs are not corresponding, a matching algorithm is required. This is the reason why we adopt the stroke matching algorithm [4]. By using the algorithm, we can obtain stroke correspondence automatically. Additionally, in case the correspondence is incorrect, we give animators the opportunity to correct the correspondence manually. If any stroke correspondences are not provided, we apply path similarity measures [18, 19] for further matching.

First, we obtain pairs of corresponding contour points from the stroke correspondences in both source and target key-frames. We already have a correspondence between contour points and skeleton paths in the skeleton extraction process. Therefore, the skeleton path correspondences can be obtained from the contour point correspondences through the contour-skeleton correspondences (see Figure 2).

The details of the skeleton path matching algorithm are described as follows. For each pair of corresponding contour points, the graph end path corresponding to a contour point of the pair votes to the graph end path which is corresponding to target contour point, and vice versa. The voting result determines which end node of the skeleton graph at the target key-frame to be corresponding to an end node. The most voted end node becomes the most likely candidate for the end node. If the most likely candidate of the end node also chooses the end node as first candidate, these end nodes are corresponding to each other. If the most likely candidate is already corresponding to other end node, the end node chooses another candidate from end nodes which choose the end node as first candidate by using the path similarity measure [19]. If an end node is not corresponding to any other end node, the graph path of the end node is removed from the skeleton graph.

Figure 3 depicts the end path matching. Figure 3(a) show two examples of pairs of corresponding contour point and their corresponding graph end paths. The matched pairs of graph end paths are shown by different colors in Figure 3(b). An redundant end path is removed from the skeleton of key-frame 2.

After graph end correspondences are obtained, these skeleton graphs are reconstructed to have the same topology. As shown in Figure 4, each junction node can obtain the same number of children as the corresponding junction node by this reconstruction process. If children of the junction node are not corresponding to the certain junction node in the other graph, the children of the junction node are merged to the most recent common ancestor. The index of each node is also revised.

Thus, a linkage which has a tree structure from the matched skeleton graphs is constructed. We adapt the dominant point detection algorithm [20] to each skeleton path for choosing the same number of joints.

Figure 5 represents examples of the linkage structures, which can be applied for our simulation model. These are key-frames before (Source) and after (Target) hand flipping motion. The same linkage structures are constructed by above procedures.

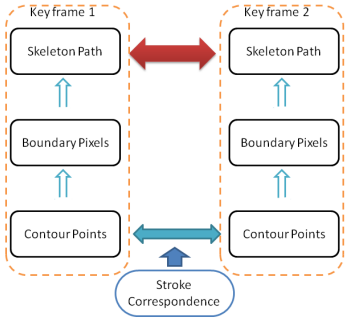


Fig. 2: Correspondences between skeleton paths are obtained from contour point correspondences.

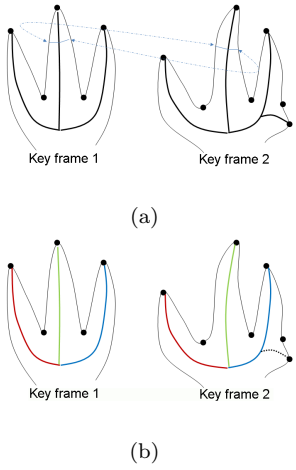


Fig. 3: The end path matching algorithm. The matching result is depicted by colors. An end path (dot line in the right skeleton) is removed.

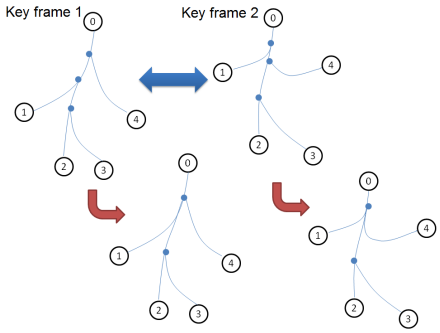
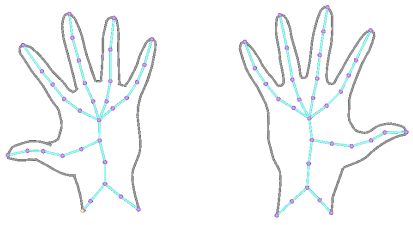
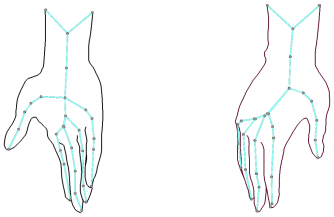


Fig. 4: The graph reconstruction algorithm. Pairs of node with the same numbers are corresponding to each other. The lower row shows the result of graphs with the same topology.



(a) Key-frame 1 (b) Key-frame 2



(c) Key-frame 1 (d) Key-frame 2

Fig. 5: Example of a skeleton linkage construction.

4 In-between Creation

Our in-between creation method generates a motion transition from the initial state of the skeleton at the source key-frame to the final state of the skeleton at the target key-frame. A state of skeleton consists of 3 components; the angles of the rotational joints, the lengths of the links, and the position and rotation of the root.

For generating a motion of the linkage structure, the motion of the root node is defined by an animator. The translation of the root is input by drawing a motion path, and the rotation of the root is specified by the axis and the angle.

Our simulation model based on multi-body dynamics [21] is applied to the constructed linkage structure. Our model can generate in-betweens which converge on the target joint angles by applying the convergence force [5]. And then, the lengths of each link are changed toward the final link lengths by linear interpolation during the simulation. In this way, we achieve creation of motion transition for the skeleton between key-frames.

After creating the motion transitions, an animator chooses desired frames as in-between frame. For user assistance, our system automatically selects feature frames as candidates of in-between frames by using the signature frame selection algorithm proposed by Yasuda et al. [22].

A screen shot of our application is shown in Figure 6. In this figure, the left window is main canvas. An animator draws input strokes and a motion path, and then the right window shows the result of in-betweens.

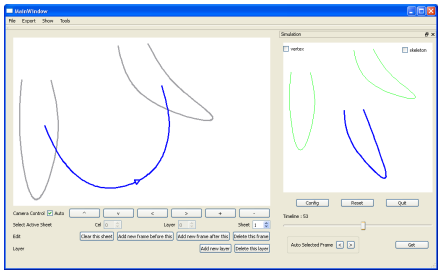


Fig. 6: A screen shot of our application. The left window is the drawing canvas. Blue line is the motion path in the window. The right window shows the result of in-betweens, the green curves are input key-frames and the blue curve is one of in-betweens.

4.1 Simulation model

We consider that the shapes of all links are cylinders which have a constant radius and the mass of each link is proportional to the length of the link. In our simulation model, every joint are rotational joints and each joint i has the spring coefficient K_s^i and the damping coefficient K_d^i. The torque exerted on the joint i is calculated as follows.

$$\tau_i = -K_s^i \left(\theta_i - \theta_i^{rest}\right) - K_d^i \omega_i \qquad (1)$$

where, θ_i and θ_i^{rest} are the current and rest angle of the joint, respectively, and ω_i is the angular velocity of the joint.

We apply the soft joint constraint [23] to a joint angle to limit the range of motion. When the joint angle exceeds the range of motion, the constraint torque is generated as a virtual spring and damping. The torque generated by the spring and the damping at a joint i is calculated as follows.

$$\tau_i^{const} = \begin{cases} -K_s^v(\theta_i - \theta_i^{Hi}) - K_d^v \omega_i & \text{if } \theta_i > \theta_i^{\mathrm{Hi}} \\ -K_s^v(\theta_i - \theta_i^{Low}) - K_d^v \omega_i & \text{if } \theta_i < \theta_i^{\mathrm{Low}} \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

where θ_i is the current angle of the joint, and the range of motion is from θ_i^{Low} to θ_i^{Hi}, and K_s^v, K_d^v are the coefficients of the virtual constraint spring and damping force, respectively. Animators can adjust these joint parameters interactively using GUI in our application.

Our simulation model can take effects of gravity and wind into account. When a wind is blowing, the external force exerted by the wind on a link is calculated by Eq. (3) [24],

$$F^{wind} = \rho W L V^2 \sin\phi \qquad (3)$$

where ρ is the density of air, W is the width of the link, L is the length of the link, V is the wind velocity, ϕ is the angle between the link and the wind direction. In fact, animators can specify the velocity and direction of the wind.

For the convergence of the motion onto the target state, we apply the joint torque toward the target angle at each joint when the position of the joint is close to the position at the target state. The torque which is called the convergence torque at a joint i is defined as follows.

$$\tau_i^{conv} = K^{conv}(\theta_i^{target} - \theta_i) \quad (4)$$

For creating more natural motion, the coefficient K^{conv} can be increased gradually.

When the trajectory of the base of the linkage is given by an animator, the spatial velocities and spatial acceleration of the base are calculated and the forward dynamics simulation is executed. The Featherstone's algorithm [21] is applied to our simulation model to solve the forward dynamics problem.

4.2 2D motion transition

The outlines of an object during in-between frames are generated according to the motion transition of the skeleton. If the transition for the root of the skeleton linkage does not contain any Z component; the motion can be achieved only in the 2D X-Y plane. In this case, we can use the 2D mesh manipulation method for deforming the input strokes such as the method Igarashi et al. [6] proposed.

Firstly, we construct a triangle mesh from the points on the input strokes and the joint positions by using Delaunay triangulation [25]. To increase the mesh density, we add a number of vertices generated in a grid pattern to the mesh. By using the mesh vertices from the joint positions as control points, the positions of the rest of vertices are calculated by a mesh manipulation method. This manipulation process should be very fast because the calculations of inverse matrix are completed in the pre-process.

Even if all joints are located on the same positions as the target at the final frame, the deformed vertices does not always lie on the same positions as the target frame. Therefore, we also construct another triangle mesh from the target key-frame and deform it. Finally, we obtain strokes during in-between frames by blending these two deformed mesh.

4.3 3D motion transition

Animators can input an axis of rotation instead of a motion path. To achieve in-betweens with rotations, we need to have a 3D structure. By inflating 2D mesh based on the input outlines, we can obtain the 3D structure for in-between frames.

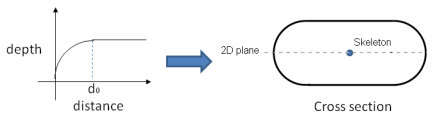


Fig. 7: Depth creation for 3D mesh construction.

Pixels of the rasterized image are converted into vertices of 3D triangle mesh. The depth value of each vertex is calculated by using the distance from the pixel to the edge of the shape. When the skeleton linkage is extracted, the distances are calculated automatically. Animators can control the shape of the 3D structure by adjusting a depth function. To provide a simple depth control to animators, the depth is defined by the following function.

$$z(d) = d_0 * \sqrt{1 - (1 - max(1, d/d_0))^2} \quad (5)$$

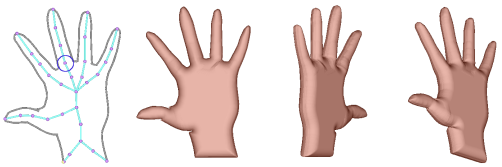
The shape for the cross section of a 3D structure created by this function is a capsule shape (Figure 7). Chosen a reference joint by animators, the radius of arc d_0 is set to the distance at the reference joint.

Figure 8 shows a result of the shape inflation. The reference joint for inflation is circle, and the thickness of the 3D geometry is defined by the diameter of the circle. Figure 9(b) also shows a 3D geometry created from Figure 9(a). To create the 3D mesh shown in Figure 9(b), two circled joints in Figure 9(a) are used as the reference joints. Therefore, the thickness of the hand and the thickness of the arm are different.

The 3D mesh is partitioned into individual parts corresponding to each link of the skeleton linkage. First, the mesh is divided by the lines perpendicular to the direction of each link, and the triangles in the divided part is associated with the link. Next, for triangles associated with more than one link or not associated with any links, these triangles are associated with the nearest link. And then, each associated region is expanded to overlap other regions. Figure 9(c) shows an example of the 3D mesh sub-division.

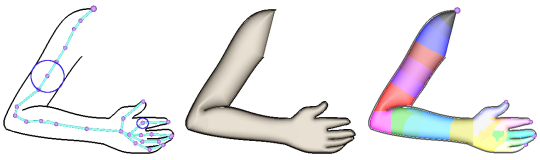
The subdivided 3D mesh is deformed according to the motion of its skeleton by traditional deformation techniques such as Skeleton Subspace Deformation (SSD) [26].

Finally, the contours from a viewpoint for the created 3D mesh are projected onto the 2D plane during in-between frames. In fact, the deformed 3D mesh based on the simulation is illustrated as 2D line drawings so the output image sequence can be utilized in 2D animation productions. In



(a) (b) (c) (d)

Fig. 8: 3D mesh construction for a hand model. (a) is a skeleton linkage of input drawing. (b), (c) and (d) are the created 3D mesh from different viewpoints.



(a) Skeleton linkage (b) Created 3D mesh (c) Subdivided mesh

Fig. 9: 3D mesh sub-divisions.

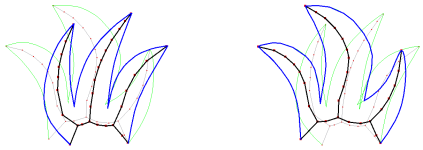
our method, we adopt the Photic Extremum Lines (PEL) [7] for extracting the contours.

5 Result

5.1 Results of 2D motion transition

Figure 10 shows a result of in-between creation of windblown grasses, and Figure 11 shows a hand with a motion of fingers. In these figures, the two hand-drawn key-frames and the extracted skeleton linkages are shown on the top of the row, and 2 key-frames and 6 in-between frames are shown in the following two rows. The skeleton linkage of the first key-frame is moved toward the second key-frame. Our algorithm can generate outlines which transform smoothly during in-between frames.

Figure 12 shows hair strands of a character while she is jumping. In the result, only a part of hairs is illustrated. The source and the target key-frames are the same frame, and the hair motion during in-between frames is created by our simulation model based on the movement on the trajectory of the character's head root point. The hair strands during its falling are spreading by a wind effect.



(a) Key-frame 1 (b) Key-frame 2

Fig. 10: Result of in-betweens for wind-blown grasses.

As shown in those figures, our method can handle more complex objects than the previous method [5] by solving the inconsistency of the skeleton linkage.

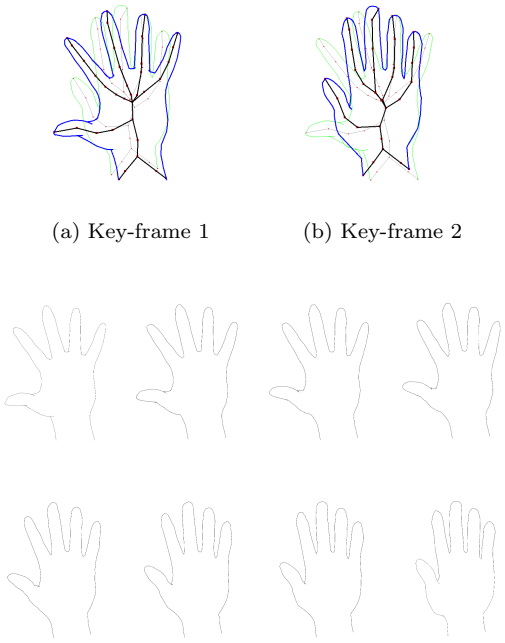
5.2 Results of 3D motion transition

Figure 13 shows the resulting in-between frames from a hand flipping motion. Figure 13(a) is a source key-frame and (o) is a target key-frame drawn by an animator. The axis of rotation and the rotation angles are also configured by the animator. To create finger motion, the animator defines several joints on fingers as flexion-extension joints and the rest angles of the flexion joints are configured.

In this result, the convergence forces are switched off during the first half of rotation by the animator's configuration. Therefore, the flexion-extension joints of the hand are moving toward and the rest angle (25 degrees of flexion) by the torsion spring of the joints. During the second half of rotation, the flexion joints are moving by the convergence forces toward the target angle (0 degrees of flexion).

We demonstrate another example "arm flipping" of in-between creation in Figure 14.

As a result, we have achieved the in-between creation such as flipping motion, which has not been created by the previous in-between creation methods.



(a) Key-frame 1 (b) Key-frame 2

Fig. 11: Result of in-betweens for a hand with motion of fingers.

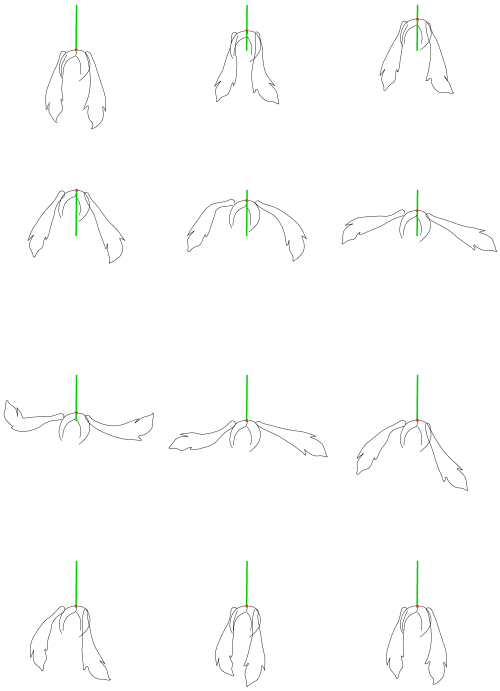


Fig. 12: Result of in-betweens for hair strands in the case of character's jumping.

6 Conclusion

We have discussed the improvement in efficiency of 2D animation production. Basically, this paper describes a method to create in-between frames from two key-frames drawn by an animator. The skeleton linkage in each key-frame is automatically constructed based on the drawn outlines. The pair of skeleton linkages extracted from the source key-frame and the target key-frame must be corresponding mutually. We have solved this problem by using stroke correspondences or path similarity. The two skeleton linkages are reconstructed according to the end path correspondences. For generating a motion sequence between two key-frames, we have successfully provided the in-between creation with our simulation model starting from the source key-frame toward the target key-frame.

Unlike the previous work, our proposed method can handle in-betweens containing a 3D rotation. To achieve this, we construct a 3D structure by inflating the input 2D shape. The constructed 3D mesh is subdivided automatically so that generic types of traditional deformation techniques can be applied. Finally, the contours from the view-point for the created 3D structure are projected onto the 2D plane during in-between frames.

In this way, we have achieved in-between creation containing spatial rotation. Our 2D/3D hybrid method can be applied to scenes more widely than the previous work.

However, there are still several future enhancements. Our simulation-based in-between creation requires several parameters such as the spring and damping coefficients of each joint, the strength of gravity and wind, and the velocity of the base of the linkage. If undesired motion is generated, animators have to adjust these parameters. Therefore, an automatic parameter adjustment is required to improve the usability of our application.

In addition, if animators want to construct more complex 3D structure, a sketch-based 3D modeling algorithm such as [27, 28, 29] shall be useful.

For creating complicated scenes, animators have to extract strokes from a composing object before applying our simulation model. If the step is automated, the effectiveness of the process of in-between creation shall be more improved.

Acknowledgment

This work has been supported by the National Research Foundation grant, which is administered by the Media Development Authority Interactive Digital Media Programme Office, MDA (IDMPO).

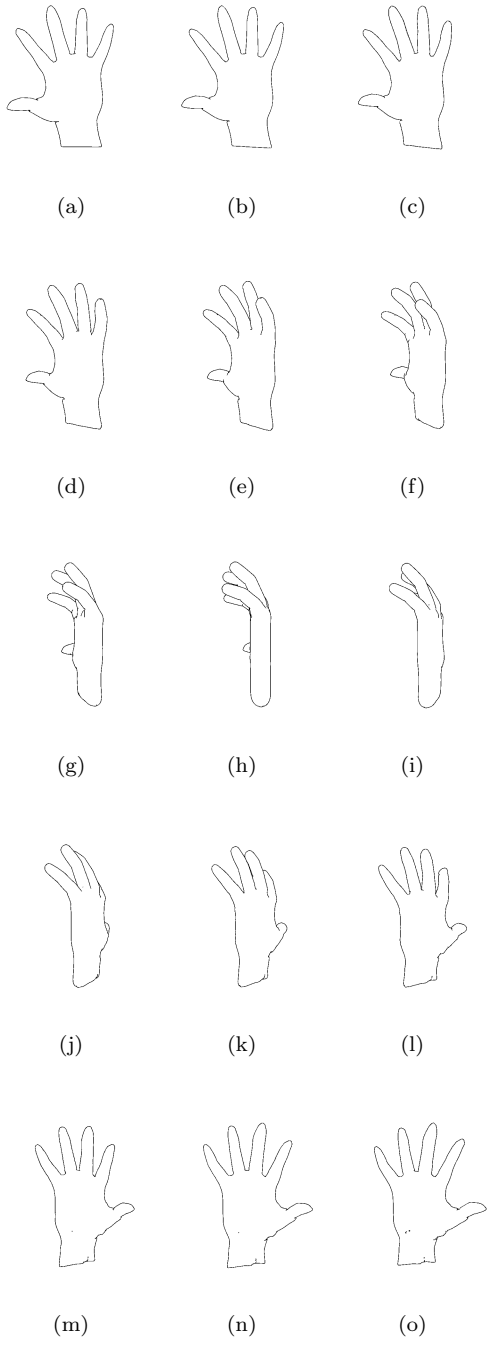


Fig. 13: Result of in-betweens for hand flipping.

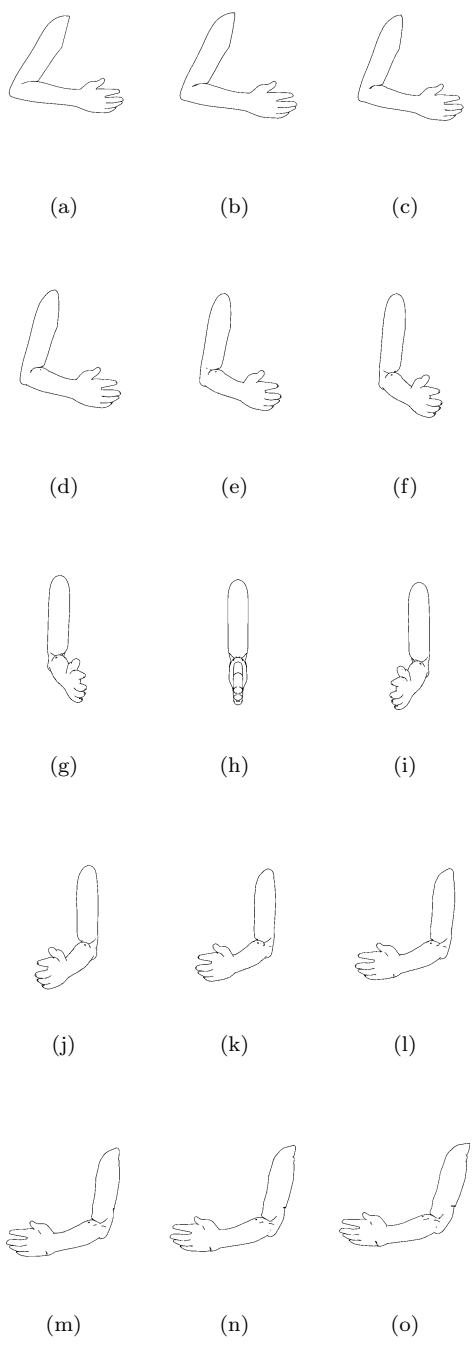


Fig. 14: Result of in-betweens for arm flipping.

References

[1] John Lasseter. Principle of traditional animation applied to 3D computer animation. *Computer Graphics*, 21(4):35–44, 1987.

[2] Quan Chen. *Computer-assisted inbetween generation for cel animation.* PhD thesis, School of Computer Engineering, Nanyang Technological University, 2008.

[3] Xiang Bai and Longin Jan Latecki. Discrete skeleton evolution. In *Proc. of EMMCVPR 2007, LNCS4679*, pages 362–374, 2007.

[4] Dongquan Liu, Quan Chen, Jun Yu, Huiqin Gu, Dacheng Tao, and Hock-Soon Seah. Stroke correspondence construction for vector based 2d animation inbetweening. In *Proc. of Computer Graphics International 2010*, 2010.

[5] Eiji Sugisaki, Fumihito Kyota, Hock Soon Seah, and Masayuki Nakajima. Simulating-based in-between creation for CACAni system. In *SIGGRAPH Asia 2009 Sketches*, 2009.

[6] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. on Computer Graphics*, 24(3):1134–1141, 2005.

[7] Xuexiang Xie, Ying He, Feng Tian, Hock-Soon Seah, Xianfeng Gu, and Hong Qin. An effective illustrative visualization framework based on photic extremum lines (PELs). *IEEE Trans. on Visualization and Computer Graphics*, 13(6):1328–1335, 2007.

[8] Andrew Witkin and Michael Kass. Spacetime constraints. In *Proc. of SIGGRAPH '88*, pages 159–168, 1988.

[9] Ken Anjyo, Michael Arias, Youichi Horry, and Yoshiyuki Momose. Digital cel animation in Japan. In *SIGGRAPH 2000 Conf. Abstracts and Applications*, pages 115–117. ACM Press, 2000.

[10] Jan Krikke. Computer graphics advances the art of anime. *IEEE Computer Graphics and Applications*, 26(3):14–19, 2006.

[11] Paul Rademacher. View-dependent geometry. In *Proc. of SIGGRAPH '99*, pages 439–446, 1999.

[12] Quan Chen, Feng Tian, Hock Soon Seah, and Jie Qiu. Motion estimation based on segmentation for key-frame inbetweening. In *Proc. of Int'l Workshop on Advanced Imaging Technology*, pages 12–17, 2006.

[13] Kun Zhou, Jin Huang, John Snyder, Xinguo Liu, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Large mesh deformation using the volumetric graph Laplacian. *ACM Trans. on Computer Graphics*, 24(3):496–503, 2005.

[14] Ryo Kondo, Takashi Kanai, and Ken-ichi Anjyo. Directable animation of elastic objects. In *Proc. of ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2005*, pages 127–134, 2005.

[15] Fumihito Kyota, Eiji Sugisaki, Hock Soon Seah, and Masayuki Nakajima. An automatic in-between creation based on hand-drawn key frames. In *Proc. of 2010 NICOGRAPH Internatinal Conference*, pages S3–4, 2010.

[16] Xiang Bai, Longin Jan Latecki, and Wenyu Liu. Skeleton pruning by contour partitioning with discrete curve evolution. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(3):449–462, 2007.

[17] Donald G. Bailey. An efficient Euclidean distance transform. In *Proc. of the Int'l Workshop on Combinatorial Image Analysis, LNCS 3322*, pages 384–408, 2004.

[18] Xiang Bai and Longin Jan Latecki. Path similarity skeleton graph matching. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30(7):1282–1292, 2008.

[19] Yao Xu, Bo Wang, Wenyu Liu, and Xiang Bai. Skeleton graph matching based on critical points using path similarity. In *Proc. of 9th Asian Conference on Computer Vision (ACCV2009)*, pages 456–465, 2009.

[20] Thanh Phuong Nguyen and Isabelle Debled-Rennesson. Fast and robust dominant point detection on digital curves. In *Proc. of 2009 IEEE International Conference on Image Processing (ICIP2009)*, pages 953–956, 2009.

[21] Roy Featherstone. The calculation of robot dynamics using articulated body inertias. *International Journal of Robotics Research*, 2(1):13–30, 1983.

[22] Hiroshi Yasuda, Ryota Kaihara, Suguru Saito, and Masayuki Nakajima. Motion belts: Visualization of human motion data on a timeline. *IEICE Trans. on Information and Systems*, E91-D(4):1159–1167, 2008.

[23] Brad Farris and Khaled El-Sawi. Physics-based robotic simulation using joint constraint. In *Proc. of the 13th Annual Conference on Industry and Management Systems*, pages 53–58, 2007.

[24] Changbo Wang, Zhangye Wang, Qi Zhou, Chengfang Song, Yu Guan, and Qunsheng Peng. Dynamic modeling and rendering of grass wagging in wind. *Computer Animation and Virtual Worlds*, 16(3-4):377–389, 2005.

[25] Der-Tsai Lee and Bruce J. Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Parallel Programming*, 9:219–242, 1980. 10.1007/BF00977785.

[26] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proc. of SIGGRAPH 2000*, pages 165–172, 2000.

[27] Ryan Schmidt, Brian M. Wyvill, Mario Costa Sousa, and Joaquim A. Jorge. ShapeShop: Sketch-based solid modeling with BlobTrees. In *2nd Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 53–62, 2005.

[28] Olga A. Karpenko and John F. Hughes. Smoothsketch: 3d free-form shapes from complex sketches. *ACM Trans. Graph.*, 25:589–598, July 2006.

[29] Alexis Andre, Suguru Saito, and Masayuki Nakajima. Single-view sketch based surface modeling. *IEICE Trans. on Information and Systems*, E92-D(6):1304–1311, 2009.

Fumihito Kyota

Fumihito Kyota received his MS degree in Computer Science and Engineering from Tokyo Institute of Technology, Japan in 2005 and BS degree in Electrical and Electronics Engineering from Yokohama National University, Japan in 2003. He has been associated with Nanyang Technological University (NTU), Singapore as research assistant from 2009 to 2010. He is currently a Ph.D student in Tokyo Institute of Technology, Japan. His research interests include but are not limited to CG and cartoon animation, computer vision and autonomous agents.

Eiji Sugisaki

Eiji Sugisaki is currently an Executive Producer and a Technical Director of Digital Magic Ltd., China. He has received a Ph.D degree from Waseda University, Japan and completed his time as a postdoc at Nanyang Technological University, Singapore. In addition, he used to be a visiting scholar at University of Illinois at Urbana Champaign, U.S. His research interests include non-photorealistic expressions, physics based hair dynamics, Data-Driven method to create a human motion, and Image Processing.

Hock Soon Seah

Hock Soon Seah is currently a Professor and Director of the gameLAB at the School of Computer Engineering (SCE) at Nanyang Technological University (NTU), Singapore. Concurrently, he is also a Co-Director of the NTU Institute for Media Innovation. Seah is the inventor and principal investigator of the Computer Assisted Cel Animation (CACAni) research on innovative drawing and animation software for traditional animation.

Masayuki Nakajima

Masayuki Nakajima received the BEE, MS, and DrEng degrees from Tokyo Institute of Technology, Japan, in 1969, 1971, and 1975, respectively. Since 1975, he has been with the Department of Imaging Science and Engineering, Tokyo Institute of Technology, Yokohama, Japan. He is now a professor in the Department of Computer Science, Faculty of Graduate School of Information Science and Engineering, Tokyo Institute of Technology, Japan. His fields of interest are computer graphics, pattern recognition, image processing, and virtual reality. He published 20 books, more than 350 papers, and 150 international conference papers.